UNIVERSITAT DE BARCELONA

Bridging Natural Language and Hierarchical Multivariate Data Visualisation to Support Data Analysis

Ecem Kavaz

UNIVERSITAT DE BARCELONA

BRIDGING NATURAL LANGUAGE AND HIERARCHICAL MULTIVARIATE DATA VISUALISATION TO SUPPORT DATA ANALYSIS

Ecem Kavaz

Supervisors:
Dr. Anna Puig Puig & Dr. Inmaculada Rodríguez Santiago

A dissertation submitted to the department of mathematics and computer science of the university of barcelona in fulfillment of the requirements for the degree of doctor of philosophy

December 2024

Abstract

Tracking and analysing the vast amounts of data generated from social networks and digital platforms presents important challenges, not only due to the overwhelming volume but also the complex relationships embedded within the data. This thesis addresses these challenges through data visualisation techniques, focusing on hierarchical and multivariate data, where visual clutter and effective use of space are key concerns. Furthermore, the rise of Visual Natural Language Interfaces (V-NLIs), also referred to in this thesis as VisChatbots, offers new opportunities to facilitate the interaction with data visualisations via natural language.

This thesis contributes to the fields of Hierarchical Multivariate Data Visualisation and Visualisation-oriented Natural Language Interfaces. Specifically, we introduce a novel categorisation algorithm to classify hierarchical data, from which we propose the most suitable visual designs for their visualisation. Additionally, we propose a new incremental design methodology for VisChatbots, called VisChat. This structured approach guides the development of chatbots integrated into visualisation platforms, establishing smooth communication among stakeholders—end users, designers, and developers—and introducing new design artefacts, such as the VisAgent persona, visualisation conversation patterns, and conversational transcripts that help guide and validate the design of the VisChatbot.

Following the VisChat methodology, we have integrated a VisChatbot into a platform for visualising hierarchical and multivariate data. To validate our proposal, we present a case study on the analysis of hate speech in online news articles, where the suitability of the proposed visualisations was evaluated, as well as the capability of the visualisation chatbot to enable users to easily explore and understand, through Natural Language interactions, both the structural relationships and the feature-based relationships within the data.

In conclusion, this thesis not only advances data visualisation techniques for multivariate hierarchical data but also establishes a framework for integrating natural language interfaces into visual analysis platforms, thereby promoting a more efficient and effective analysis of data.

Resum

Actualment, les xarxes socials i plataformes digitals presenten reptes no només pel volum de dades que generen, sinó també per la complexitat d'aquestes dades. Aquesta tesi aborda aquests reptes usant tècniques de visualització de dades, centrant-se en dades jeràrquiques i multivariades, on és especialment important transmetre la informació de manera comprensible i completa alhora. A més, el creixement de les interfícies visuals de llenguatge natural (V-NLIs), també denominades en aquesta tesi com a VisChatbots, ofereix noves oportunitats per facilitar la interacció amb visualitzacions de dades.

Aquesta tesi contribueix als camps de la visualització de dades jeràrquiques multivariades i de les interfícies de llenguatge natural orientades a la visualització. Concretament, introduïm un nou algoritme de categorització per classificar dades jeràrquiques, a partir del qual es proposen els dissenys visuals més adients per a la seva visualització. Addicionalment, proposem una nova metodologia de disseny incremental per a VisChatbots, denominada VisChat. Aquest plantejament estructurat guia el desenvolupament de xatbots integrats en plataformes de visualització, establint una comunicació fluida entre els diferents actors (usuaris finals, dissenyadors i desenvolupadors) i proposant nous artefactes de disseny, com ara el VisAgent persona, patrons de converses de visualització i transcripcions conversacionals que permeten guiar i validar el disseny del VisChatbot. Seguint la metodologia VisChat, hem integrat un VisChatbot en una plataforma per visualitzar dades jeràrquiques i multivariades. Per validar la nostra proposta, presentem un estudi de cas sobre l'anàlisi del discurs d'odi en notícies en línia, on s'han avaluat la idoneïtat de les visualitzacions proposades, així com la capacitat del xatbot de visualització per permetre als usuaris explorar i comprendre fàcilment, amb interaccions en llenguatge natural, tant les relacions estructurals com les basades en característiques dins de les dades.

En conclusió, aquesta tesi no només fa avançar les tècniques de visualització de dades per a dades jeràrquiques multivariades, sinó que també estableix un marc per integrar interfícies de llenguatge natural en plataformes d'anàlisi visual, fomentant així una anàlisi més eficient i efectiva

de les dades.

Resumen

Actualmente, las redes sociales y plataformas digitales presentan desafios no solo por el volumen de datos que generan, sino también por la complejidad de los datos. Esta tesis aborda estos desafíos mediante técnicas de visualización de datos, centrándose en datos jerárquicos y multivariados, donde es especialmente importante transmitir la información de forma comprensible y completa a la vez. Además, el auge de las interfaces visuales de lenguaje natural (V-NLIs), también denominadas en esta tesis como VisChatbots, ofrece nuevas oportunidades para facilitar la interacción con visualizaciones de datos.

Esta tesis contribuye a los campos de la visualización de datos jerárquicos multivariados y de las interfaces de lenguaje natural orientadas a la visualización. Concretamente, introducimos un nuevo algoritmo de categorización para clasificar datos jerárquicos, a partir de los cuales se proponen los diseños visuales más apropiados para su visualización. Adicionalmente, proponemos una nueva metodología de diseño incremental para VisChatbots, denominada VisChat. Este enfoque estructurado guía el desarrollo de chatbots integrados en plataformas de visualización, estableciendo una comunicación fluida entre los distintos actores (usuarios finales, diseñadores y desarrolladores) y proponiendo nuevos artefactos de diseño tales como el VisAgent persona, patrones de conversaciones de visualización y transcripciones conversacionales que permiten guiar y validar el diseño del VisChatbot. Siguiendo la metodología VisChat, hemos integrado un VisChatbot en una plataforma para visualizar datos jerárquicos y multivariados. Para validar nuestra propuesta, presentamos un estudio de caso sobre el análisis del discurso de odio en noticias online, donde se han avaluado la idoneidad de las visualizaciones propuestas, así como la capacidad del chatbot de visualizacion para permitir a los usuarios explorar y comprender fácilmente, con interacciones en lenguaje natural, tanto las relaciones estructurales como las basadas en características dentro de los datos.

En conclusión, esta tesis no solo avanza las técnicas de visualización de datos para datos jerárquicos multivariantes, sino que también establece un marco para integrar interfaces de lenguaje

natural en plataformas de análisis visual, fomentando así un análisis más eficiente y efectivo de los datos.

Acknowledgements

To my supervisors, Inma Rodríguez and Anna Puig, I am deeply grateful for your invaluable guidance and support throughout this journey. Nothing would have been possible without you; your unwavering support, expertise, dedication, and belief in me have been truly indispensable. I thank you both from the bottom of my heart—you have truly changed my life.

I extend my heartfelt appreciation to all UB collaborators. In particular, I am deeply grateful to Mariona Taulé and Montse Nofre, our linguistics partners on this journey. Your encouragement, expertise, and input have been instrumental to the success of this work.

To my family—my mom, dad, and sister—I owe you endless gratitude. You have been my greatest champions, offering unwavering support and encouragement every step of the way. Your love, patience, and belief in me have been my foundation, and this work would not have been possible without your constant presence throughout this journey.

To my boyfriend, Kemal, I am immensely grateful for standing by me through every challenge and triumph. Your encouragement gave me strength and motivation; you've been my rock through it all. Thank you—without you, I wouldn't be here.

Finally, to my friends, thank you for your continued support, and motivation. You have been there every step of the way, reminding me that I am never alone in this journey.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

This introductory chapter provides background information on this PhD thesis in the context of Data Visualisation. It highlights the issues in the field and further explores Visualisation-oriented Natural Language Interfaces (V-NLI), examining how these interfaces can enhance user interaction and understanding of complex data and visualisations. The issues explored serve as the foundation for the challenges that shape the objectives of this PhD. Additionally, we present our contributions that extend beyond theoretical insights by applying them to case studies. Finally, an overview of the organisation of the contents of this manuscript is provided, organised as a series of chapters, each contributing to the overarching goals of this doctoral thesis.

1.1 Data Visualisation of Hierarchical and Multivariate Data

In this context, many modern datasets are structured hierarchically or as networks, where elements are distributed across different levels or connected through various relationships [158]. Indeed, many research areas need to analyse **hierarchical and networked** data, such as taxonomies of language terms in linguistics [76], organisational structures in business [140], genomics in biology [41], and related comments in social media [92]. These hierarchies and networks are often vast, with both densely interconnected and sparsely distributed regions, making them difficult to navigate and analyse. Visualisations facilitate the exploration of hierarchical data by organising and presenting information to convey intricate relationships more clearly and efficiently [207, 145]. Furthermore, when this data is also **multivariate** (i.e., each data point is characterised by multiple features), solely hierarchical and network visualisations are insufficient to convey the full scope of information. The relationships between these features can reveal important details and distinctions about the data. Additional visual elements such as colour, size, and shape should be incorporated to effectively represent these relationships. Therefore, considering the datasets' size, the number of attributes (i.e., multidimensional data), and the relationships among them (i.e., correlations, dependencies, hierarchical relationships, network configurations), it is particularly important to incorporate appropriate visual representations [103].

Nevertheless, visualising hierarchical and networked multivariate data in a meaningful way comes with several problems. Achieving a meaningful representation requires a delicate balance between detail and clarity. The aim is to transmit the maximum amount of information **without losing context** while ensuring that viewers maintain an understanding of the overall structure. It involves ensuring that viewers can comprehend the relationships within the data without becoming overwhelmed. Effective visualisations must facilitate an understanding of how individual components interrelate, all while maintaining an awareness of the overarching narrative the data convey.

Another major issue is **the visual clutter**, which occurs when an excess of information is displayed simultaneously on canvases with limited spaces, making it difficult to discern meaningful patterns. Visual clutter can significantly reduce the effectiveness of visual displays, leading to cognitive overload and decreased comprehension, ultimately defeating the purpose of visualisations. Additionally, there are limited visual channels available for effectively representing multivariate data, such as colour, position, and shape, which can quickly become insufficient. While icons and glyphs can help to analyse and communicate additional information, they can also contribute to visual overload and overcrowding in visualisations[201, 135].

Moreover, in recent years, a wide range of complex and multidimensional data visualisations have been proposed in the scientific community [166], either for specific datasets in different areas of the study such as social network data [62] and biological data [6] and as more general visualisation methods [61, 213], Sankey diagrams [182], Sunburst maps [213], tree maps [173] and network graphs [14]. Not only academic research interested in data analysis to improve or examine their work; many companies also rely on data analytics to enhance their businesses and, hence, enhance the services provided to their users [165]. Consequently, visualisation methods and tools are evolving rapidly and constantly to solve new issues and adapt to changes in the field.

With these increasing varieties of visualisations, a significant dilemma arises: how to choose **the most informative** visualisation method for specific data types. By informative, we mean selecting a visualisation that preserves the context and minimises visual clutter. This is particularly challenging for non-expert users who are not accustomed to working with visualisations or analysis tools, especially when dealing with networked and hierarchical structures. Some works [159, 202] propose hierarchical visualisation taxonomies based on dimensionality (i.e., 2D or 3D) and node alignment, but identifying the most understandable visualisation remains a challenge.

Indeed, the complexity and multidimensionality of the data require a wide range of interaction possibilities, such as filtering specific data, displaying projections in 2D and 3D, examining connections between data items, clustering them, and obtaining statistics, among others [177]. Therefore, interactive platforms are necessary for users to engage with these visualisations dynamically, allowing for real-time exploration, analysis, and interpretation of complex datasets. However, these interactive visualisation platforms also come with their own set of issues as described below.

Many visualisation platforms lack the flexibility to **handle high-level visualisation tasks** effectively. We mean, visualisation tasks that involve making sense of patterns, trends and relationships in the data, which usually require more than one step to complete. While simple User Interface (UI) filters can be used to solve low-level tasks that involve interpreting individual data points or simple relationships (and can be solved in a single step), they are often insufficient for addressing complicated queries that require a more nuanced understanding of the data.

Furthermore, the **high cognitive load** associated with managing numerous filters, buttons, and options in these WIMP (Windows, Icons Menus Pointers) systems can overwhelm users, reducing their efficiency and effectiveness. Navigating various interaction combinations to accomplish specific tasks can make these platforms cumbersome and less user-friendly. These GUIs, while highly detailed and multifaceted, offer users a wide array of features and functions that can be particularly challenging to navigate on canvases with limited space [167].

Finally, many visualisation solutions are **domain-dependent** and not generic, which limits their applicability across different fields and data types. Domain-dependent solutions arise from visualisations and interactions that are specifically designed based on the characteristics of the selected data itself. This domain specificity makes it more challenging to create general, and accessible visualisation tools. Therefore, despite the potential of interactive platforms, there is a pressing need for more advanced, universally applicable, and user-friendly solutions to effectively manage sophisticated data and visualisation tasks.

1.2 Visualisation-Oriented Natural Language Interfaces

The issues described above, related to complex interactions with high cognitive demands and intricate interfaces, have been addressed by advancements in sensor technologies and Natural Language Processing (NLP), as they enable a natural interaction through gestures and conversations, respectively [107, 130]. These innovations contribute to the creation of seamless and comfortable user experiences (UX) [17]. Recently, generative large language models such as ChatGPT [136] and DALL-E [137] have given a great impulse to the NLP field. However, Natural Language Interfaces (NLIs) still face major problems. For instance, users' expectations are often very high, as they expect to communicate with the system in the same way as they interact with other human beings. Therefore, the conversational system must handle **ambiguities** that might be interpreted differently by different people [179]. Moreover, **contextual understanding** is essential for accurately interpreting user queries in NLI systems. However, current systems still require improvements in this area, often resulting in misinterpretations and less effective responses.

Specifically, Visualisation-Oriented Natural Language Interfaces (V-NLI) focus on using natural language to create and interact with data visualisations [167] [96]. Indeed, many academic research projects and popular companies, such as Tableau [37], IBM Watson [88] and Microsoft [129], have introduced and integrated them into their visualisation tools. These tools are effective and easy to learn, democratising data analysis by allowing users to interact with visualisations using natural language. Natural language removes the need to transform queries into tool-specific actions, enabling a wider range of people to focus on their analysis [167]. In this context, natural language and gestures are considered complementary input modalities to direct manipulation in WIMPs. In fact, the results of various studies have confirmed that users were more comfortable and interested in using multiple input modalities (i.e., multimodality) [147, 178]. Another major benefit of incorporating natural language in visualisations is its inclusiveness [132], as it can support blind and low-vision individuals when interacting with visualisations.

Cox et al., pioneers in this field, proposed a basic system using **form-based** interaction, meaning that users typed their queries (analytical intents) into a text box to obtain the corresponding visualisation outputs [42]. As research has advanced, more sophisticated V-NLIs have been developed, such as those referred to as **chatbot-based** (also referred to in this thesis as VisChatbots). Chatbots are intelligent conversational systems that not only provide visual outputs to users but also guide them–particularly users with less experience in visual analytics–by offering additional aids such as textual feedback, recommendations, and complex multi-stepped queries [127].

Note that the creation of a V-NLI presents numerous issues due to its multifaceted nature: integrating visual elements into a conversational interface requires careful planning to ensure a seamless user experience. Effectively leveraging **the synergies between visualisation and natural language** is key, while the specifications need to be broad enough to encompass factors such as target user profile, goals, tasks, preferred visualisations, and input/output modalities.

The literature has often concentrated on exploring and discovering specific aspects of V-NLIs without establishing a standardised methodology. The **lack of a standardised methodology** is a significant issue, as fundamental concepts and techniques in this relatively new field are still under development. Without a structured approach, integrating natural language understanding, conversational flow, and visualisation techniques can lead to significant challenges in creating effective chatbot systems for data analysis. Although some toolkits, such as NL4DV [133] and ncNet [119], facilitate the creation of V-NLIs, they fail to guarantee their robustness and efficacy, as shown by Feng et al. [59].

Moreover, most V-NLIs started out relying on limited jargon (i.e., vocabulary based on specific data), simple visualisations (e.i., bar charts, line charts) instead of **complex visualisations** and functions such as filtering and selection. For example, some works presented V-NLIs in health [108] or business [89] domains, and also delved into specific aspects of chatbots such as query recommendations [176] and ambiguity resolution [65]. These works often focused on using simple data rather than **complex data** with interconnections and on basic charts like bar charts, pie charts, and similar visualisations. However, few works used hierarchical or networked data with more complex visualisations such as tree graphs or network diagrams [171, 178]. Additionally, while new technologies like large language models (LLMs) are advancing rapidly in the visualisation domain, they still face significant issues. These include a reliance on basic visualisations and tabular data, as well as issues such as contextual misunderstandings and difficulties in handling complex visual encoding properties [122].

1.3 Open Challenges and Research Direction

This section presents the identified open research challenges and the research direction of this thesis, which can categorised into two domains; *Challenges in Hierarchical and Multivariate Data Visualisations* and *Challenges in Visualisation-oriented Natural Language Interfaces.*

1.3.1 Challenges in Hierarchical and Multivariate Data Visualisations

Visualising hierarchical and multivariate data presents several challenges. The complexity of large datasets makes analysis difficult **without losing context**. Moreover, the limited space on visualisation canvases often leads to **cluttered** and overwhelming displays. Thus, a key challenge lies in effectively visualising the entire structure or specific regions while minimising clutter, maximising the amount of information conveyed, and preserving clarity. To prevent information overload, some approaches reduce the number of graph elements displayed in the active view. These methods include filtering, interaction, and the use of hybrid or multiple views [102]. For example, some works [18, 33] displayed each attribute combination in separate views, with one main graph and others shown as thumbnails to visualise all multivariate attributes. On the other hand, some studies [32, 73] used a combination of methods to overcome the limited canvas challenge, allowing users to switch views, highlight visual patterns, and use an interactive facet legend and dynamic query filter to explore data in detail. In addition, hierarchical multivariate data is widely used across various domains [79, 134], and the effectiveness of these methods has been tested with real case studies to ensure their applicability and effectiveness in real-world scenarios. Thus, our first challenge is defined as follows:

> **Challenge 1**. *Developing a* ***novel interactive visualisation platform*** *that can visualise hierarchical multivariate data in a meaningful way, validated through real-world case studies.*

Furthermore, the literature offers an extensive visual bibliography of hierarchical visualisations [159], which currently includes 341 techniques and is updated regularly. This makes it challenging to identify **the most informative** visualisation for different hierarchical structures. Even for professionals experienced in data visualisation, this task can prove daunting. For individuals unfamiliar with the field, it becomes even more challenging. When attempting to choose the best visualisation for hierarchical data, it is important to consider the specific structure and relationships within a dataset while ensuring the overall visualisation remains comprehensible. In the context of hierarchical structures, these structures have different *shapes* depending on the connectivity degree of the internal nodes, and the number of nodes, among others. By shapes, we refer to the different distribution of nodes within hierarchies, i.e., the topology of the hierarchical structures. For example, a tree diagram may be ideal for displaying a small to medium-sized hierarchy with nodes distributed uniformly across different hierarchy levels. However, a radial layout could be more effective for representing larger datasets where most nodes are concentrated at the first level of the hierarchy, as a tree diagram may become excessively long and lose context if too many nodes are at the first level. Moreover, the literature provides some automatic layout selection tools limited to basic visualisations such as bar charts and pie charts [121, 37]. Thus, our second challenge is defined as follows:

> **Challenge 2**. *The automatic selection of* ***the most informative visualisation layout*** *according to the hierarchical structure of the data.*

Moreover, when dealing with **multivariate** data in hierarchies, additional complexities arise. Multivariate data includes multiple attributes or dimensions, which can be ordinal, nominal, interval, ratio, vectorial, temporal, and spatial. Visualising such data effectively requires techniques that can convey the relationships between these multiple variables. For example, ordinal attributes, which have a clear, ordered relationship, can be represented using gradient colours or size variations to indicate different levels. Nominal attributes, which are categorical without an inherent order, can be either abstract or concrete. In such cases, icons, glyphs, colours, or shapes can differentiate categories. Icons offer a clear and intuitive representation of concrete attributes, while shapes or colours are better suited for representing more abstract attributes. Temporal attributes, representing time, are typically visualised using timelines or time-series charts, while spatial attributes are displayed with maps or spatial plots for geographic data. Furthermore, visualising multivariate data is challenging with basic visualisations and becomes even more complex when attempting to integrate them with hierarchical visualisations [213, 135]. Thus, another challenge arises when combining hierarchies with multivariate data. The third challenge is as follows:

> **Challenge 3**. *Choosing the most informative method to visualise* ***multivariate data attributes within hierarchical visualisations****.*

1.3.2 Challenges in V-NLI

Early-stage Visualisation-oriented Natural Language Interfaces were mainly conceived as *form-based* question-answering systems, where the users ask the system questions using User Interface (UI) widgets, and the system's answer takes the form of text, a filtered visualisation and/or a new visualisation [81, 174]. Nevertheless, recent advances in Natural Language Processing have facilitated a double enhancement of these systems, both in their inner workings (NLU—Natural Language Understanding and NLG—Natural Language Generation) and in their interface. The interface is now a chatbot (embodied or not) that engages in conversation with the users to facilitate their interaction with visualisations [69, 11]. However, there has been no attempt in the V-NLI literature to specifically examine the **relationship** between the fields of Data Visualisation and Chatbots (VisChatbots). Hence, our fourth challenge is as follows:

> **Challenge 4**. *Understanding the* ***synergies*** *between Data Visualisation and Chatbot technologies to enhance interactive data exploration and interpretation.*

In the literature, there exists a considerable body of research on Visualisation-oriented Natural Language Interfaces (V-NLIs), with several surveys also available [200, 177, 80]. The works presented in these surveys usually focused on a specific application and each was dedicated to exploring a distinct dimension of the V-NLIs (e.g., ambiguity, visual narrative, multimodality) and articulating the insights derived from their analyses. However, when we turned our attention to crafting a chatbot for visualisations, we discovered a notable gap in **standardised design methodologies** for VisChatbots. Thus, our fifth challenge is defined as follows:

> **Challenge 5**. *The proposal of a* ***visualisation chatbot design methodology****.*

While a traditional data visualisation system designed with direct manipulation interfaces (WIMP – Windows, Icons, Mouse, Pointer) may facilitate the analysis of hierarchical multivariate data, the cluttered design of these interfaces requires users to invest significant time and effort to fully utilise them for analysing complex data [114]. In contrast, a V-NLI (Visual Natural Language Interface) can offer a more intuitive and efficient way to interact with hierarchical multivariate data visualisations. By allowing users to describe their visualisation goals using natural language, V-NLIs can automate the creation of visualisations, reducing the cognitive load and enabling them to focus on data exploration and analysis [163, 176]. The literature mostly focuses on using V-NLI for simple data in place of **complex data** with interconnections and emphasises more basic visualisations such as bar charts, pie charts, and line charts over **complex visualisations**. Thus, our next challenge is described as:

> **Challenge 6**. *Designing a* ***VisChatbot*** *that effectively enables users to interact with* ***hierarchical multivariate data visualisations****.*

Additionally, V-NLIs can be seamlessly integrated with **WIMP interfaces**, providing a **hybrid approach** that combines the best of both worlds. The V-NLI component can handle complex visualisation tasks and automate the creation of visualisations, while the WIMP interface can provide a familiar and intuitive way for users to interact with the visualisations [81, 178]. This hybrid approach can lead to more efficient and effective data analysis, particularly for users who are not experts in data visualisation techniques. Nevertheless, the designed VisChatbot should be aware of user interactions with the WIMP, and in reverse, the WIMP should be updated according to the visual analytics conversation maintained with the user. Thus, we present the following challenge:

Challenge 7. *Propose a **hybrid approach** by integrating the **VisChatbot** within a **WIMP interface**.*

A **case study** demonstrating the practical application of V-NLI in real-life scenarios is essential. This case study should showcase how these tools facilitate visual data analysis effectively. Moreover, evaluation methods have been proposed for conversational interfaces in general [148, 149], however, limited proposals have been made in the field of V-NLIs [65]. Additionally, the establishment of a well-defined methodology for evaluating VisChatbots is crucial. This methodology should outline clear criteria and processes for assessing the performance, usability, and effectiveness of these tools in facilitating visual data analysis. By defining standardised **evaluation metrics and procedures**, researchers and practitioners can systematically measure the capabilities and limitations of V-NLIs across different contexts and datasets. Thus, our final challenge is as follows:

Challenge 8. ***Evaluating VisChabots** with standardised metrics in real **Case Studies***

1.4 Objectives

This thesis lies in the intersection of two fields, Visualisation of Hierarchical Multivariate Data and Visualisation-oriented Natural Language Interfaces. It investigates approaches to the categorisation of hierarchical data structures, categorisation of multivariate attributes, novel visualisation for hierarchical multivariate data, VisChatbots and VisChatbot design methodologies. Below, we summarise the two main objectives in these two fields, **O1** and **O2**, and corresponding the sub-objectives of this thesis:

- **O1**: Enhancement of hierarchical multivariate visualisations, focusing on selecting the most informative layouts and methods to accurately represent complex data structures and multivariate attributes.
  - **O1.1** Proposal of categorisation algorithm that is used to select **the most informative** layout based on the hierarchy structure and the most informative way to visualise multivariate attributes.
  - **O1.2** Proposal of different Glyphs to visualise the abstract **multivariate** data to maximise the amount of information conveyed **without cluttering** the hierarchical visualisation.

- **O1.3** Proposal of a novel Data Visualisation platform for analysing hierarchical multivariate data.
- **O1.4** Validation of the proposal through a real Case Study on hate speech analysis, using annotated conversational data from online news articles, including a user evaluation.

- **O2-VisChatbot**: Development and demonstration of a VisChatbot design methodology by exploring and integrating chatbots with data visualisation techniques.
  - **O2.1** Exploring the synergies between Natural Language Interfaces and Visualisation.
  - **O2.2** Proposal of a VisChatbot Design Methodology.
  - **O2.3** Application of the proposed VisChatbot Design Methodology in a Case Study that deals with hierarchical multivariate data.
  - **O2.4** Integrate the VisChatbot in a WIMP-based interface.
  - **O2.5** Validating the VisChatbot with users.

1.5 Contributions

This section presents our contributions and related papers. First, we present our contributions concerning the objective **O1-Vis** and second, our contributions regarding the objective **O2-VisChatbot**.

1.5.1 Enhancement of hierarchical multivariate visualisations, focusing on selecting the most informative layouts and methods to accurately represent complex data structures and attributes.

We contributed by developing a categorisation algorithm that classifies hierarchies and is used to automatically select **the most informative** layout for various **hierarchical structures**, and visualise different types of **multivariate** attributes **without causing clutter** in the layout. Based on the assumption of an existing variety of hierarchical structures characterised by their shape, this contribution formalises their categorisation in *Elongated (narrow)* and *Compact (broad)* structures and argues for the adequacy of a visualisation method depending on defined attributes, such as the growing factor and the number of direct children of a node, which can be applied to any hierarchical dataset. We also contribute with a formalisation of features of multivariate data and, consequently, integrate their visualisation in a hierarchical structure. This contribution fulfils our first objective (**O1.1**) presented in the previous section. This work has been published in the Information Visualization Journal by Sage Journals.

- Kavaz, E.; Puig, A.; Rodríguez, I.; Chacón, R.; De-La-Paz, D.; Torralba, A.; Taule, M.; Nofre, M. **Visualisation of Hierarchical Multivariate Data: Categorisation and Case Study on Hate Speech** Information Visualization 2023, 22, 1, pages 31-51. `https://doi.org/10.1177/14738716221120509` — **Quartile: Q3 - Impact Factor (JCR): 2.3**.

Furthermore, to visualise these hierarchical visualisations, we developed a novel visualisation platform called Data Visualisation in Linguistics (DViL), where we visualise multivariate hierarchical data and incorporated the categorisation algorithm introduced above, which execute our objective **O1.3**. In addition, we present a case study for the DViL using conversational data about hate speech collected from online news articles. Although, it should be noted that our platform can be adapted to any hierarchical multivariate data. In this case study, we test the categorisation of hierarchical structures with a user evaluation. This meets our objective **O1.4**. This validation is published in the Information Visualization Journal by Sage Journals along with the categorisation. Also, the presentation of the DViL is published in SEPLN-CEDI-PD 2024: Seminar of the Spanish Society for Natural Language Processing.

- Kavaz, E.; Wright, F; Nofre, M.; Puig, A.; Rodríguez, I.; Taule, M.; **Introducing the Multidisciplinary Design of a Visualisation-Oriented Natural Language Interface.** Proceedings of the Seminar of the Spanish Society for Natural Language Processing: Projects and System Demonstrations (SEPLN-CEDI-PD 2024), Vol. 3729: 142-147. 7th Spanish Conference on Informatics (CEDI 2024), A Coruña (Spain), June 19-20, 2024. `https://ceur-ws.org/Vol-3729/d11_rev.pdf`.

Moreover, to demonstrate how to visualise multivariate attributes, we focused on our case study of exploring hate speech in conversational data from online news articles, aiming to determine the best methods for visualising its multivariate attributes. We conducted a preliminary user evaluation to gather insights, and the lessons learned from this evaluation have guided us in refining our approach to visualising multivariate data. This work has been published in WSCG: International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision.

- Kavaz, E.; Puig, A.; Rodriguez, I.; Taule, M.; Nofre, M. **Data Visualization for Supporting Linguists in the Analysis of Toxic Messages.**. WSCG 2021: full papers proceedings:

29. International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision, pages 59-70. `http://hdl.handle.net/11025/45010` — **Rank: B**.

Following this study, based on the formalisation described for the contribution for **O1.1**, we introduce two types of glyphs to visualise multivariate attributes: i) the *one-by-one*, where features are depicted by coloured dots placed one next to each other, and ii) the *all-in-one*, where a single pie chart represents all the features, which contributes for the objective **O1.2** and this contribution is also published in the Information Visualization Journal by Sage Journals.

1.5.2 Develop and demonstrate a VisChatbot design methodology by exploring and integrating chatbots with data visualisation techniques.

We studied the synergies between the fields of Data Visualisation and Natural Language Interaction in a scoping review. Specifically, we focus on chatbot-based V-NLI approaches. We propose an analysis framework based on the three spaces of the data visualisation pipeline, i.e., Data Space, Visual Space, and Interaction Space as well as on a characterisation of chatbots using 4 dimensions called AINT(A-Anthropomorphic, I-Intelligence, N-Natural Language Processing, and T-InTeractivity). Moreover, we extract insights and challenges that will be helpful for researchers to develop and improve V-NLIs. This contribution fulfils our objective **O2.1**. This work has been published in the Applied Sciences Journal by MDPI special edition AI Applied to Data Visualization.

- Kavaz, E.; Puig, A.; Rodríguez, I. **Chatbot-based Natural Language Interfaces for Data Visualisation: A Scoping Review.** Applied Sciences 2023, 13, 12, 7025. `https://doi.org/10.3390/app13127025` — **Quartile: Q2 - Impact Factor (JCR): 2.7**.

Additionally, we proposed a new Visualisation Chatbot methodology (VisChat Methodology). This is an iterative process that includes the three main classical phases of a development process: Analysis, Design, and Development. In our case, however, each phase introduces various stages that deal with specific characteristics of VisChatbots, encouraging collaboration among the stakeholders (end-users, developers, and designers). Specifically, we contribute with the definition of phases and stages of the VisChat methodology, three new design artefacts (a VisChat Persona template, Visualisation Conversation Patterns and VisChatbot Transcripts), as well as VisChatbot metrics for evaluation, and a case study describing a real application of the VisChat methodology for creating a chatbot for hate speech analysis on data based in conversations in social media. This contribution achieves our objective **O2.2**. This work has been submitted to Visual Informatics Journal.

- Kavaz, E.; Rodríguez, I.; Puig, A.; Simoff, S. **Bridging Natural Language and Data Visualization with VisChat Methodology.** Visual Informatics Journal. **Quartile: Q2 - Impact Factor (JCR): 3.8**

Moreover, we proposed a VisChatbot designed using the VisChat methodology we introduced previously, incorporated with our novel visualisation platform, DViL, to support users in analysing hierarchical multivariate data. We incorporated it into our existing platform where users can interact with both the chatbot and the WIMP, filter data, highlight parts of the hierarchy, change visual mapping, and enable/disable glyphs, among others. This contribution focuses on the objectives **O2.3** and **O2.4**. This work has been published in The Eurographics Association EuroVis conference and also, a part of it in the SEPLN-CEDI-PD Seminar presented above.

- Kavaz, E.; Rodríguez, I.; Puig, A.; Vives, E. **A Conversational Data Visualisation Platform for Hierarchical Multivariate Data.** in: C. Gillmann, M. Krone, S. Lenti (Eds.), EuroVis 2023 - Posters, The Eurographics Association, 2023 `https://doi.org/10.2312/evp.20231053` — **Rank: B**.

Finally, we contribute with a user evaluation, again using the hate speech case study, conducted using a well-defined methodology, and we also establish thorough evaluation metrics for analysing VisChatbots, which fulfils the objective **O2.5**. This contribution is described in Chapter 7 of this manuscript.

1.6 Connection to Sustainable Development Goals

This thesis contributes to advancing the broader goals outlined in the **2030 Agenda for Sustainable Development** of the United Nations [87]. Our research on data visualisation of hierarchical multivariate data in conversations. Our research also supports Sustainable Development **Goal 4 - Quality Education**. By enhancing the comprehension of data patterns, through our visualisation framework, we provide clear and interpretable visualisations that support initiatives aimed at increasing digital literacy and awareness. This democratisation of data facilitated by our visualisation platform and VisChatbot makes these insights available to a broader audience, including non-experts. The VisChatbot guides users through data analysis in a natural conversational manner, breaking down technical barriers and promoting engagement with the visualised data. This approach aligns with SDG 4's focus on inclusive and equitable education, ensuring that awareness of key issues reaches diverse individuals and communities. Additionally, our work indirectly supports **Goal 17 - Partnerships for the Goals** by advancing tools that facilitate more effective collaboration between technical and language experts in data analysis.

Moreover, our case study of a real problem focuses on analysing hate speech that aims to help linguists improve societal understanding and responses to harmful online behaviours by enabling them to analyse complex data with ease through visualisations. By employing advanced data visualisation techniques, we facilitate the clear communication of intricate hate speech patterns and trends. This allows linguists to more effectively study and interpret these behaviours, ultimately enhancing their ability to develop insights that contribute to broader societal understanding. This aligns with the objectives of Sustainable Development **Goal 16 - Peace, Justice, and Strong Institutions**.

Finally, our study promotes Sustainable Development **Goal 10 - Reduced Inequalities** and **Goal 5 - Gender Equality**, as the implementation of our data visualisation framework helps to identify and illuminate disparities in hate speech targeted at marginalised communities, including women and minority groups. Our approach facilitates the analysis of complex data by providing a tool that simplifies and enhances the examination of these patterns, making it easier for researchers to uncover and address discrimination.

1.7 Research Project and Funding

This doctoral thesis is part of CLIC Research Group and was supported by the SGR CLiC project (2021 SGR 00313, funded by the Generalitat de Catalunya), FairTransNLP-Language (PID2021-124361OB-C33, MICIU/AEI/10.13039/501100011033/FEDER, UE), and ACISUD (PID2022-136787NB-I00 funded by MICIU/AEI/10.13039/501100011033).

1.8 Thesis Outline

This section describes the outline of the dissertation. The thesis is divided into Parts. Parts I refer to Data Visualisation for Hierarchical Multivariate Data and Part II refer to Visualisation-Oriented Natural Language Interface to Analyse Hierarchical Multivariate Data.
The first part focuses on the following chapter:

Chapter 2 introduces the foundational concepts of hierarchical multivariate data by outlining the Data Visualisation Pipeline (DataVis Pipeline), which includes three key spaces: Data Space, Visual Space, and Interaction Space. Within these frameworks, we define essential vocabulary related to data visualisation. This will be followed by a literature review focusing on three critical areas: the visualisation of network and hierarchical data, multivariate data, and conversational data, highlighting both their contributions and the existing gaps within each domain.

Chapter 3 proposes a categorisation algorithm to classify hierarchical structures as Elongated or Compact, based on defined attributes such as the growth factor, the number of direct children of a node, and significant nodes. We will then present an analysis of this categorisation to justify the selection of the threshold values that are used in it. Additionally, we discuss appropriate visualisation layouts for each category, suggesting that Tree and Circle layouts are the most informative for Elongated structures, while Radial and Force layouts are more effective for Compact structures. Later, we will use this algorithm and integrate it with our visualisation platform to automatically select the most informative layout. Furthermore, we formalise features of multivariate data and incorporate their visualisations into a hierarchical structure. Based on this formalisation, we explore the best methods for visualising each type of multivariate data.

Chapter 4 introduces our visualisation platform, Data Visualisation in Linguistics (DViL), which is specifically designed to visualise hierarchical multivariate data. Our objective is to create visualisations of hierarchical structures that facilitate a clear analysis of parent-child relationships and feature distributions, ensuring a user-friendly experience that minimises cognitive overload while maintaining clarity. To illustrate the capabilities of DViL, we conduct a case study on hate speech data, collected and annotated from online newspapers, thoroughly presenting this data alongside its visualisation within our platform. Furthermore, we will present a user evaluation conducted using this case study to assess the categorisation we introduced in Chapter 3.

The second part consists of the following chapters:

Chapter 5 introduces the V-NLI pipeline, an extension of the DataVis Pipeline, and explores the key vocabulary associated with V-NLI. We present a literature review that examines the synergies between data visualisation and natural language interfaces, highlighting both contributions and existing gaps within the field. Concretely, we will analyse the type of V-NLIs, their inputs and outputs, the technology behind V-NLIs, and relevant design methodologies.

Chapter 6 focuses on addressing the gap in the standard design methodology for VisChatbots by proposing a new methodology called the Visualisation Chatbot (VisChat Methodology). This framework aims to guide the incremental creation of VisChatbots specifically for visual analytics processes. Our iterative approach encompasses three main classical phases of the development process: Analysis, Design, and Development. In the Analysis phase, we outline various stages, including User Analysis, Data Analysis, VisTask (visualisation task) Analysis, Visualisation Type Analysis, and Interaction Requirements Analysis. The Design phase introduces three key design artefacts: the VisAgent Persona, Visualisation Conversation Patterns, and Annotated Transcripts, all derived from the Analysis phase. Finally, the Development phase includes stages for VisChatbot Modelling, Visualisation-Chatbot Connection, and User Evaluation.

Chapter 7 presents how we utilise the VisChat Methodology presented in Chapter 6 to design our VisChatbot and integrate it with the DViL platform. We demonstrate how we apply each phase and its stages to develop our VisChatbot. Specifically, this chapter presents a user evaluation to assess the performance of our VisChatbot with users.

Chapter 8 presents the conclusions of the dissertation, summarising the main findings and contributions. Additionally, it outlines potential directions for future research, identifying areas where the current work can be extended or further developed.

Appendix A contains supplementary information for Chapter 2.

Appendix B contains supplementary information for Chapter 7.

Part I

Data Visualisation of Hierarchical Multivariate Data

Chapter 2

Background on Hierarchical Multivariate Data

In this chapter, we present a review of the current state of the art in data visualisation techniques, beginning with an overview of the Data Visualisation Pipeline based on [34, 167], detailing its three spaces: Data Space, Visual Space, and Interaction Space. This pipeline outlines the stages required to systematically transform raw data into visual representations that effectively convey information. The pipeline serves as the context in which various types of visualisations will be explored. Following this, the chapter is organised into three key areas that this PhD focuses on: (1) hierarchical and network data visualisation, (2) multivariate data visualisation, and (3) the visualisation of conversational data, which is a specific case that combines elements of the previous two. For each area, we review the related work by outlining the specific requirements of each field and then analysing the related work based on these requirements. First, we present the state of the art of hierarchical data visualisations. Next, we delve into the visualisation of multivariate data, exploring these techniques in depth to inform our subsequent visualisation strategies and presenting related work. Finally, as a real and specific case study of hierarchical data, we explore the background of visualising data related to conversations, discuss the state of the art, and identify key challenges and opportunities in the visualisation of hierarchical multivariate data within conversational contexts.

2.1 Overview of the Data Visualisation Pipeline

A well-known data visualisation process presented in the bibliography [34, 167] consists of several steps; Data Transformation, Visual Mapping, and View Transformation. Figure 2.1 details the three spaces in which the visualisation takes place—Data Space, Visual Space and Interaction Space— and the data flow through these steps constructing the visual structures and how the end-user can interact with the data involved in each step (from right to left, see the arrows in the lower part of the figure), filtering regions (**View Transformation**), changing visual parameters (**Visual Mapping**), and making more complex requests on the data (**Data Transformation**). In the following, we present the most relevant characteristics that will serve as a basis for examining the related work and the contributions of this PhD.

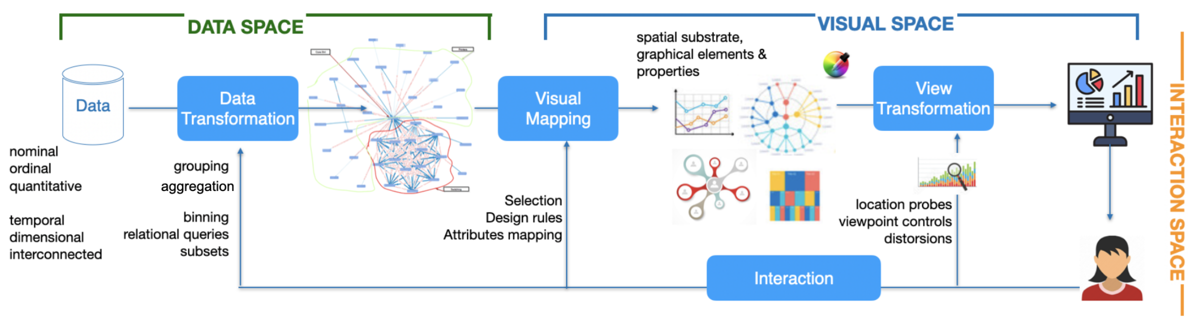


Figure 2.1: Overview of the Data Visualisation pipeline adapted from [167].

Data Space

The **Data Space** (shown in green in the upper-left part of Figure 2.1) covers the space in which raw data is manipulated through transformations before being mapped into visualisations. When the input data are in a tabular format, the **Data Transformation** stage (see the first blue square in Figure 2.1) usually offers a set of operations to filter, cluster, and aggregate data, among other functions, which can help to provide some data insights. We categorise the data according to Shneiderman's [170] framework based on the implicit nature of the data, which are: data where items are distributed along the orthogonal axis (1D, 2D and 3D), data containing items in higher dimensionalities (complex use of the context when the dimension is greater than three), trees or **hierarchical** distributions (simple connected data), and networks (complex interconnected data). The two former categories are based solely on dimensionality, considering data as a set of individual items or sampled points, in a structured or unstructured way, but without interconnections between them. However, trees and networks encode relationships between the sampled points: trees describe data containing parent-child relationships, while networks codify more complex relationships, which may be directed or undirected [158]. Moreover, in all the data categories, each point contains samples of different **attributes** that Shneiderman categorised as nominal, numerical (ordinal or quantitative) and temporal. Additionally, if these attributes are mapped into a 2D or 3D space, they are considered spatial.

These data categories help to identify the Data Transformation, which is decisive for discovering insights in the data. Classical data transformations such as grouping, aggregation, enclosure and binning temporal items are widely associated with specific data categories in the visualisation community [145]. For instance, while aggregation functions, such as mean and sum, are suitable for quantitative data, grouping is better suited to nominal and ordinal data, and binning intervals is the right transformation in the case of temporal samples [167]. In addition, some works have proposed more complex transformations of multidimensional datasets to extract meaningful subsets using relational queries [72, 119, 156]. In the case of connected structures, the topology can play an important role in the transformations, and also in the next stage of Visual Mapping [97]. For instance, extracting the largest path is a common transformation in trees, and obtaining the widest level is a more typical transformation in hierarchies. Therefore, regarding the **data types** and their different transformations, in this dissertation, we categorised data as (1) **tabular data**, i.e., data with individual and non-connected items, where classical data transformations are enough, and (2) **complex data**, i.e., high-dimensional, temporal, and interconnected data, which require more complex transformations. Moreover, both categories of data not only involve different transformations but also different strategies in the successive steps of the pipeline.

Visual Space

The second space involved in the data visualisation process is the **Visual Space** (shown in blue in the upper-middle part of Figure 2.1), which refers to how to map the data in visual structures (**the Visual Mapping Step**) and how to display them in a viewport **(the View Transformation Step**).

The Visual Mapping Step involves the definition of the next three aspects:

- The spatial substrate—i.e., the space and the layout used to map the data;
- The graphical elements—i.e., marks such as points, lines, images, glyphs, lines, etc.;
- The graphical properties—also called retinal properties, i.e., size, colour, orientation, etc. [34].

In the spatial substrate, a wide variety of layouts for displaying data have been proposed, from the simplest, such as those based on coordinate axes, to the more complex, such as those representing networks [24, 105]. In fact, the more basic and simple they are, the more they are exploited in different applications. In this research, we classify these layouts as **basic** and **advanced**. **Basic layouts** refer to chart-based layouts, which have x and y axes (e.g., bar chart, line chart, scatter plot), table-based layouts, and map-based layouts (such as a bubble map). We consider **advanced layouts** to be those that deal with higher dimensionalities (e.g., parallel coordinates) and with connections (e.g., radial tree, circle packing, network graph, sunburst diagram, and chord diagram). Refer to Figure 2.2 for examples of basic and advanced visualisation layouts, respectively.

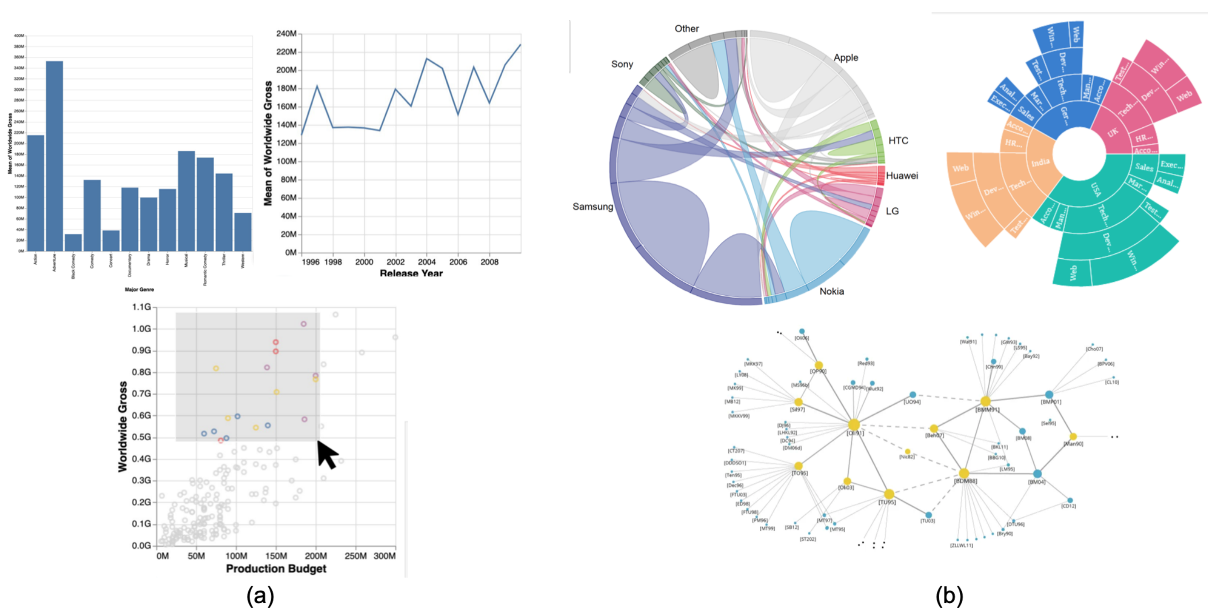


Figure 2.2: (a) Basic visualisation layouts; bar chart, line chart, and scatter plot and (b) Advanced visualisation layouts; chord diagram, sunburst diagram, and network graph.

Even with this simple classification into basic and advanced, we still have a wide range of layouts, and identifying the appropriate layout is therefore complex, especially if the users who analyse the data are not experts. Again, depending on the data types, some layouts fit better (i.e., a 3-aligned axis is a good choice to show quantitative spatial 3D data where each axis corresponds to one coordinate, and the circle packing layout fits well for simple hierarchical data). Moreover, once the layout has been selected, the next challenge is how to map the data attributes onto it. End-users can select and assign these characteristics manually, i.e., **user-defined** [183], but systems commonly use **pre-defined** layouts that only fit specific data. For example, Hoque et al., [79] maps conversational hierarchical data with specific labelled attributes (e.g., negative or positive) to a stacked bar layout that is custom-designed for their data with indentations showing the hierarchy and is therefore not flexible enough to be adapted to other data. Indeed, other approaches propose **rule-based** strategies to choose the layouts and their configuration dynamically according to the analysed data.

These rule-based approaches are commonly used in commercial systems such as PowerBI [2] and Tableau [37]. Tableau integrated the "Show Me" algorithm [121], which selects and maps layouts depending on data type (text, date, date and time, numeric or boolean), data role (measure or dimension), and data interpretation (discrete or continuous). For example, to create a bar chart, users need to place at least one quantitative attribute and one categorical attribute to the y and x axes, respectively, and then, Tableau automatically creates the bar chart. Similarly, Tableau needs two quantitative attributes to automatically create a scatter plot. Several academic studies used the "Show Me" algorithm to select visualisation methods [164, 81, 176]. More **intelligent** approaches infer the most suitable layout using some visual examples given by users [195], while others recommend layouts from among five key design choices [83], and more recently use pre-trained Neural Network (NN) models that map data to predefined chart templates [119].

Additionally, the Visual Mapping step must consider which **graphical elements** to use and their properties. There is a broad range of graphical elements (also called mark types) used to map attributes, such as points, lines, glyphs, icons and symbols. Some of them are more suitable for displaying quantitative attributes such as points, while others are better suited to nominal data, where a symbol can communicate the meaning of the data in a pictorial way [23, 103]. In this research, we explore a semantic continuum of the graphical elements which goes from the more **abstract** (e.g., points, cross, stars) to the more meaningful or **symbolic** (e.g., glyphs, icons). We also take into account the graphical properties that can enhance one's understanding of the graphical elements, such as colours, size, position, orientation, value, textures, shapes, connectivity, grouping, and animation. In addition, as in layout selection, finding adequate graphical elements for a given dataset and its properties is not a trivial task. In general, users can interactively select these graphical elements, although, as in the case of layouts, other methods have been proposed based on expert-defined rules [215] and intelligent algorithms that recommend [211], or infer the elements by means of pre-trained models with the most commonly used graphical elements [197]. Thus, in summary, we define the concept of **visual mapping identification** in terms of selecting layouts, graphical elements, and properties, using the following categories: **fixed**, **user-defined**,

rule-based (where basic rule-based methods follow a set of heuristics and make decisions based on them), and **intelligent methods** (involving the use of machine learning, artificial intelligence, or other computational techniques to enable systems to learn from data, adapt, and make decisions in a flexible and adaptive manner).

Once the visual mapping is performed, the **View Transformation** stage allows users to change the viewpoint (e.g., zooming and panning), perform location probes (to measure values in samples), and create some distortions in the image (i.e., change the projection type) [34]. Additionally, view transformation allows users to take into account multiple views simultaneously, as well as animations and others. Some view transformations emphasise data with importance-driven strategies to enhance values and regions of interest, among other factors. **Focus+Context** [180] highlights the important data (focus) while the rest of the data provide additional information on the background (context), which allows users to see the details as well as the entire perspective, without losing context. For example, imagine a line chart showing sales over time in which the peak point is highlighted (focus) but you can still see the all sales over time in the background (context). Other methods use the size of the items to show **different levels of detail** simultaneously, such as the multi-resolution approach [100], which allows users to select different resolutions to drill down and see details as needed, avoiding visual cluttering in the limited space of the canvas. For example, treemap exploits multiresolution, showing overall sales of all the continents in the outer rectangles so that the user can select a specific continent to view the details of sales of its countries in nested rectangles. We categorise the related works based on the number of views they use simultaneously **(Single/Multiple)** and the **strategy** employed to emphasise regions or parts of the view (zoom, panning, focus+context, level of detail, multiresolution, and others).

Interaction Space

Last but not least is the **Interaction Space** (shown in blue in the upper-right part of Figure 2.1), where the users interact with all the previous steps defined above. **User intents** refer to the specific goals or purposes that users have when interacting with a visual representation of data. These intents guide how users engage with the visualisation and determine what they aim to achieve, such as exploring patterns, comparing data points, or making decisions.

Visualisation tasks (VisTasks from now on) define how users will interact with visualisations to derive insights. These tasks are the actions users take to achieve their intents. Through these tasks, users conduct visual analytics and analyse their data, effectively transforming raw data into meaningful information that aligns with their specific goals. We use the main VisTasks Types reported in the literature [26]: Lookup, Locate, Browse, and Explore the data, in the following referred to as LookupT, LocateT, BrowseT and ExploreT. Brehmer et al. [26] analysed these four visualisation tasks in terms of **known/unknown** data **Location** and **Target** data.

Specifically, **known data Location** indicates that users can directly identify the part of the visualisation where they want to perform the VisTask. In contrast, **unknown data Location** means they can not explicitly identify the data location.

On the other hand, **known Target data** refers to situations in which the users know exactly what they are looking for. This data can be provided either explicitly by the user or inferred from the context of the visualisation task. An example of target data provided by the users is when they have a specific value or metric they want to find, such as *"Finding the most negative messages."*. Target data inferred from context is when the visualisation itself guides the users towards specific data points. For example, a bar chart with a clear legend allows them to easily find the value for a particular category. In contrast, **unknown target data** refers to data that the user is not initially aware of, or cannot identify directly in the visualisation, such as patterns, anomalies, and trends, among others.

Location and **Target** data in terms of **known/unknown** are exemplified in Table 2.1. All the examples included in the table are related to social network data where the messages are nodes and the connections between them correspond to responses to these messages. A thread is a set of interconnected messages. All the messages have labels indicating their level of toxicity (0 is a non-toxic message and 4 is a very toxic message). Moreover, we assume that the user first views the complete social network, before viewing the tasks, where the darkest coloured nodes are the most toxic and the lightest coloured nodes are the least toxic (see Figure 2.3).

Table 2.1: VisTasks taxonomy introduced by [26]. All the locations are coloured in blue and targets in magenta.

	Location	Target	Example
LookupT	known	known	Find the most toxic messages in the current view (messages' toxicity is visible)
LocateT	unknown	known	Find the most toxic thread in the current view (the most toxic thread is not visible)
BrowseT	known	unknown	Find a pattern in the in the current thread (the thread is visible)
ExploreT	unknown	unknown	Find trends of toxicity in the data

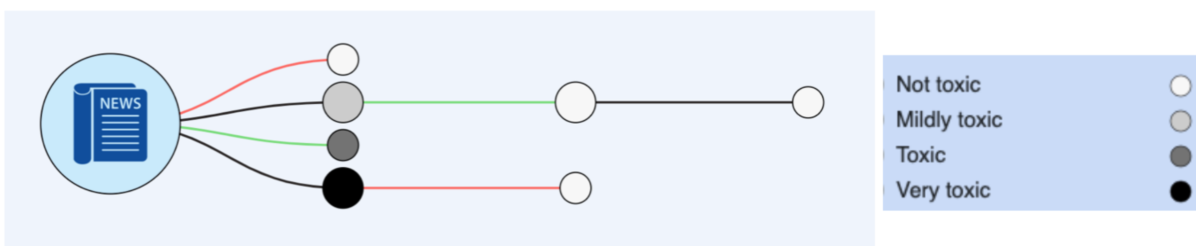


Figure 2.3: Tree layout visualising interconnected messages and showing the level of toxicity.

In addition, these VisTasks are performed through user Queries (in the following referred to as VisQueries) such as *IdentifyQ* (e.g., identifying a data point), *CompareQ* (e.g., comparing two data sets), and *SummariseQ* (e.g., summarising the whole data/visualisation).

Furthermore, there have been many attempts to categorise different interactions that are used to fulfil or solve VisTasks and VisQueries [47, 4]. Yi et al. [209] proposed seven interaction methods based on the user's intents: **select**, **explore**, **reconfigure**, **encode**, **abstract/elaborate**, **filter**, and **connect**. Select is used for marking data points choosing data points, and layouts, while explore refers to navigating through the data, including functions such as zooming and panning. Reconfigure can be used to swap layout attributes on the x and y axes or can use an algorithm to cluster some data points together in a network visualisation. Encode is used to assign or change graphical properties in terms of colour, size and shape. Abstract/elaborate displays details on demand such as collapsing/drilling down on a visualisation. The filter method shows data that fulfil a given condition. Finally, the connect method highlights the relationships between data items. These interaction types defined are either directly or indirectly related to all VisTasks and VisQueries. These interactions facilitate users in achieving their specific goals by enabling various VisTasks and VisQueries related to data exploration and analysis. Here are some examples illustrating how these interactions are applied:

- Select is crucial for LookupT, allowing users to pinpoint and examine individual data points. It also aids ExploreT by helping users narrow their focus during exploratory analysis and aligns with IdentifyQ by enabling detailed views of specific data points. Additionally, Select supports CompareQ by facilitating the comparison of selected items.
- Explore enhances BrowseT by allowing users to navigate through and visually scan large datasets. In ExploreT, it supports in-depth investigation and analysis of data patterns and relationships.
- Reconfigure plays a role in CompareQ by altering the arrangement of data to facilitate side-by-side comparisons. It also assists LocateT by modifying the layout to reveal item positions and supports ExploreT by enabling the rearrangement of data elements for deeper analysis.
- Encode supports LocateT by helping users differentiate and identify specific items through visual attributes. It also facilitates CompareQ by enabling comparisons based on encoded features and aligns with SummariseQ by aiding in the visualisation and aggregation of summarised information.
- Abstract/Elaborate is essential for ExploreT, allowing users to dynamically show or hide details and view broader patterns. It also supports SummariseQ by enabling users to collapse or expand data to summarise key insights effectively.
- Filter is useful in LocateT for isolating and focusing on specific subsets of data, making it easier to identify particular items. In BrowseT, it helps by displaying only relevant data points according to specified criteria.

- Connect supports ExploreT by linking related items and revealing patterns. It also aids CompareQ by showing how different data points or trends are interconnected, enhancing the comparison process.

Note that the visual analytics process may involve a sequence of VisTasks, each comprising a set of VisQueries to solve user intents. For instance, in the example of the toxicity of comments in online news articles, where the darker the message, the more toxic it is, the user notices several areas of the visualisation with unusual black&white patterns. That is, areas with dispersed non-toxic messages are in the middle of many very-toxic messages (many black nodes and few white nodes). These areas could be potential locations of interesting non-toxic messages to analyse. Then, the user 1) *"Zooms-in on this area"* (*IdentifyQ*, Interaction type: Explore). Here, the tool enables the user to zoom into specific data points or regions of interest for detailed examination. Once in the zoomed view, the user wants to perform a further exploration of additional details about non-toxic nodes: 2) *"Finding the attributes of non-toxic nodes"* (SummaryQ, Interaction type: Abstract/Elaborate and Filter), viewing them in a separate pop-up window. Finally, 3) *"Comparing with the toxic ones"* (*CompareQ*, Interaction type: Abstract/Elaborate), opening another chart. Finally, thanks to the exploration task (ExploreT) that consisted of several queries, the user validates the hypothesis that even in the most toxic message threads, there are users who give positive opinions.

Finally, we consider that users can utilise all these methods through different interaction styles. In this thesis, we consider a coarse two-labelled categorisation: Basic and Advanced. **Basic** styles refer to the WIMP (Windows, Icons, Mice, Pointer), while **Advanced** styles involve techniques such as Virtual Reality (VR), Augmented Reality (AR) and Natural Language.

In summary, we defined the following terms, clearly outlined in Table 2.2, which will serve as the context for exploring different types of visualisations (a more detailed summary can be found in Appendix A). It should be noted that in this dissertation, we primarily focus on the Visual and Interaction Spaces. Specifically, our emphasis is on complex data with advanced visualisations, symbolic graphical elements, rule-based visual mapping identification, multiple view transformations, and advanced interaction styles.

Data Space		Visual Space					Interaction Space	
Data Transformation	**Data Type**	**Visualisation Type**	**Graphical Elements**	**Visual Mapping Identification**	**View Transformation**	**Interaction Style**	**VisTasks**	**Interaction Style**
Filter	Tabular	Basic	Abstract (lines, points, bar)	Fixed/Pre-defined	Single	Basic-WIMP	LookupT	Basic-WIMP
Cluster	Complex	Advanced	Symbolic (glyphs, icons)	User-defined	Multiple	Advanced-NL	LocateT	Advanced-NL
Aggregate				Rule-based			BrowseT	
				Intelligent			ExploreT	

Table 2.2: Summary of Data, Visual, and Interaction Spaces.

2.2 Visualisation of Network and Hierarchical Data

In the following, we will discuss the requirements for effective hierarchical and network visualisations. We will then review the existing work around these criteria and conclude with a summary.

2.2.1 Networked and Hierarchical Data Visualisation Requirements

Hierarchical and networked visualisations are used to represent graph structures, typically in a two-dimensional plane ($\mathbb{R}^2$) or in three-dimensional space ($\mathbb{R}^3$), although this dissertation focuses on the 2D plane. In these visualisations, networked and hierarchical data are displayed as graphs, where nodes—represented by points, disks, rectangles, text, or other shapes—are connected by edges, visualised as curves or straight lines. We define a visualisation **canvas** of area $W \times H$, where W and H are the width and height in pixels. Within this canvas, the main problem to solve is projecting the graph structure into the 2D plane. This process involves several challenges while designing effective layouts: arranging nodes and edges to optimise both clarity and informational value, ensuring maximum information transmission without losing contextual relevance, reducing visual clutter in the available space and channels, and selecting an optimal layout that best suits the characteristics of the data being visualised.

In the following, we will explain the principal requirements of hierarchical and network visualisations. The main requirements are related to a) visualising maximum information while preserving context, b) ensuring non-cluttered visualisations, c) selecting the best visualisation layout based on the data and the associated VisTask, and d) prioritising domain-independent visualisations to make them more widely applicable.

(A) Visualising maximum information while preserving the context is a delicate balance. If too much information is displayed at once, it can become overwhelming and difficult to interpret.

The goal is to maximise the amount of data shown while keeping the context clear. The challenge here is to represent multivariate data within the nodes and edges of a hierarchy or network in a way that preserves the structural context and relationships while ensuring that the multiple variables are visually encoded without overwhelming the user. This problem can be broken down into several concrete criteria:

(1) **Node Representation:** All nodes in a graph should be represented in simplified or detailed ways depending on the context, allowing the viewer to grasp key information without feeling overwhelmed. (2) **Relationship Representation:** The relationships between nodes should be shown either explicitly (e.g., with lines) or implicitly (e.g., through spatial arrangements), helping users understand more easily the connections between elements. (3) **Consistency in Size and Length:** Similar lengths and sizes should indicate the same meaning, maintaining consistency and helping users interpret the data structure clearly. For example, node sizes in hierarchical layouts can either be uniform or reflect the number of descendant nodes. (4) **Scalability:** The design should be scalable, allowing for the compaction of information such as grouping related nodes or using multiple views to divide the visual information to ensure the visualisation remains clear and accessible as more data is added.

(B) Non-cluttered visualisations is another challenge to overcome in creating effective hierarchical visualisations, as excessive detail or disorganised elements can obscure the intended message. In hierarchical and networked data visualisations, it is imperative to maintain clear and intuitive parent-child relationships, along with distinct levels in hierarchical views like tree diagrams. For instance, in network visualisations, node-link diagrams can easily become overwhelmed with excessive nodes and edges, leading to visual congestion that distorts the underlying relationships and structural patterns, thereby diminishing the impact of the visualisation. Maintaining the **clarity** and **simplicity** in hierarchical visualisations is essential for helping users grasp the structure and relationships within the data [109]. For instance, a tree layout illustrating online conversations with **clear and concise labels** and **appropriate spacing** allows users to easily identify relationships between conversations. Additionally, maintaining **readability** involves **avoiding overlapping** nodes and ensuring that text remains legible as the hierarchy becomes more complex. This helps preserve the visual flow and ensures that each part of the structure can be easily interpreted.

Therefore, the following principles should be followed to avoid clutter in hierarchical visualisations: (1) **Nodes should be distributed evenly across the canvas** to reduce overcrowded or empty areas, ensuring a balanced and clear layout. (2) **Nodes must not overlap either in their projection in the 2D plane or their scaling in the WxH canvas**, as doing so can confuse viewers and obscure the relationships between elements. (3) **Edges should not overlap**, meaning crossing edges must be avoided to prevent visual clutter and maintain clear connections between nodes. (4) **Proximity between connected nodes should be preserved** to convey their logical relationship and strengthen the visual understanding of the data. (5) **Maximise the angles between edges of the same node** to ensure that each edge is distinguishable and the visualisation remains legible. (6) **Empty spaces between nodes should be preserved** to help clarify the structure of the data, allowing for easier identification of relationships and better overall comprehension.

(C) Selecting the best visualisation layout for the data and the associated VisTask is crucial, as not all hierarchical data share the same structure. Some datasets feature more nodes directly connected to the root, resulting in a compact layout, while others have nodes distributed along a vertical axis, creating a more elongated shape. Choosing the right layout is important not only for avoiding clutter but also for ensuring the data is represented in a visually compelling and easily comprehensible manner. Since hierarchical structures and network configurations vary widely and datasets can grow in size, a scalable design must be adaptable to handle both structural diversity and increasing complexity. Therefore, when choosing a layout, it is important to consider its ability to manage large, deep, or wide datasets effectively. Poor **scalability** can result in overcrowded and confusing visualisations, where critical information is obscured by excessive detail [105].

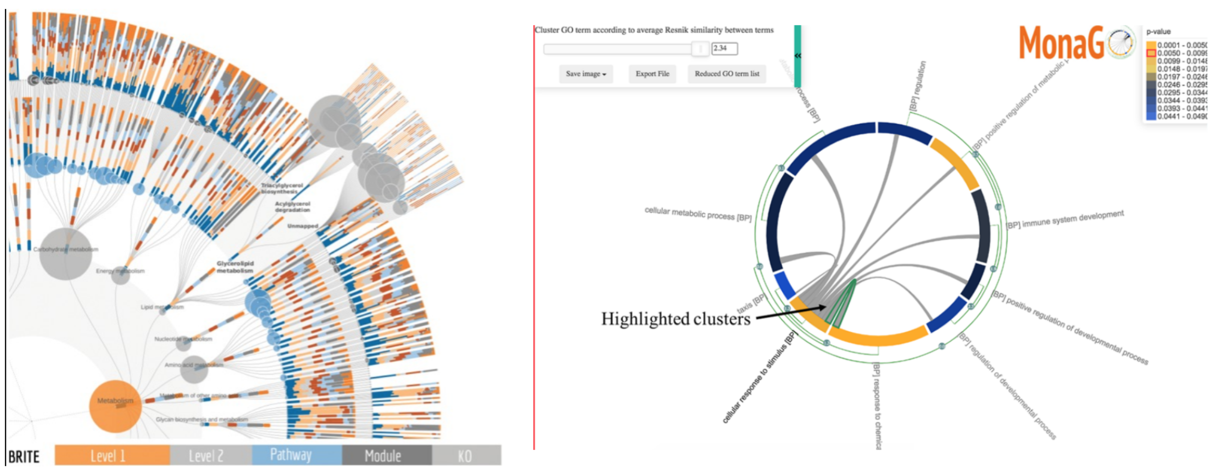
(D) Domain-dependent vs domain-independent visualisations is another aspect that should be considered. **Domain-dependent** visualisations are designed with a specific dataset or application in mind, meaning they are tailored to the unique characteristics and requirements of a particular type of data. These visualisations often cannot easily accommodate other datasets, as they rely on predefined assumptions and structures. For instance, uploading a different dataset might require significant modifications or may not be supported at all. While such visualisations can be highly effective for their intended use cases, they lack the flexibility to be universally applicable. On the other hand, **domain-independent** visualisations are designed with generality in mind, allowing them to adapt to a wide variety of datasets. These tools are built to work without requiring extensive changes or customisation, making them versatile and suitable for analysing diverse types of data. Such flexibility can be more advantageous as it appeals to a broader audience.

2.2.2 Review of the Literature

Hierarchical and network data visualisation is a critical area of study within the broader field of data visualisation, driven by the growing complexity and volume of interconnected data [21]. In the following literature review, we will explore various hierarchical and network data visualisation techniques, assessing their strengths, limitations, and suitability.

Several researches explored visualisations of **hierarchical structures** [44, 205, 61, 10, 28, 50, 208, 173, 204]. In the following, we highlight the most relevant ones for this thesis. For instance, Darzi et al., [44] designed Functree2, a radial layout to visualise omics data, such as genome, and proteome (see Figure 2.4.a). They **selected** a **pre-defined** radial layout as they presumed that tree layout was insufficient with a large number of nodes and **scaled** poorly. However, as illustrated in the figure, the implementation of coloured rectangles to represent additional attributes **clutters** the visualisation and impacts **readability**. Overlapping nodes and the placement of outer nodes can make the visualisation challenging to interpret. The paper notes that they addressed these issues by incorporating **interactive elements** and additional charts and bar graphs. These supplementary visuals help summarise and clarify the extensive information displayed in the radial layout, enhancing the overall user experience and making complex data more accessible. Moreover, they stated that the Functree2's ability to handle generic reference trees makes it a **domain-independent** solution. While particularly designed for bioinformatics pipelines, it is flexible enough to visualise diverse hierarchical data, demonstrating its applicability across various domains.

Similarly, MonaGo [205] handles data related to genomics, transcriptomics, proteomics, and metabolomics. MonaGO uses a **pre-defined** chord diagram to effectively visualise clusters and similarities among entities (see Figure 2.4.b). The visualisation employs colour-coding based on significance values, with arc lengths representing the number of relevant items. Green arcs on the diagram indicate potential hierarchical clusters, while numbers on these arcs show the percentage of shared items between clusters. Grey links within the diagram connect clusters with common items. Moreover, explanations of the clusters are written next to the circle. By hierarchically clustering similar entities, MonaGO **reduces clutter** while trying to present **maximum information without losing the context**. It allows users to collapse or expand clusters, which also helps with the **readability** and **clarity**. They also use **multiple views** to display additional information, helping to avoid clutter. However, **scalability** can be an issue here, as a large amount of data can lead to overlapping cluster explanations, making it difficult to read and interpret the information clearly. Also, **interactivity** allows users to dynamically explore and manipulate the data, enabling them to focus on specific subsets or adjust the clustering parameters to better understand the underlying relationships. Moreover, MonaGO is considered a **domain-dependent** tool, as it is designed to assist biologists in exploring biological data. However, it is not stated whether it can be adapted for other domains.



(a) (b)

Figure 2.4: (a) Functree2 [44] and (b) MonaGo [205]

In addition, Graphia [61] offers a comprehensive solution for the visual analysis of the vast amounts of quantitative and qualitative data generated from genomic, proteomic, metabolomic, and cellular studies (see Figure 2.5.a). The core functionality of Graphia revolves around its ability to perform **data transformation** by calculating correlation matrices from any tabular matrix of continuous or discrete values, which can then be visualised in **pre-defined** large network graphs in both 2D and 3D spaces. However, the visualisation can become **cluttered** as the large amount of data may lead to overlapping clusters. To help with **visualising maximum information without losing the context** they use complementary bar charts and heat maps. But even with these, the main visualisation can be hard to interpret. Nevertheless, the flexibility and interactivity offered by Graphia, including algorithms for graph transformation and attribute visualisation, make it a powerful tool for analysing complex biological data. Graphia is designed with biological data in mind, but it is a **domain-independent**, and it can be used to analyse network data from any source.

Moreover, GrouseFlocks [10] focused on taking an input hierarchy and showing other related hierarchies of it (e.g. from all movies to action movies) using several **pre-defined** layouts, concretely, they mix implicit and explicit types using both tree and circle packing. By allowing users to see several different possible hierarchies on the same graph, the tool helps users investigate graph hierarchy space instead of a single, fixed hierarchy. While it is interesting to explore hierarchies inside others using several layouts and having a zoom functionality, circle packing gets crowded and begins to visualise hard-to-follow nesting when the data structure gets deeper and wider causing poor **scalability because of the clutter**. Which makes it hard to **read** and lose its **context with too much information displayed at once**. For instance, it can be observed in the Figure 2.5.b, the example at the top demonstrates the theory effectively. However, as seen in the real-life application at the bottom, it becomes cluttered with larger data. Furthermore, it is not designed for a specific domain but rather as a **domain-independent** tool.

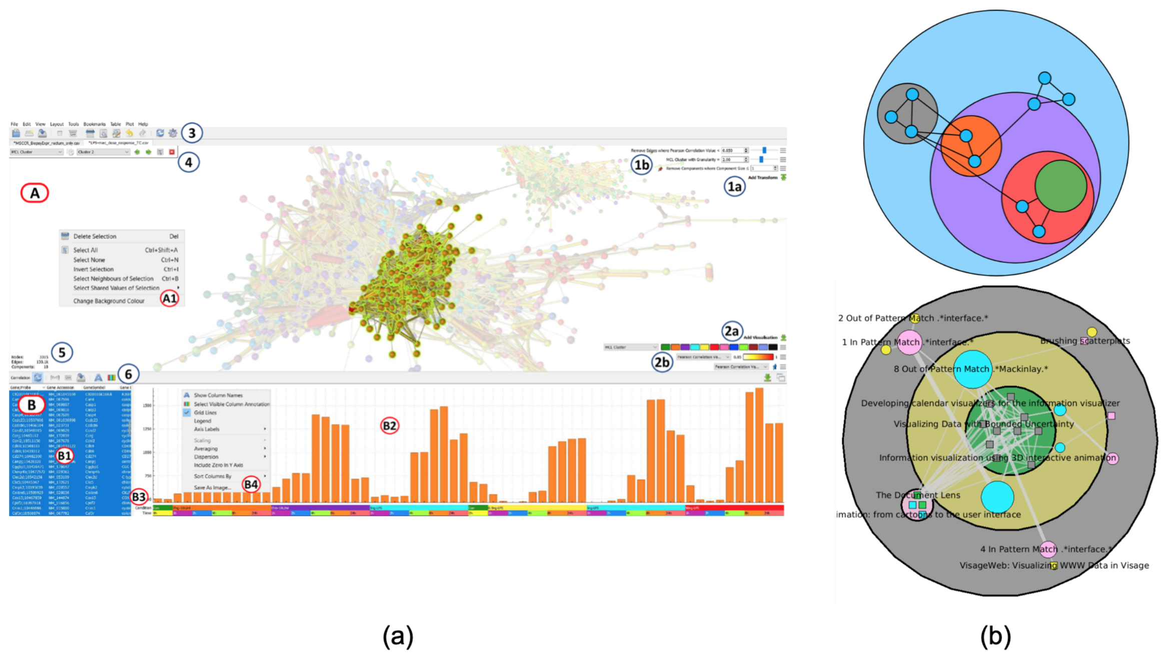


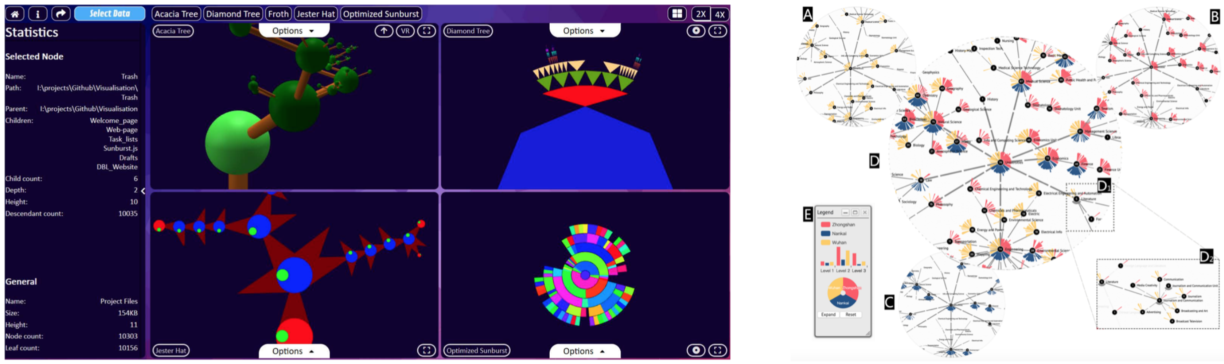
Figure 2.5: (a) Graphia [61] and (b) GrouseFlocks [10]

Furthermore, VizWick [28] was designed to provide visualisations for hierarchical data in a 3D environment. While they emphasised that only one visualisation layout is not enough to visualise all hierarchies as they have different properties such as size, depth, and branching factor, instead of trying to use and **select the best layout according to the data**, they tried to solve this problem by introducing a **multiple views** dashboard (see Figure 2.6.a). Authors suggested that visualising a single dataset simultaneously with different layouts can give more analytical information about it. They included five visualisation methods, circle packing, and sunburst between others, and up to four windows to visualise the dataset. While these layouts are pre-defined, users can select which one to visualise based on their preferences. This approach can be useful as each visualisation can offer information about a different viewpoint of the data. VizWick offers users **view transformations**, such as the ability to drag and manipulate shapes to able to see graphs from different perspectives. Moreover, **interactivity** is beneficial as it allows zooming in, with views synchronised to help **reduce clutter** when examining details. However, when the data is too large in the main view, the **clutter** becomes unavoidable. This tool is **domain-independent**, allows users to upload their data and is easily accessible as it is web-based.

On the other hand, the PansyTree a **pre-defined** visualisation technique addresses a critical challenge in hierarchical data representation by enabling the simultaneous visualisation of multiple hierarchies [50]. Traditional methods typically allow for the display of only a single hierarchical dataset at a time, making it difficult to compare and analyse different datasets. PansyTree introduces a novel approach by **merging up to three hierarchical datasets** into a single visualisation. This is achieved using a unique iconography, where each node in the tree is represented by a "pansy," a flower-like symbol that encodes data from three distinct datasets through different colours and structural elements such as petals and sepals (see Figure 2.6.b). This design not only ensures a clear and readable representation of each dataset's attributes but also helps to display **maximum information without losing context**. In larger datasets, **clutter** is likely to occur due to overlapping nodes and edges, which can hinder the comparison of hierarchical structures. The PansyTree further incorporates a force-directed layout with encoded links, where the width and animations of connections convey additional hierarchical information. Also, PansyTree is designed as **domain-independent**.

2.2.3 Summary

In this section, we have described the criteria that effective hierarchical and network visualisations should possess, such **maximising visual information without losing context**, avoiding **visual clutter**, **selecting the best layout** according to the data, and **domain-dependent vs domain-dependent techniques**. We then conducted a literature review of various visualisation techniques used for network and hierarchical data. The summary of the literature review can be seen in the Table 2.3. This review highlights the importance of choosing the right visualisation method, as each technique offers distinct strengths and limitations. Ensuring that visualisation is well-designed



(a) (b)

Figure 2.6: (a) VizWick [28] (b) PansyTree [50]

to effectively convey complex information without becoming overcrowded or difficult to interpret.

	A) Max info and context	B) Non-cluttered Vis	C) Selecting layout	D) Domain
Functree2 [44]	-	↑ interactivity ↑ multiview ↓ scalability	Pre-defined	Independent
MonaGo [205]	↑ clustering	↑ interactivity ↑ multiview ↓ scalability	Pre-defined	Dependent
Graphia [61]	↑ multiview	↑ interactivity ↑ multiview ↓ scalability	Pre-defined	Independent
GrouseFlocks [10]	-	↑ interactivity ↓ scalability	Pre-defined	Independent
VizWick [28]	↑ multiview	↑ interactivity ↑ multiview ↓ scalability	Pre-defined	Independent
PansyTree [50]	↑ simultaneous hierarchies ↑ iconography	↓ scalability	Pre-defined	Independent

Table 2.3: Summary of the explored hierarchical and network visualisation related works, where ↑ indicates a high level of a feature and ↓ indicates a low level. The green colour represents features that are positive or desirable, while the red colour represents features that are negative or undesirable.

We examined several **visualisation types**, including radial layouts, chord diagrams, and circle packing, each suitable for different **types of data**. For instance, radial layouts, as used in Functree2, can be effective for compact datasets but may suffer from readability issues when additional attributes are visualised through coloured rectangles. Chord diagrams, as demonstrated by MonaGo, are useful for visualising relationships and similarities in genomic data, but scalability can be a concern as data complexity increases which can cause a loss of the context while visualising maximum information. Graphia's network diagram and its ability to perform data transformations via correlation matrices provide a powerful tool for visualising large graphs, though they may become cluttered with very large datasets.

Circle packing, explored in GrouseFlocks is useful for visualising nested hierarchies but can be challenging to interpret with more complex or expansive data structures. VizWick addresses hierarchical data visualisation in a 3D environment with multiple views, allowing simultaneous analysis from different perspectives. Despite its advantages, this method can also face issues with scalability and clarity for very large datasets. The PansyTree technique stands out by integrating multiple hierarchical datasets into a single view, using distinct iconography to enhance data comparability and clarity, and offering a novel approach to visualising complex hierarchical information.

Moreover, it should be noted that most of the related work featured a **multiview** option, a feature we consider essential. Multiview capabilities can effectively mitigate clutter, help, and align closely with Shneiderman's principle of "*details on demand*." Furthermore, **interactive** elements are used in all the related works to help reduce clutter or visualise more information. Therefore, our remark underscores the necessity for integrating multiview and interactive functionalities to enhance clarity and usability in visualisation design.

> **Remark 2.1**. *Both **Multiview** and **Interactivity** are widely used techniques to reduce clutter and maintain the scalability and clarity of the visualisations.*

While all the works aim to reduce clutter, we conclude that the challenge lies in **selecting the best visualisation layout** to convey meaningful insights, which depends heavily on the implicit structures within hierarchical data. These works consistently relied on **pre-defined** visualisations, applying the same approach to all datasets. However, not all hierarchical structures are alike, and scalability remains a significant issue across the board. This highlights the importance of offering diverse visualisation layouts that can adapt to the requirements of different hierarchical structures. Additionally, while these approaches rely on pre-defined visualisations, the ability to automatically select the most suitable layout from multiple options would add significant value, especially for users unfamiliar with choosing the best layout.

> **Remark 2.2**. *Related works focused on **pre-defined** visualisations instead of using **automatic layout selection**, as they often overlooked the flexibility required to accommodate diverse hierarchical structures.*

Finally, while most tools are designed with a specific area in mind, they are domain-independent. The majority claim their tool is compatible with any hierarchical data, which is beneficial as it makes them accessible to a broader range of users.

2.3 Visualisation of Multivariate Data

In the following, we will present the requirements for multivariate data immersed in hierarchical and networked visualisations. We will subsequently examine the current works and conclude with a summary.

2.3.1 Requirements of Multivariate Data in Hierarchical and Networked Visualisations

Multivariate data refers to datasets where each entity (node or edge) is characterised by multiple attributes or features. To interpret this data within the context of a hierarchy or network, it is essential to analyse how these attributes are structured and how they relate to one another. Analysing the relationships between these variables can reveal patterns, such as how certain node attributes influence other node attributes, or how edge attributes affect the relationships between nodes. Creating an understandable hierarchical multivariate visualisation is a difficult job as these kinds of hierarchies are very complex due to the variety of information that can be stored in them [135]. The main challenge arises from the complexity of both the data and the structures, as each node or edge can carry a variety of attributes. The visualisation must effectively communicate both the structure and the underlying data. In the following, we present the requirements to overcome these challenges, following the same categorisation used in the section 2.2.1.

(A) Maximum information preserving the context: The challenge of representing multivariate data within the nodes and edges of a hierarchy or network lies in preserving the structural context and relationships while visually encoding the multiple variables without overwhelming the user. To achieve this, several criteria must be met. These include (1) ensuring that the representation of variables **remains close to the relevant nodes and edges**, either **inside** or **beside** them, (2) maintaining **consistent visual mappings of colour, size, and thickness** for both nodes and edges, (3) ensuring that all variables are **always visible**, and (4) supporting **scalability** to accommodate varying amounts of data.

(B) Non-Cluttered visualisations: Embedding multiple variables into each element of a hierarchy or network can lead to visual clutter, making it difficult for users to interpret the data.

The requirement lies in displaying diverse variables on a single graph without overwhelming the viewer, particularly when information becomes densely packed. For example, if too many attributes are clustered on glyphs, the result may be a loss of clarity. To avoid this, it is important to **reduce overlaps** and **complexity**, with some variables, represented implicitly (e.g., colour for node groups) and others requiring explicit encoding (e.g., edge width for connection strength). Interactivity can improve the visualisation of such data. Features like filtering enable users to selectively display or hide data based on specific attributes or conditions, facilitating focused analysis and reducing the clutter on glyphs. This approach allows for multivariate data to be displayed without overwhelming the user, providing access to details on demand rather than keeping them constantly visible. Moreover, strategic use of colour, shape, and size can effectively distinguish between categories or levels, improving clarity and understanding.

(C) Choosing the best representation of the variables for the data and the VisTask: Selecting the appropriate visualisation approach for multivariate data involves determining the best way to represent the variables without overwhelming the user. This decision might include depicting variables directly on nodes or edges, using icons or glyphs, or employing separate views to improve clarity. Another consideration is whether to visualise these glyphs on the nodes as an all-in-one representation or separately next to the nodes, one-by-one. The challenge lies in determining how to represent the variables effectively—whether through icons, shapes, glyphs, or a combination of these—while ensuring the visualisation remains clear and comprehensible.

It is also important to consider the type of interaction or analysis the visualisation supports. If the aim is to compare multivariate attributes, visualising variables such as colour, size, and shape becomes essential. However, for structural analysis (e.g., identifying subgroups), it might be more effective to summarise or downplay the multivariate data. The requirement is aligning the visual representation of multivariate data with the intended task—whether that's comparison, discovery, or interaction—ensuring that the visualisation facilitates the desired insights without causing confusion.

Finally, we also considered the last requirement D) Prioritising domain-independent over domain-dependent visualisations, which is introduced in Section 2.2.1.

2.3.2 Review of the Literature

Each data point in the hierarchy can be associated with any number and type of attributes. There have been various attempts in the literature to address the challenge of visualising multivariate attributes on hierarchical or network graphs [102]. These approaches often focus on **reducing the**

number of graph elements displayed in the active view, as displaying all the information at once can overwhelm the viewer and obscure clarity and meaning. Common strategies include filtering, interactive techniques, and using hybrid or multiple views to effectively manage and present the data. In the following, we explore the most relevant literature related to this thesis.

The literature proposed different techniques for visualising multivariate data such as interaction-based, hybrid visualisations [99], glyph-based [102], icon-based[98], and animation-based [203]. For example, Bezerianos et al. [18] used the method of reducing the number of graph elements shown on the active view by displaying multivariate data in pairs one at a time (see Figure 2.7.a on the centre the main view and on the left side the combination of each attribute). This approach fulfils the requirement of **non-cluttered visualisations**, as it limits the amount of data displayed at once, reducing the risk of visual clutter and making the information more digestible. They added additional features of interaction and filters to enhance the visualisation such as animation between each graph and graph selection history. However, this method can be limiting, especially in cases where more than two attributes need to be analysed simultaneously. However, it does not fulfil the requirement of **maximum information preserving the context**, as displaying only a few attributes at a time might result in a loss of the full structural context and relationships in the data. Furthermore, while reducing the number of elements displayed may enhance clarity, it can also cause information loss when navigating between different views, potentially disrupting the continuity of analysis. This tool is **domain-independent**.

(a) (b)

Figure 2.7: (a) GraphDice [18] and (b) FacetAtlas [32]

On the other hand, FacetAtlas [32] consists of big bubble-like density maps and relation links (see Figure 2.7.b). They place diseases as nodes in these density maps grouping them according to their similarities and linking diseases that share the same symptoms and treatments with different coloured lines. This method covers the requirement **non-cluttered visualisations**, as it visually groups related data and uses colour to differentiate connections, reducing complexity and aiding in the interpretation of multivariate data. Also, the system allows users to switch to symptom view, highlighting visual patterns, and choosing other links from the interactive facet legend which also helps to reduce clutter. For communicating multivariate data, they relied on dynamic query filtering where users can interact with the visualisation by zooming in to explore only some parts of the data in detail. While FacetAtlas presented its application using biological data, the system itself is **domain-independent**. Similarly, Shareflow [82] used various filters and directly mapped attributes (e.g., colour) onto the main visualisation (e.g., node-link graph) to communicate multivariate data.

Nerex [54] used an icon-based approach. They have 10 icons mapped on an interactive node-link diagram to analyse conversational transcripts, such as multi-party conversations, focusing on entities like persons and geo-locations (see Figure 2.8.b). Additionally, the icons have three different outliners to visualise more attributes such as synonyms (see Figure 2.8.c). This approach fulfils the requirement **non-cluttered visualisations**, as it avoids visual clutter by overlaying the icons directly on top of the nodes. By doing so, the visualisation presents detailed information without overwhelming the user. Moreover, this method reflects principles from **choosing the best representation of variables for the data and the VisTasks**, as the use of icons proves effective for representing the data in a clear and organised manner. Also, the placement of the outliners helps **maximising the information while preserving the context**. Furthermore, Nerex is designed for conversational transcript data, therefore, it is **domain-dependent**.

Moreover, glyph-based visualisation is a popular option. A glyph is a small visual object that represents several attributes simultaneously, often providing a more comprehensive representation of data compared to a single icon [23]. It can be used individually [51, 210, 94] and also in combination with other graphs [206] to add more meaning to the data being presented.

For example, Du et al. [51] designed a donut based glyph as the main visualisation and used other graphs to support the glyph visualisation. This study is designed as a recommendation system to find similarities and dissimilarities between peers such as students. Their main visualisation is called LikeMeDonuts which is supported by the Ranking glyph and History Heat map. LikeMeDonuts has a picture of the person who is the subject at the centre and each attribute is placed as a ring around the picture (see Figure 2.9.a). An example of attributes is gender and a ring is formed with female and male peers with their respective numbers reflected as sizes. Three colours are mapped onto visualisation namely, bright green, dark green and grey. Bright green shows peers that exactly match with the subject, dark green represents peers that are in the tolerance range

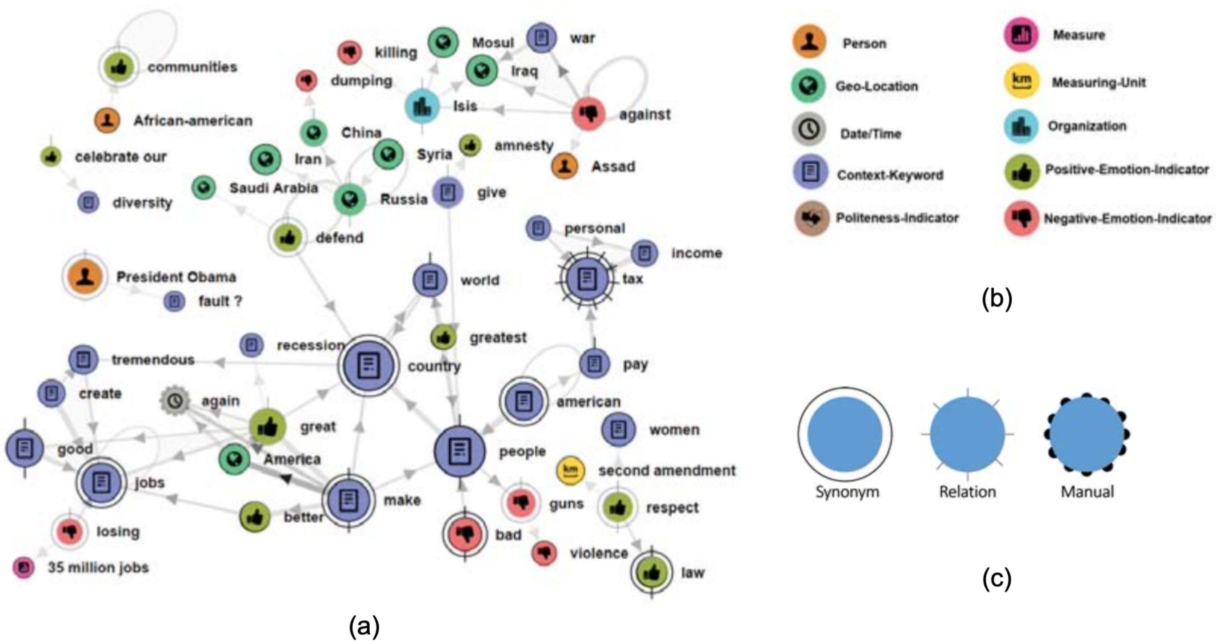


Figure 2.8: Nerex [54]

and grey represents peers that are not in the tolerance range. The tolerance range is the range where subjects share similar records with their peers. For example, if the subject is a male PhD student, he has similarities with female PhD students and male Master students but not with female Master students. This glyph is an interpretation of a hierarchical tree as the number of rings increases the more detailed the LikeMeDonuts gets and the glyph becomes a radial hierarchical tree. The design covers the requirement **non-cluttered visualisations** by segmenting attributes into distinct, easily interpretable rings. The use of three chosen colours enhances clarity, ensuring users can easily distinguish between levels of similarity. Furthermore, supporting graphs, like the Ranking Glyph and History Heat map, distribute information across multiple views, preventing visual overload while enabling deeper analysis. This visualisation is **domain-independent** and can work with analysing any types of peers.

On the other hand, Xu et al. [206] conducted a study where they used glyphs as part of their visualisation. Their study is designed to explore debates that happen on online reviews, such as movie reviews, restaurant reviews, etc. Their glyphs are formed of pie charts and each pie chart represents a topic (see Figure 2.9.b). The pie chart is divided in two showing positive and

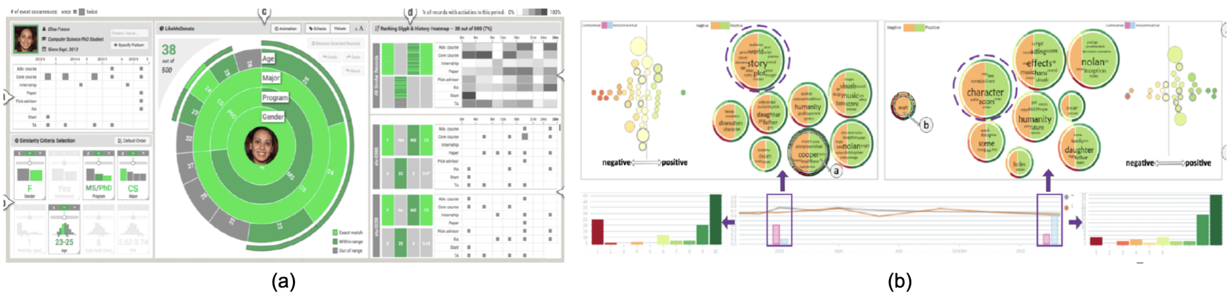


Figure 2.9: (a)LikeMeDonuts [51] and (b)Reviews [206]

negative reviews. The size of each side of the pie represents the number of reviews. Also, the positive side is represented with green and the negative side is represented with orange. There is a word cloud placed on the pie chart showing the most used words. There is an outer ring around the pie chart that shows the ratings of the reviewers on that topic. This ring chart is represented with colours from green to red, green being positive and red being negative. While these glyphs effectively represent sentiment, ratings, and word frequency, they can become crowded and difficult to interpret when applied to a higher number of topics involving larger datasets. This study compromises the requirements **maximum information preserving the context**, and **non-cluttered visualisations**, as overloading the visualisation with too many glyphs can make it harder to analyse each element effectively due to the overwhelming amount of information to process. This visualisation is also **domain-independent** as it is showcased using datasets about movies, products, and restaurants.

Moreover, Social Wave [184] used glyph-based approach in cooperation with their main visualisation to analyse the distribution of popular hashtags (collected from X, previously Twitter) in various locations (see Figure 2.10.a). Their main graph is a network graph but they created three glyphs to communicate more information about hashtags (e.g. proportion of used hashtags) and assigned these glyphs to each node as **all-in-one** presentation, according to their sizes. This is an interesting approach because glyphs added more information to their main visualisation as well as **reduced the clutter** by having different versions according to the size. Social Wave is **domain-independent**, as it is demonstrated using diverse data sets, including information about the sociopolitical events and an epidemic.

ConToVi, [53] was designed to explore speaker behaviours in multiparty conversations. The tool has four main animation-based visualisations (see one of the main views in Figure 2.10.b). In addition to these four views, they included an argumentation glyph to detail the features of each

(a) (b)

Figure 2.10: (a)SocialWave [184] and (b)ConToVi [53]

utterance. Ten argumentation attributes such as assurance, and common ground, among others, were mapped onto an **all-in-one** glyph (similar to a pie chart) to explore the degree of justification and stances of the speakers in utterances (see Figure 2.10.b on the right-hand side the legend and how they are used). The main drawback of this approach is that the glyph is only shown in a separate view, requiring thus a change in the context of users' attention which can require retaining some details of the hierarchical structure and so alter the ability of the user to effectively process the information [90]. This approach compromises the requirement **maximum information preserving the context**, as the use of an **all-in-one** glyph to represent multiple argumentation attributes works well for consolidating data. However, the glyph is displayed in a separate view, requiring users to shift their focus away from the main visualisation. This disruption in context can make it harder for users to process the information effectively, as they must retain some details of the hierarchical structure while interpreting the glyph, potentially hindering their ability to understand the data as a whole. As ConToVi is designed to analyse speaker patterns in multi-party conversations and uses glyphs to specifically analyse conversations, it is **domain-dependent**.

2.3.3 Summary

Creating understandable hierarchical multivariate data presents challenges due to the diverse range of information they can contain. Various methods have been proposed to address these challenges, including filtering techniques, interactive tools, and the use of hybrid or multiple views, glyphs, and icons. We conducted a review of the related works in the previous section and we summarise our results in Table 2.4.

We explored tools that utilised multiple views, such as displaying attributes in pairs, although this approach can sometimes lead to information loss, which challenges **(A) Maximum Information Preserving the Context**. On the other hand, interactions and supplementary charts can add value by allowing attributes to be displayed on demand, with additional information presented in separate views. This approach helps to minimise clutter while still displaying more information, also aligning with **(B) Non-Cluttered Visualisations**.

	A) Max Info and Context	**B) Non-Cluttered Vis**	**C) Choosing Rep.**	**D) Domain**	**Visual Mapping: Layout and Location**
GraphDice [18]	↓ only two attributes at a time	↑ limiting the amount of data displayed at once ↑ interactivity	-	Independent	Scatter plot with links, on x-y axis
FacetAtlas [32]	-	↑ clustering ↑ switch views ↑ interactivity	-	Independent	Node-link diagram, on the nodes as colours
Nerex [54]	↑ outliners	↑ icons on the nodes	icons and outliners	Dependent	Node-link diagram, on the nodes as icons
LikeMeDonuts [51]	-	↑ different rings ↑ multiviews	-	Independent	Glyph, **all-in-one glyph**
Reviews [206]	↑ overcrowded view	↑ overcrowded view	-	Independent	Bubble graph, on the nodes as **all-in-one glyphs**
SocialWave [184]	-	↑ different glyphs	-	Independent	Node-link diagram, on the nodes as **all-in-one glyphs**
ConToVi [53]	↓ glyphs on separate view	↑ glyphs on separate view	-	Dependent	Animation-based visualisation, **all-in-one glyphs** on a separate view

Table 2.4: Summary of the explored related work about multivariate visualisations, where ↑ indicates a high level of a feature and ↓ indicates a low level. The green colour represents features that are positive or desirable, while the red colour represents features that are negative or undesirable.

In the literature, we encountered a study that focused on icons, but extensive use of icons can lead to clutter, especially when representing abstract attributes that are difficult to depict with distinct icons. We reviewed several works that preferred and incorporated glyphs as either integral or primary visual elements. Most of the studies, whether for integral or primary use, showcased glyphs that were designed as **all-in-one** representations. In some cases, glyphs remained constantly present in the visualization, contributing to clutter. Alternatively, glyphs were visualised separately, limiting their integration into more comprehensive visual representations. Additionally, aesthetics play a significant role in visual design; appropriate colours should be chosen to avoid overcrowding the glyphs.

> **Remark 2.3**. Glyphs are the preferred method for visualising multivariate data, and they are often chosen as an **all-in-one**, customised representation, tailored to the specific needs of the dataset or task.

Furthermore, in the case of integral glyphs used in hierarchical or network visualisations, efforts were made to choose locations that avoided clutter while still conveying more information. For example, SocialWave [184] designed three types of glyphs, all presented on the nodes, but some provide more detail, while others are simpler, helping to reduce clutter and improve clarity.

> **Remark 2.4**. *The **location** of the variable representation typically depends on the **visual layout chosen** for the hierarchical or network data.*

2.4 Visualisation of Conversational Data

In this section, we will present data based on conversations, as the following sections will use a case study on this type of data, specifically focusing on hate speech. Similarly to the previous sections, we will also review current research in the field and conclude with a summary.

2.4.1 Conversational Data

Conversational data refers to the collection of information generated through human interactions, often in the form of textual exchanges, multimedia content (such as images, videos, or links), and other elements like emojis or audio [49]. This data is inherently dynamic, characterised by its diverse content and temporality [56]. This diversity not only enriches communication but also presents challenges in processing and analysing heterogeneous data formats. Conversations often unfold linearly, forming a hierarchy where each exchange (or utterance) is part of a broader dialogue or interaction. Each conversation can thus be seen as a hierarchical structure where utterances build upon or respond to one another, with conversations growing in complexity over time.

Moreover, the informal and spontaneous nature of conversations introduces complexity, requiring sophisticated methods to capture and interpret key aspects like sentiment, topics of interest, and user engagement dynamics. Conversational data is inherently multivariate, consisting of different types of variables that can be classified as nominal, ordinal, or categorical. For example, text-based data can include nominal variables such as the speaker's identity, the topic of discussion, or sentiment. These are often encoded in categorical variables, like sentiment (e.g., positive, negative, neutral), topic (e.g., politics, sports, entertainment), or user engagement (e.g., number of responses, likes, shares). Such diverse data types provide valuable insights into both the immediate context and the broader trends within a conversation.

In addition, **temporality** is a key dimension that influences both the structure and interpretation of conversational data. Conversations unfold over time, creating patterns that reflect evolving sentiments, behaviours, and events. For instance, the timing and order of messages shape their meaning and impact, as seen in conversations that transition through stages like introduction, elaboration, and conclusion. Temporal analyses can reveal short-term dynamics, such as immediate shifts in sentiment, while also providing long-term insights into behavioural trends and evolving user engagement.

Furthermore, conversational data can also be analysed through the lens of **layered** networks, each representing distinct dimensions of interaction. For example, a social layer captures stable relationships, such as followers or connections, while a conversational layer maps the transient dynamics of replies, mentions, and shared content. These layers not only reveal the flow of information but also uncover patterns in user behaviour and relationships. Within each layer, hierarchies add further structure and depth. In the conversational layer, threads often form hierarchical trees, where replies branch from an initial post, reflecting the natural organisation of dialogue. Similarly, in the social layer, hierarchies of influence can highlight key users driving engagement and shaping discussions.

Finally, in Sections 2.2.1 and 2.3.1, we introduced key requirements to achieve effective data visualisations: **A) Displaying maximum information while preserving the context**, **B) Avoiding cluttered visualisations**, and **C) Selecting the best visualisation layout/representation for the data and the associated VisTask**. These requirements are highly relevant to the visualisation of conversational data. **A)** helps ensure that the flow and context of conversations are preserved, which is necessary for understanding patterns such as how topics evolve or sentiment shifts over time. **B)** is essential for managing the complexity of large-scale conversational data, ensuring that visualisations remain clear and interpretable despite the volume and variety of the data. **C)** supports the selection of appropriate representations to balance the multivariate nature of conversational data which has a hierarchical structure, enabling the effective exploration of variables like sentiment, topic, and interaction dynamics without overwhelming the user.

2.4.2 Review of the Literature

The literature provides several works that focus on visualisation of conversations [212, 31, 150, 144, 79, 82, 126, 135, 199, 7, 63]. Some works have focused on visualising trends and values of non-hierarchical variables by exploring various techniques such as colours, icons, and glyphs, while others have proposed hierarchical visualisations to effectively represent multivariate data within structured conversations.

FluxFlow [212] is an interactive data visualisation system designed to display and evaluate anomalous information spread (rumours and misinformation) on Twitter. The novel visualisation design consists of packed circles (retweets) arranged along a timeline showing how an original message spreads among people over time (see Figure 2.11). The size of the circles symbolises the power of the influence of a user, and the colour represents an anomaly score. In terms of visualisation requirements, it adheres to **avoiding clutter** by using size and colour to distinguish between the volume and significance of retweets. However, dense clusters can still reduce clarity. It also supports **scalability** by allowing the layout to adapt to varying data volumes, although handling larger datasets may pose challenges.

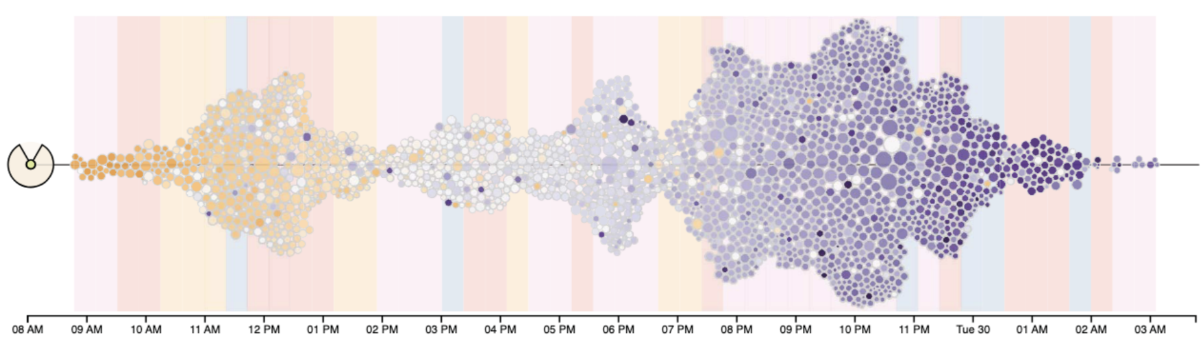


Figure 2.11: FluxFlow [212]

Episogram [31] was designed to analyse retweeting behaviours on Twitter (see Figure 2.12). It visualises the activity of each person separately, displaying every message as a single line on a timeline. While this approach focuses on **preserving context** by representing individual interactions over time, it becomes overwhelming due to **clutter**, as the abundance of lines makes it challenging to interpret the data at a glance. Furthermore, while the visualisation design is simple, it lacks **scalability**, as the timeline can become overcrowded with larger datasets, hindering clear comprehension of the retweeting behaviour patterns.

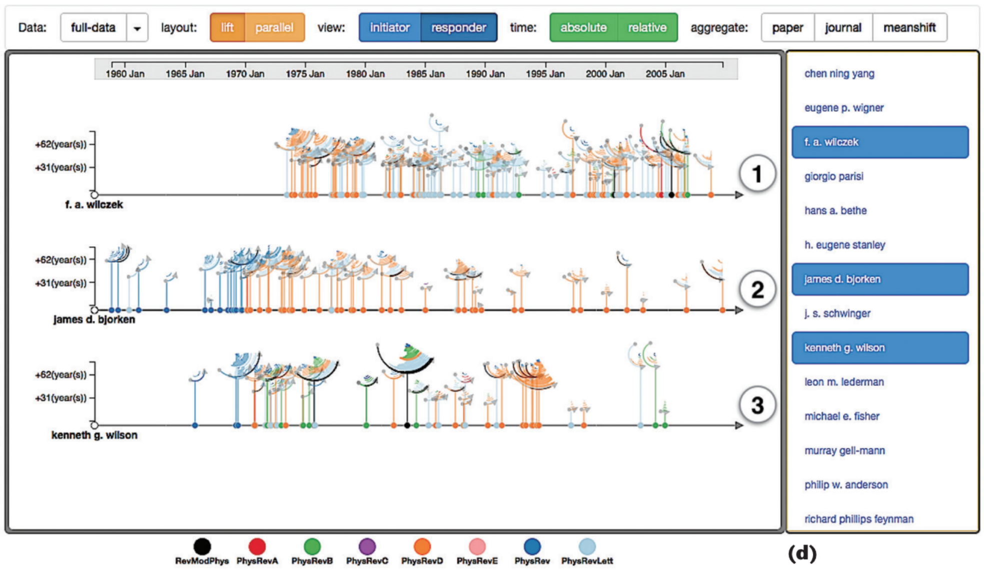


Figure 2.12: Episogram [31]

RumorLens [199] offers a comprehensive tool for detecting and analysing the diffusion of rumours on social media (see Figure 2.13). It **preserves context** by providing a Location Distribution View that maps geographical spread and a Topic Evolution View that tracks the rise and fall of themes over time, ensuring a clear representation of rumour dynamics. The tool **avoids clutter** by using separate, well-organised views that focus on specific aspects, such as sentiment, influence, and post content, ensuring clarity. Finally, it **selects appropriate visualisation layouts** to represent different data types, like the Propagation View's circular diagram for understanding rumour dissemination, and the Features Projection View's clustering technique, which allows for easy comparison of rumour characteristics.

Mandola [144] was designed with NLP and ML techniques to monitor and detect online hate speech on social media (see Figure 2.14). The visualisation includes several interactive tools to present hate speech data effectively. The Hate-map (Figure 2.14.a) uses heat spots on a world map to indicate the intensity of hate speech in specific regions, with brighter and larger spots highlighting areas of higher activity. A temporal slider allows users to analyse trends and spikes over time. The Heat map visualisation (Figure 2.14.b) organises hate speech data by thematic categories, such as ethnicity, politics, religion, and social issues, using a colour-coded matrix to display the density of incidents over time. The Hotspot Map (Figure 2.14.c) uses a colour scale

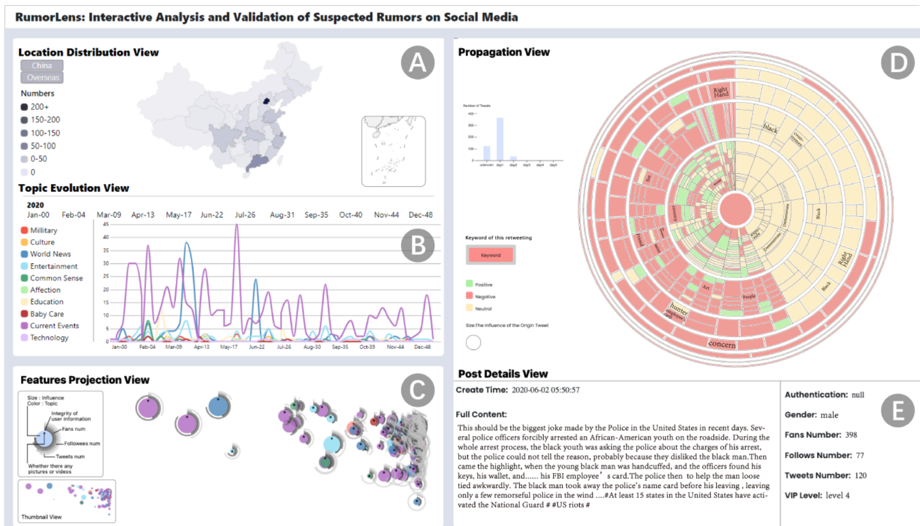
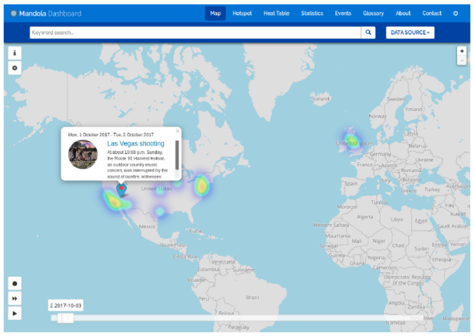


Figure 1: *RumorLens*, a multi-level visual analytics system to help users analyze and validate suspected rumors on social media in an interactive way. A) Location Distribution View provides a summary of suspected rumors' spatial distribution; B) Topic Evolution View shows changes of suspected rumors with different topics over time; C) Features Projection View reveals overall characteristics of suspected rumors and similarity with each other; D) Propagation View visualizes the dynamic spreading details of a suspected rumor with a novel circular visualization design; E) Post Details View displays the details of user information and post contents.

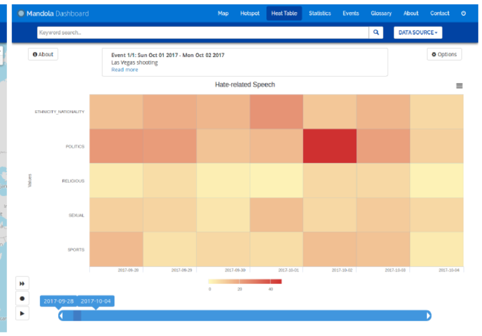
Figure 2.13: RumorLens [199]

to represent six levels of hate speech intensity across countries, providing a global overview of hate-related activity. Finally, the all-purpose statistical visualisation (Figure 2.14.d) combines multiple data elements, including a temporal graph, a U.S. map with regional details, and bar charts showing hate speech distribution across cities and themes. The system design follows the key visualisation principles by preserving context through the integration of geographical and temporal data. It **avoids clutter** by using multiple views, each focusing on different attributes of the data, allowing users to explore the information without overwhelming the screen. The use of distinct colour gradients further enhances clarity, and the design maintains **scalability** by grouping hate speech data by regions and topics, making it adaptable to larger datasets.

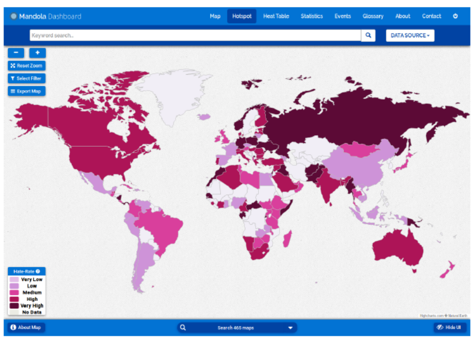
Moreover, some works utilise **hierarchical visualisations** for analysing **conversations**; here, we will explore those that are most relevant. ConVis [79] was designed to analyse comments in online conversation threads and focused on getting the perception of the whole conversation at first glance. They display each comment as a horizontal stack bar and all the replies are stack bars placed under each other by their order in the thread (see Figure 2.15, comments are stacked together). The levels of the threads are shown by positioning bars with indentation. However, this approach has **limitations in preserving context** and **avoiding clutter** when dealing with deeper



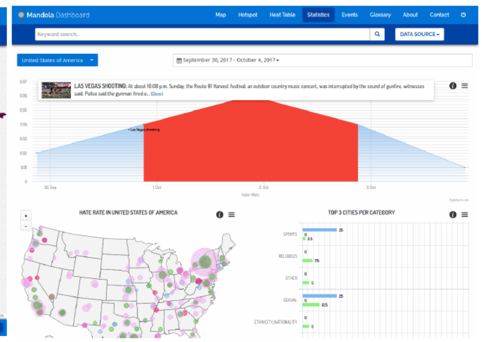
(a) Hate-map visualization



(b) Heat-map visualization



(c) Hotspot map visualization



(d) All-purpose statistical visualizations

Figure 2.14: Mandola [144]

or broader hierarchies. The architecture may struggle to visualise extensive hierarchies effectively, as such data may not fit on a single screen, failing the criteria for **scalability**. Additionally, the use of indentation to represent thread levels can result in stretched bars for long and narrow threads, leaving large empty spaces that disrupt the balance and clarity of the visualisation, contrary to principles of even node distribution and space preservation. Moreover, this layout incorporates multivariate data using a **one-by-one** style glyph representation on the bars. This approach is effective in this case because the bars provide sufficient space to accommodate and display the glyphs.

Other works used different layouts, such as ShareFlow [82] used a radial tree layout to show information diffusion between individuals on social media fan pages' comment sections (see Figure 2.16). Nevertheless, their method did not show the data hierarchically and rather showed hierarchical items side by side. This design choice **limits the ability to preserve context**, as the side-by-side arrangement may obscure the natural parent-child relationships inherent to hierarchical data. Furthermore, the layout risks creating **visual clutter** when dealing with complex hierarchies,

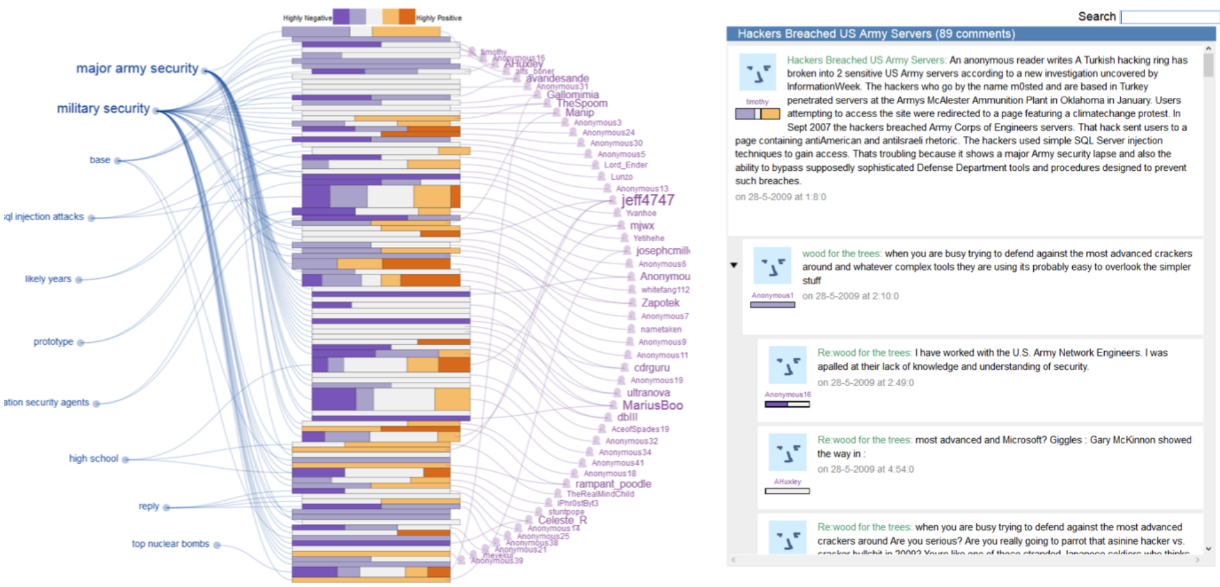


Figure 2.15: ConVis [79]

as it does not fully exploit the available canvas space to distribute nodes evenly or maintain proximity between connected elements. These shortcomings highlight the need for scalable and context-preserving designs when visualising hierarchies within conversational data.

Additionally, Forum Explorer [126] was designed to visualise threaded conversations on websites like Reddit (see Figure 2.17). They used a radial layout to visualise the threads as a whole, i.e., achieve a complete view of all the threads, and they used a traditional tree layout in addition to the radial layout to visualise some of the large sub-conversations in the hierarchies separately. This approach aligns with the requirement of **preserving context** by making both the overall structure and detailed sub-threads accessible. However, radial layouts may struggle with **scalability** when dealing with highly large hierarchies. If the data set is large or densely connected, the visualisation can become cluttered, as **overlapping nodes and edges** obscure relationships. While the technique is useful for analysing larger sub-threads, it may not provide the most informative or adaptable representation for all hierarchical structures, especially when prioritising clarity and avoiding visual clutter.

2.4.3 Summary

The literature presents various approaches to visualising conversations and information diffusion on social media, each addressing unique aspects of hierarchical and multivariate data representation.

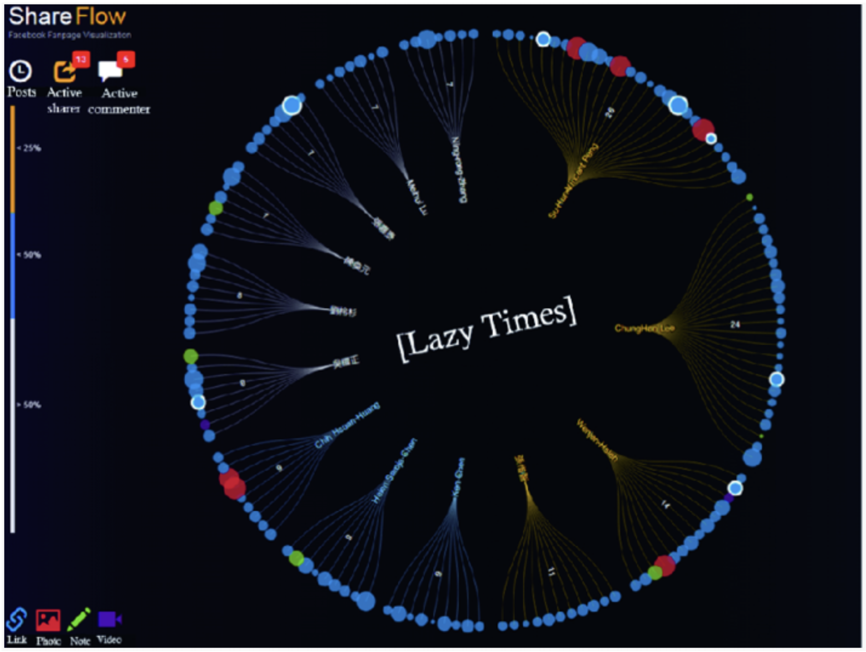


Figure 2.16: ShareFlow [82]

We summarise our findings in the Table 2.5.

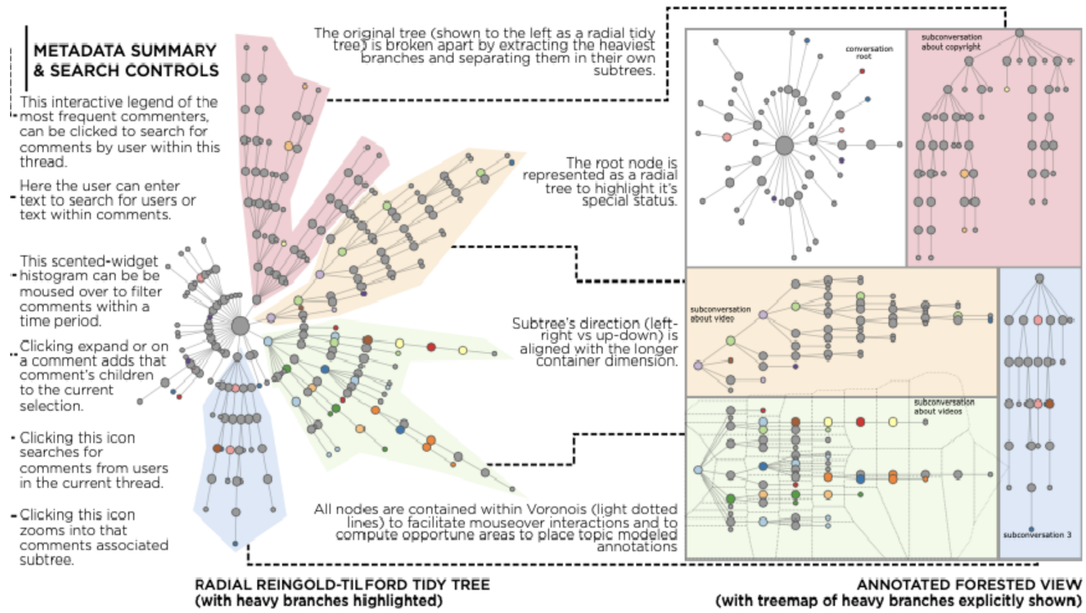


Figure 2.17: Forum Explorer [126]

Vis Name	**A) Max Info and Context**	**B) Non-Cluttered Vis**	**C) Choosing Rep.**	**Visual Mapping: Layout and Conv. Data**
FluxFlow [212]	-	↑ size ↑ colour ↓ scalability	-	Bubble graph, colours
Episogram [31]	↑ visualising messages separately	↓ visualising messages separately ↓ scalability	-	Lines, lines and colours
RumorLens [199]	↑ multiviews	↑ multiviews	Chooses suitable layouts for various data types	Sunburst diagram, colours and shapes
Mandola [144]	↑ multiviews	↑ multiviews	-	Map and heat map, colours
ConVis [79]	↓ scalability	↓ scalability	-	Bar blocks on top of each other, colours
ShareFlow [82]	↓ scalability	↓ scalability	-	Radial, colours
ForumExplorer [126]	↑ subtree views	↓ scalability	-	Radial and tree, colours

Table 2.5: Summary of the literature review of conversational data visualisations, where ↑ indicates a high level of a feature and ↓ indicates a low level. The green colour represents features that are positive or desirable, while the red colour represents features that are negative or undesirable.

For instance, FluxFlow and RumorLens employ packed circles and circular diagrams to depict the spread of rumours and anomalies on X (previously Twitter), complying with the principle of preserving context (A) by illustrating the flow of information explicitly. However, these methods may struggle with avoiding clutter (B), as densely packed circles or flow paths can obscure relationships when not well-distributed.

Episogram and ConVis focus on retweeting behaviours and conversation threads, using timelines and stack bar visuals. These layouts effectively preserve the temporal aspect of conversations while remaining scalable for smaller datasets. However, the stack bar approach in ConVis can lead to inefficient use of space and visual clutter when dealing with long or narrow threads.

Mandola leverages maps and heat maps to monitor hate speech, offering a clear spatial context that aligns with task-specific visualisation needs (C). Yet, its utility for hierarchical data is limited, as heat maps do not inherently convey parent-child relationships within the data. Conversely, ShareFlow and Forum Explorer use radial layouts for information diffusion and threaded conversations, which can effectively depict hierarchical structures when appropriately applied. However, radial layouts often fail to avoid overlapping nodes and edges, reducing clarity in large datasets.

Overall, these works demonstrate how criteria such as preserving context, avoiding clutter, and selecting appropriate layouts influence the effectiveness of visualisations for hierarchical and multivariate data of conversations.

> **Remark 2.5**. *Most commonly, conversational data is mapped to* ***colours*** *and visualised using* ***radial layout****, visualisations can become cluttered when displaying large datasets.*

Moving forward, the next chapters will focus on optimising these visualisation approaches by integrating the strengths of hierarchical layouts to better represent complex, multivariate data with greater clarity. The goal will be to address the gaps identified in existing works, particularly in effectively visualising deeper hierarchies and integrating multivariate data in a manner that remains clear and uncluttered.

Chapter 3

Categorisation of Hierarchical Multivariate Data

Visualising hierarchical data, particularly when it is multivariate, is a challenging task. Multivariate hierarchies are complex, large, and contain a variety of information and properties. The main challenges involve representing this information without introducing visual clutter or losing contextual meaning. Numerous well-known visualisation methods exist to display hierarchies, such as Treemaps, Tree diagrams, and Sunburst charts [103]. In this context, choosing an appropriate visual mapping is a difficult task and it becomes even more challenging for users who are unfamiliar with visualisations. This necessitates the use of automatic visual mapping of layouts to facilitate the creation of effective visual representations as it can be observed in the Remark 2.2 in the previous Chapter.

Nonetheless, the overarching challenge, as we mentioned in Challenge 2 in the Introduction 1, lies in finding the visualisation method that best fits a specific hierarchical structure. Hierarchical structures can vary greatly in shape, which is influenced by factors such as the degree of connectivity of internal nodes and the number of nodes. The term ”shape” refers to the varying distributions of nodes and their connections, which can differ in levels and fan-outs, resulting in structures that may be broad or narrow and deep.

We propose that to create effective visualisations, it is important to select the right visualisation layout while considering these different inner shapes. Moreover, when hierarchical data is also multivariate, additional methods are required to effectively represent these attributes or variables embedded in the hierarchy. This can be accomplished by incorporating visual variables such as colour, position, and shape, as well as by utilising icons and glyphs, depending on the type of variable or attribute being represented and the layout in which it will be embedded. Additionally, when multiple attributes need to be displayed simultaneously, this influences the choice of visual elements to ensure clarity and minimise cognitive load.

Thus, in this chapter, we first introduce the notation of hierarchical data and variables. Afterwards, we propose a categorisation of hierarchical data based on its internal shape and present a formalisation algorithm. Additionally, we analyse this algorithm with examples, and we propose an automatic layout selection based on this categorisation. Then, we provide guidelines for visual mapping of multivariate data and conclude with a discussion.

3.1 Networked and Hierarchical Data Categorisation

Hierarchical data contains nodes and edges that connect a pair of nodes. Depending on the number of nodes and their connexions, the "shape" of the internal structure of the hierarchy can vary. In this section, we propose the formalisation of two prominent shapes that often emerge, which are the elongated shape and the compact shape. The **elongated** shape is characterised by a hierarchy that extends linearly, meaning they are narrow (see Figure 3.1.a). In contrast, the **compact** shape features a more condensed structure, with nodes densely packed together, often concentrated in the upper levels of the hierarchy and branching out from a central point (see Figure 3.1.b). Additionally, some hierarchical structures can exhibit mixed topologies, combining elements of both elongated and compact shapes (see Figure 3.1.c). In the following, we will define specific properties, which will later be used to accurately characterise these different shapes.

3.1.1 Notation of Networked and Hierarchical Data

In this section, we introduce the notation of networked data and hierarchical data, along with some properties that will be used in the next chapters of this manuscript. **Networked** data is defined by a system of objects (nodes) and links (edges) representing many-to-many relationships between them. In this type of data, entities can have multiple, reciprocal connections, making it suitable for analysing systems with complex, interdependent components, such as social networks or biological networks.

A **networked dataset** can be formalised as a graph $G = (Nodes, Edges)$, where:

- $Nodes = \{n_0, n_1, \ldots, n_i\}$ is the set of nodes (vertices), representing entities in the network.

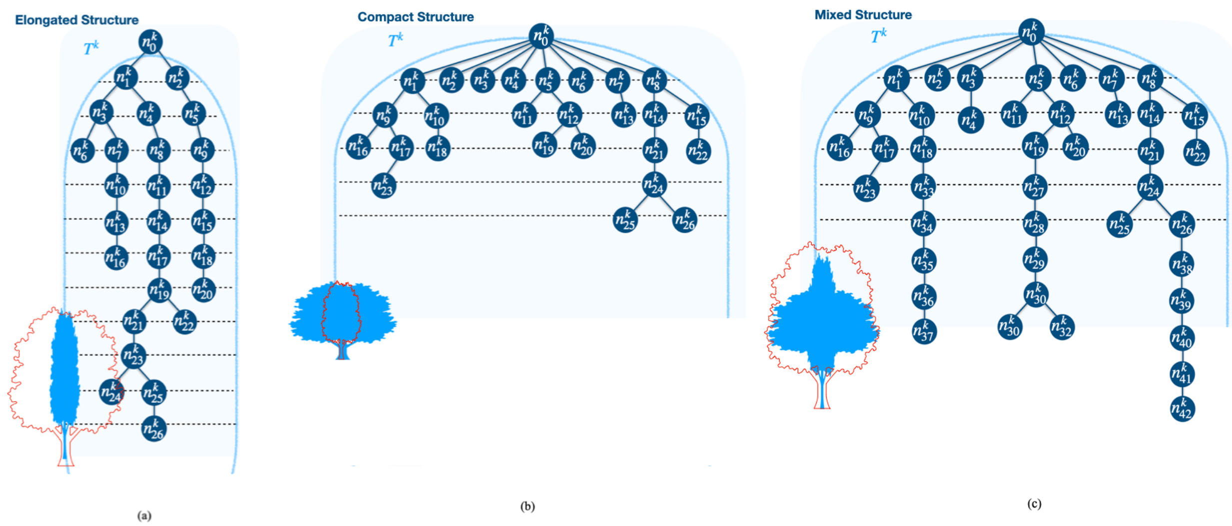


Figure 3.1: Examples of (a) Elongated, (b) Compact, and (c) Mixed structures.

- $Edges = \{e_{i,j} \mid n_i, n_j \in Nodes\}$ is the set of edges (or links), where each edge $e_{i,j}$ represents a relationship between nodes n_i and n_j.

Each edge $e_{i,j} \in Edges$ can be either directed or undirected, depending on the nature of the relationship between n_i and n_j.

When we deal with real data, we can obtain several connected components in a networked dataset. We define a **connected component** of a graph $G = (Nodes, Edges)$ is a subgraph $G^k = (Nodes^k, Edges^k)$ where:

- For every pair of nodes $n_i^k, n`k_j \in Nodes^k$, there exists a path of edges in $Edges^k$ connecting them.
- The subgraph is maximally connected, meaning that no node from $Nodes \setminus Nodes^k$ can be added to $Nodes^k$ while preserving this property of connectivity.

In contrast, **hierarchical** data is organized in a tree-like structure where entities are arranged in levels, with each entity (node) having a clear parent-child relationship. Connections in hierarchical data follow a strict one-to-many pattern, ensuring a top-down organization where each lower-level entity is subordinate to a higher-level one. In the following, we formalise **hierarchical structures**, introducing some basic definitions about them.

We define a **set of hierarchical structures**,

$$\mathcal{T} = \{T^1, T^2, T^3, ..., T^n\}. \tag{3.1}$$

where each T^k, the k-th **hierarchy**, is a directed rooted tree,

$$T^k = < Nodes^k, Edges^k >, \tag{3.2}$$

being,

$Nodes^k = \{n_0^k\} \cup N^k$ is the set of nodes of the hierarchy , being n_0^k the root node of T^k, and $N^k = \bigcup\{n_j^k\}, \forall 1 \leq j \leq num^k$, the set of all the other nodes of the tree T^k, and

$Edges^k = \bigcup\{e_{i,j}^k\}$, is the set of edges of the hierarchy, where $e_{i,j}^k$ is the directed edge from n_i^k to n_j^k, if n_j^k is related to n_i^k, and n_i^k, $n_j^k \in Nodes^k$.

Notice that the **size** of T^k is the number of nodes of the tree, $\#Nodes^k = num^k + 1$, where num^k is the number of nodes directly or indirectly related to the root node, n_0^k. Moreover, all the hierarchies T^k are **weakly connected** and **acyclic** graphs. Each T^k is weakly connected since when we change all of its directed edges for non-directed edges, we get a connected non-directed tree, where there is one and only one path from any node to any other node in T^k. In addition, T^k, as a **single rooted** graph, is acyclic, i.e., no node has more than one parent; thus, it presents no cycles. We define a subtree of T^k rooted at n_j^k as $Subtree(T^k, n_j^k)$.

Furthermore, there are some relevant properties of the hierarchy to bear in mind such as depth and width. First, let's consider the **depth** of any node of the tree T^k as the distance of the node n_j^k to the root n_0^k, taking as distance between two nodes (n_i^k, n_j^k), the number of connected edges from the node n_i^k to the node n_j^k,

$$depth(T^k, n_j^k) = distance(n_0^k, n_j^k), \forall 0 \leq j \leq num^k$$

Note that we can define similarly the depth of any node, n_i^k, of a subtree rooted in n_j^k:

$$depth(Subtree(T^k, n_j^k), n_i^k) = distance(n_j^k, n_i^k),$$
$$\forall n_i^k \in Subtree(T^k, n_j^k)$$

Then, we define $\mathcal{D}(T^k) = \max_{n_j^k \in N^k} depth(T^k, n_j^k)$ as the depth of the tree. Likewise, the depth of a subtree is $\mathcal{D}(Subtree(T^k, n_j^k))$.

Secondly, we define $directChildren(n_i^k)$ as the set of nodes n_j^k belonging to T^k such that $distance(n_i^k, n_j^k) = 1$, and then we define $witdh(n_i^k)$ of a node as the number of its direct children,

$$width(n_j^k) = \#directChildren(n_j^k) \ \ \forall 0 \leq j \leq num^k$$

Thus, we define $\mathcal{W}(T^k) = \max_{n_j^k \in N^k} width(n_j^k)$ as the **width** of the tree.

Based on the characteristics, now we will define how a tree grows. First of all, we define a node as **significant** when it has enough descendants in relation to its parent's descendants. That is, significant nodes fulfil the following conditions:

$$\frac{size(Subtree(T^k, n_j^k))}{size(Subtree(T^k, parent(n_j^k)))} > tolerance \tag{3.3}$$

We define $significant(n_i^k)$ as the set of significant direct children of the node n_i^k.

Then, the **Growing Factor** of a tree rooted in n_i^k, either for the whole tree (T^k rooted in n_0^k) or any subtree ($Subtree(T^k, n_j^k)$ rooted in n_j^k), refers to how n_i^k branch out, i.e., it defines the relationship between its width and the width of any of its s-th subsequent levels. Within a level, we only consider those significant nodes, n_j^k. Thus, the Growing Factor of the subtree rooted in the node n_i^k of a s-th sublevel defined as:

$$GrowingFactor(n_i^k, s) = \frac{\displaystyle\sum_{n_j^k \in s\text{-th } level(n_i^k)} width(n_j^k)}{width(n_i^k)}$$

being s-th $level(n_i^k) = \{n_t^k\}$, where each node n_t^k is located at the s level of the subtree rooted at n_i^k, i.e., $depth(Subtree(T^k, n_i^k), n_t^k)$ equals to s.

Given any hierarchical structure - a whole tree or a subtree -, we define the **tendency** of its inner shape based on the Growing Factor. This trend tells us how the tree grows through its sublevels: elongated or compacted (see Equations (3.4) and (3.5), respectively). These conditions depend on certain threshold values, N, L, $GF_{Elongated}$, and $GF_{Compact}$ related to the number of direct children, a certain number of levels and the growing factor thresholds related to elongated and compact tendencies, respectively. Next, we formalise the **Elongated and Compact tendencies** as a combination of two conditions AND ($\wedge$), and OR ($\vee$):

- **Elongated Tendency**, $ET(n_i^k, L, GF_{elongated})$ means that the tree rooted at n_i^k is narrow along l levels.

$$\begin{aligned}
&width(n_i^k) \leq N && \wedge \\
&\Big(\#significant(n_i^k) = 0 && \vee \\
&\quad GrowingFactor(n_i^k, s) \leq GF_{elongated}, && \\
& && \forall s : 1 \leq s \leq L\Big)
\end{aligned} \tag{3.4}$$

- **Compact Tendency** $CT(n_i^k, L, GF_{compact})$ means that the tree rooted at n_i^k is broad along L levels.

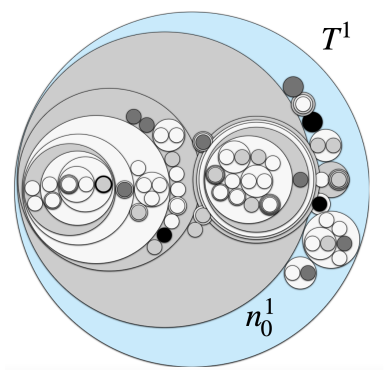
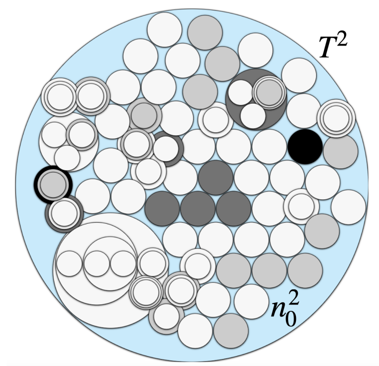
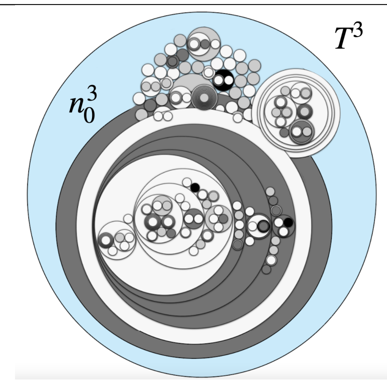
$$\begin{aligned}
&width(n_i^k) > N && \wedge \\
&\Big(\#significant(n_i^k) = 0 && \vee \\
&\quad GrowingFactor(n_i^k, s) \geq GF_{compact}, && \\
& && \forall s : 1 \leq s \leq L,\ GrowingFactor(n_i^k, s) \neq 0\Big)
\end{aligned} \tag{3.5}$$

The first condition to distinguish both tendencies checks the number of direct children of the root node, $width(n_i^k)$. This value fixes the tendency of the tree, which is elongated whenever this value is below the threshold value, N, and compact otherwise.

Moreover, the second condition checks the significance of the node n_i^k, i.e., if it has significant children and its growing factor. It should be noted that when a node has no direct significant children, its Growing Factor makes no sense, i.e., $\#significant(n_i^k) = 0$, Table 3.1.(b) is an example, node n_0^k in blue colour, that has no significant children. Thus, in those cases, the tendency is only based on the width of the node because $\#significant(n_i^k) = 0$ is true (see the first part of the second condition in Equations 3.2 and 3.3).

The value L in the second condition models the proportion of the tree needed to determine its shape's tendency. For example, by setting L to 1, we are only considering the trend of the shape at the first level but not in the rest of levels, and by setting L at the maximum depth of the tree, we are demanding all its levels to strictly follow that tendency. Note that this constraint is

Table 3.1: Circle Packing with categories (a) Elongated, (b) Compact, and (c) Unspecified. The threshold values used to compute significant nodes and Growing Factors are: $N = 25$, $L = 4$, $GF_{Elongated} = 1.9$, $GF_{Compact} = 0.25$, and $tolerance = 0.15$.
Note that data sets are automatically classified by means of an algorithm that implements the categorisation presented in section 3.1.

		
(a) Data set categorised as Elongated	(b) Data set categorised as Compact	(c) Unspecified data set
$\mathcal{W}(T^1) = 18$ $\mathcal{D}(T^1) = 11$ $\#Nodes^1 = 100$	$\mathcal{W}(T^2) = 60$ $\mathcal{D}(T^2) = 4$ $\#Nodes^2 = 100$	$\mathcal{W}(T^3) = 45$ $\mathcal{D}(T^3) = 15$ $\#Nodes^3 = 200$
$width(n_0^1) = 9$ $\#significant(n_0^1) = 2$ $GrowingFactor(n_0^1, 1) = 1.1,$ $GrowingFactor(n_0^1, 2) = 1.5,$ $GrowingFactor(n_0^1, 3) = 0.8,$ $GrowingFactor(n_0^1, 4) = 0.7$	$width(n_0^2) = 60$ $\#significant(n_0^2) = 0$	$width(n_0^3) = 45$ $\#significant(n_0^3) = 3$ $GrowingFactor(n_0^3, 1) = 0.08,$ $GrowingFactor(n_0^3, 2) = 0.11,$ $GrowingFactor(n_0^3, 3) = 0.26,$ $GrowingFactor(n_0^3, 4) = 0.33$

theoretically possible but it is not easy to happen in real data sets. Actually, an intermediate value of L fixes the trend in the first L levels of the tree, without considering how the rest of the levels behave. Table 3.1 shows the $GrowingFactor(n_0^k, s)$, with s between 1 and 4 ($L = 4$) in columns a and c.

In addition, the value of the growing factor, GF, tell us about the tree's growth, i.e., it determines how the L sublevels of a node maintain in relation to how it does. For example, if the same tendency is maintained in the successive L sublevels of the tree, then the $growingFactor$ is approximately 1, which strictly guarantees the tendency of the whole tree. In the case of elongated tendency, to control how the tree expands we define a maximum value, $GF_{Elongated}$ below which we

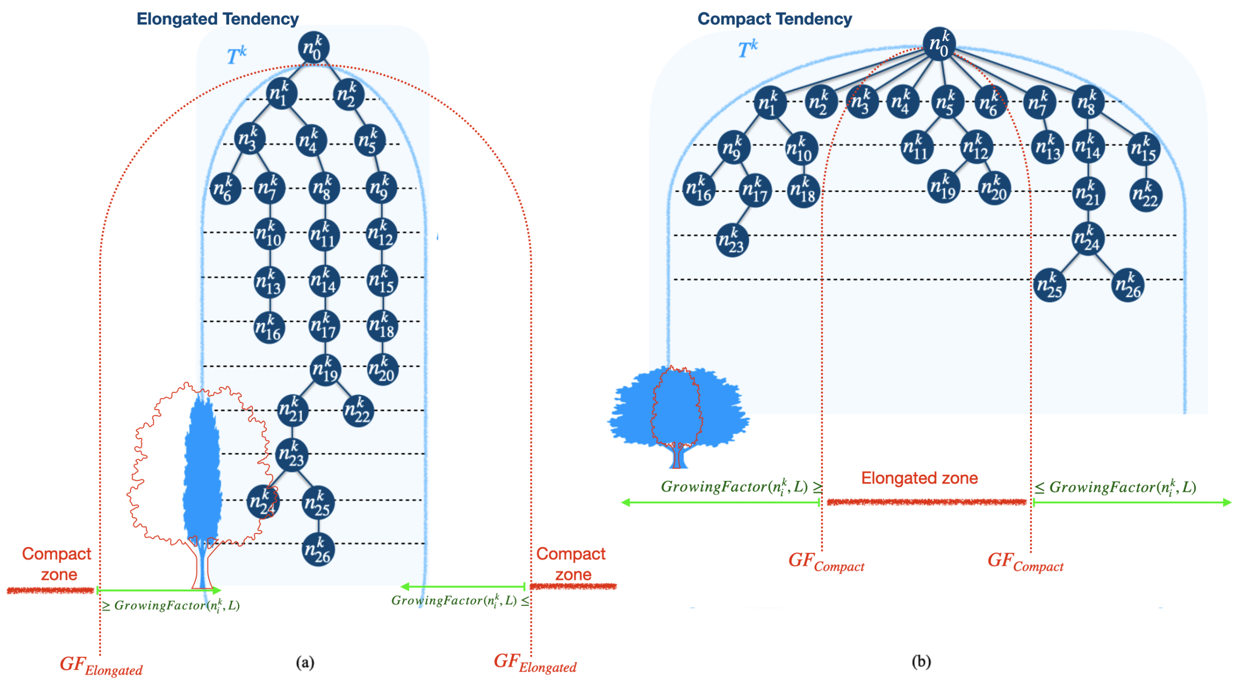


Figure 3.2: Overview of (a) Elongated and (b) Compact tendencies. In red, are zones where the tendency changes from elongated to compact, and from compact to elongated.

consider that the tendency is maintained (see Figure 3.2.a). Similarly, we define a minimum value, $GF_{Compact}$ up to which we consider that the tendency is maintained in the case of compact tendency (see Figure 3.2.b). For instance, Table 3.1.(a) shows a root node with 9 direct children (less than $N = 25$) and the following s levels up to $L = 4$ with a $(GrowingFactor(n_i^k, s) \leq GF_{elongated}(= 1.9))$. Thus, according to Equation (3.2), we can affirm that the root node, n_0^k follows an Elongated Tendency (ET).

Additionally, we can detect some parts of the hierarchies that become linear sequences, i.e., such **spines**. Then we can easily identify that a spine starts at node n_i^k when the relation between the number of nodes of the subtree of n_i^k, and the number of sublevels of that subtree is close to 1. Let denote $spine(n_i^k)$ if $\frac{\mathcal{D}(Subtree(T^k, n_i^k))}{\#Subtree(T^k, n_i^k)} = 1$.

Considering elongated and compact tendencies, as depicted in Figure 3.2, we propose the following **categorisation of hierarchies**:

1. T^k is **Elongated** when all the significant nodes through L levels starting at the root node of T^k fulfil the elongated tendency property, $ET(n_0^k, L)$

(see Figure 3.3). Notice also that spines are a particular case of elongated structures.

2. T^k is **Compact** when the main trend of the hierarchy fulfils compact properties at the root of T^k or any node/s at some distance to it, D, and also it has not narrow (elongated) subtrees (see Figure 3.4) .

$$(CT(n_0^k, L, GF_{compact}) \vee$$
$$\exists n_i^k : CT(n_i^k, L, 1.0), distance(n_i^k, n_0^k) < D)$$
$$\wedge \ \nexists n_i^k : ET(n_i^k, L) \ \forall 0 \leq i \leq \#Nodes^k$$

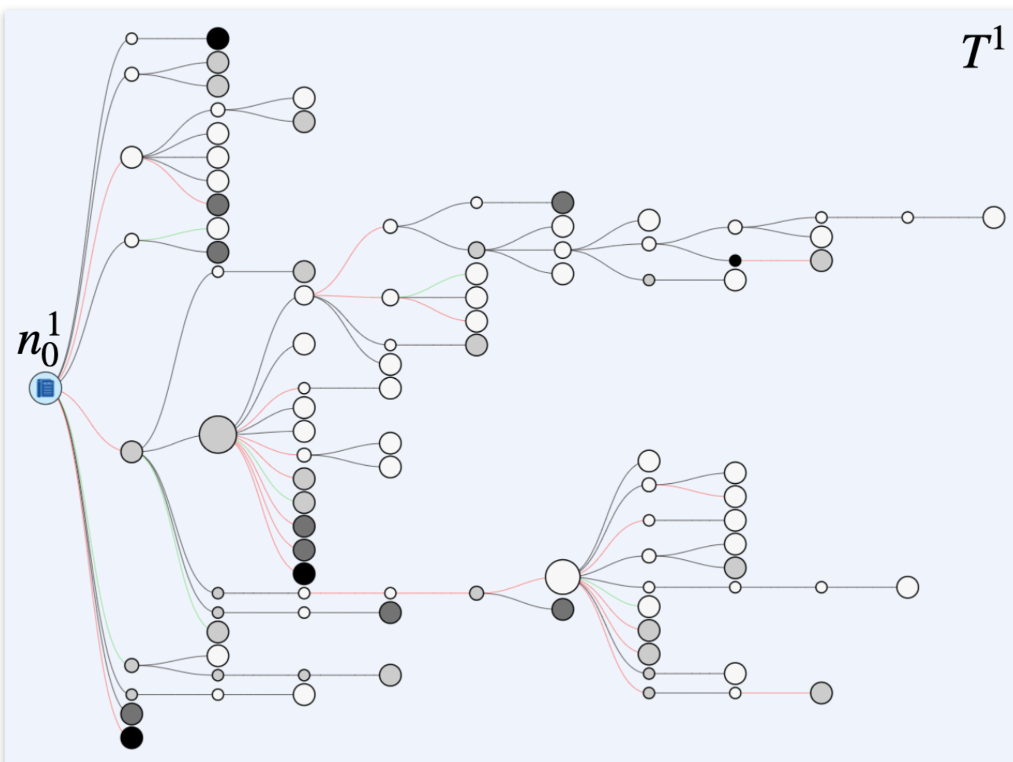


Figure 3.3: Example of Elongated hierarchy: $width(n_0^1) = 9, \#significant(n_0^1) = 1$, $GrowingFactor(n_0^1, 1) = 0.55, GrowingFactor(n_0^1, 2) = 1.33, GrowingFactor(n_0^1, 3) = 0.55$, $GrowingFactor(n_0^1, 4) = 0.66$, Threshold values: $N = 25$, $L = 4$, $GF_{Elongated} = 2$, $tolerance = 0.15$.

It should be noted that not all hierarchies fall into these two categories and thus may remain

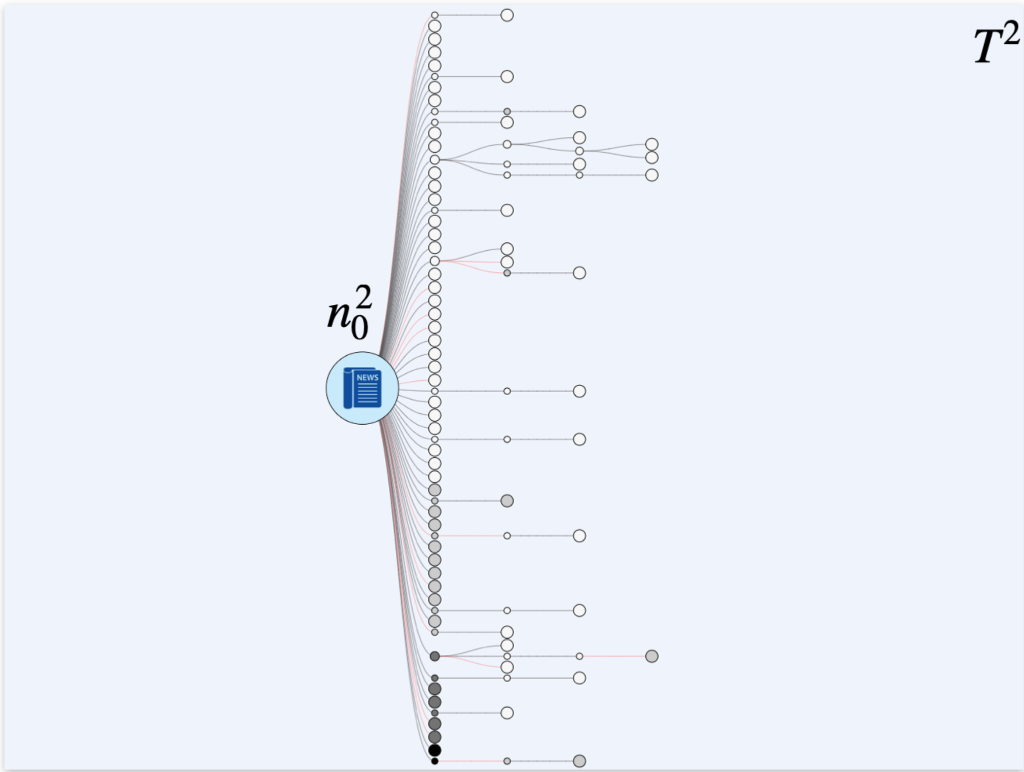


Figure 3.4: Example of Compact hierarchy: $width(n_0^2) = 60$, $\#significant(n_0^2) = 0$, Threshold values: $N = 25$, $L = 4$, $GF_{Compact} = 0.25$, $tolerance = 0.15$

Unspecified. Those Unspecified hierarchies may contain both elongated and compact subtrees, having a mixed structure or several compact clustered regions. In this chapter, we focus on the study of the layouts that fit in well with Elongated and Compact ones.

3.1.2 Algorithm

In this section, we depict the main strategy followed to categorise the inner shapes of hierarchical data (see Algorithm 1). As inputs, it takes the entire hierarchy (T^k), the root node (n_0^k), and threshold values. It is noteworthy that n_i^k can be any node of the hierarchy. As we stated above, the algorithm uses the threshold values to check some properties of the hierarchy (N, L, D, $GF_{Elongated}$, $GF_{Compact}$, $tolerance$).

First, in lines 1–12, we check if the root node has an Elongated Tendency (see $ET(n_0^k, L, GF_{elongated})$), verifying the conditions stated in Equation (4)). If so, we go through all the significant children ($getAllSignificantChildren()$ to test their compactness (using the method $CT(n_i^k, L, GF_{compact})$). Only if no compact subtree exists, the hierarchy will be categorised as Elongated. Otherwise, the hierarchy is elongated in the first level but contains some compact structure and thus, it is categorised as Unspecified.

Secondly, in lines 13–29, the algorithm finds a node with Compact Tendency, n_c^k, (i.e. a node that fulfils the condition of Equation (5), $CT(n_0^k, L, GF_{compact})$). Note that this node can be directly the root node of the hierarchy, n_0^k, or any node close enough to the root node, $dist(n_i^k, n_0^k) < D$ (see lines 13–16). Then, similarly to the Elongated case, we traverse all the significant children ($getAllSignificantChildren()$ to test their elongation (using the method $ET(n_i^k, L, GF_{elongated})$). Again, only if no elongated subtree exists, the hierarchy will be categorised as Compact. Otherwise, the hierarchy is compact in the first level but contains some elongated structure and thus, it is categorised as Unspecified.

Finally, in case the algorithm does not find elongated or compact tendencies in the root node, we categorise the hierarchy as Unspecified.

3.1.3 Analysis of the Categorisation

In the previous sections, we introduced a categorisation method to detect the tendencies in hierarchical structures, focusing on the factors that influence their shape. By fixing certain values, we aim to distinguish between **elongated** and **compact tendencies** in hierarchies. Specifically, we examine the number of direct children of the root node, denoted as $width(n_i^k)$, which is one of the conditions that determine the hierarchy's **tendency**—elongated when below a threshold value, N, and compact when above. Furthermore, we identify significant nodes in the hierarchy, which help control the expansion and maintain the consistency of the hierarchy's shape. We also explore the concept of the **growing factor**, GF, which describes how sublevels maintain the hierarchy's tendency as it grows. In the case of an elongated tendency, the growing factor is defined with a maximum value, $GF_{Elongated}$, below which we consider that the elongated tendency is maintained. Conversely, for compact tendencies, a minimum value, $GF_{Compact}$, is set to determine when the compact tendency is preserved. These values help control the hierarchy's expansion and ensure the consistency of its shape. In this section, we will analyse our choices for these fixed values and how they contribute to detecting hierarchical tendencies. The datasets analysed are derived from real online conversations in news contexts, providing a practical foundation for the analysis. The subsequent simulations are based on these examples to ensure relevance and applicability. For the analysis of the N value, we will utilise two elongated, two compact, and two unidentified datasets,

Algorithm 1 Hierarchical Data Categorisation Algorithm

Require: T^k, n_0^k, $GF_{elongated}$, $GF_{compact}$, $L \geq 0$, $D \geq L$, $0 \leq tolerance \leq 1$, N

if $ET(n_0^k, L, GF_{elongated})$ **then**
  sign = $getAllSignificantChildren(T^k, n_0^k, L, tolerance)$
  isCompact $\leftarrow$ False
  for all n_i^k in sign **do**
    isCompact $\leftarrow CT(n_i^k, L, GF_{compact})$
  end for
  if isCompact **then**
    Return Unspecified
  else
    Return Elongated
  end if
end if
if $CT(n_0^k, L, GF_{compact})$ **then**
  $n_c^k \leftarrow n_0^k$
else if $\exists n_i^k : CT(n_i^k, L, GF_{compact}), dist(n_i^k, n_0^k) < D$ **then**
  $n_c^k \leftarrow n_i^k$
end if
if $\exists n_c^k$ **then**
  sign = $getAllSignificantChildren(T^k, n_c^k, L, tolerance)$
  isElongated $\leftarrow$ False
  for all n_j^k in sign **do**
    isElongated $\leftarrow ET(n_j^k, L, GF_{elongated})$
  end for
  if isElongated **then**
    Return Unspecified
  else
    Return Compact
  end if
end if
Return Unspecified

each with varying numbers of total nodes, inner structures, and direct children (see Table 3.2). For tolerance analysis, we will focus on one elongated and one compact dataset (datasets 2 and 3 of Table 3.2, respectively). Lastly, for $GF_{Elongated}$ and $GF_{Compact}$, we will use the same two elongated and two compact datasets previously employed in the N value analysis (datasets 1, 2, 3, and 4 of Table 3.2).

N Value

The tendency of a hierarchical data structure depends on various factors, including the threshold value N. By fixing N near 25, we standardise the process of classifying hierarchies as elongated or compact, simplifying comparisons across datasets. One of the conditions is that a hierarchy is considered elongated when the number of direct children of the root node (the "width") is below N, and compact when the width exceeds this value. Additionally, other parameters are fixed for consistency: tolerance is set to 0.15, $GF_{Elongated}$ is 2, and $GF_{Compact}$ is 0.25. This ensures that only the N value is adjusted during the analysis. It is worth noting that the size of the hierarchy plays a significant role, as $N = 25$ may not be effective for larger hierarchies. In this study, the total number of nodes in the hierarchies ranges between 65 and 265.

In Table 3.2, we explore how fixing N to 25 is a reasonable choice for the selected examples. The green cells represent the correctly identified tendency—elongated or compact—according to the chosen threshold for N. For instance, in the case of Dataset 1, with a width of 21 and $N = 25$, the hierarchy is classified as elongated, as expected, since the width is below the threshold. However, reducing N to 21 would result in it not being classified as elongated. This is evident in Figure 3.5.a, where Dataset 1 clearly exhibits an elongated tendency. The nodes are distributed vertically, and in the radial layout (see Figure 3.5.b), the compactness typically seen in fully occupied levels is notably absent with one branch standing out having an elongated tendency.

Dataset	Width	N	Elongated Tendency	Compact Tendency
1	21	25	Yes	No
1	21	21	No	No
2	10	25	Yes	No
2	10	9	No	Yes
3	26	25	No	Yes
3	26	26	Yes	No
4	64	25	No	Yes
4	64	65	Yes	No
5	36	25	No	Yes
5	36	36	Yes	No
6	10	25	No	No
6	10	9	No	Yes

Table 3.2: Elongated and Compact Tendencies Based on Width (total number of direct children) and N Values. Green cells represent correctly classified cases, while red cells indicate misclassified cases.

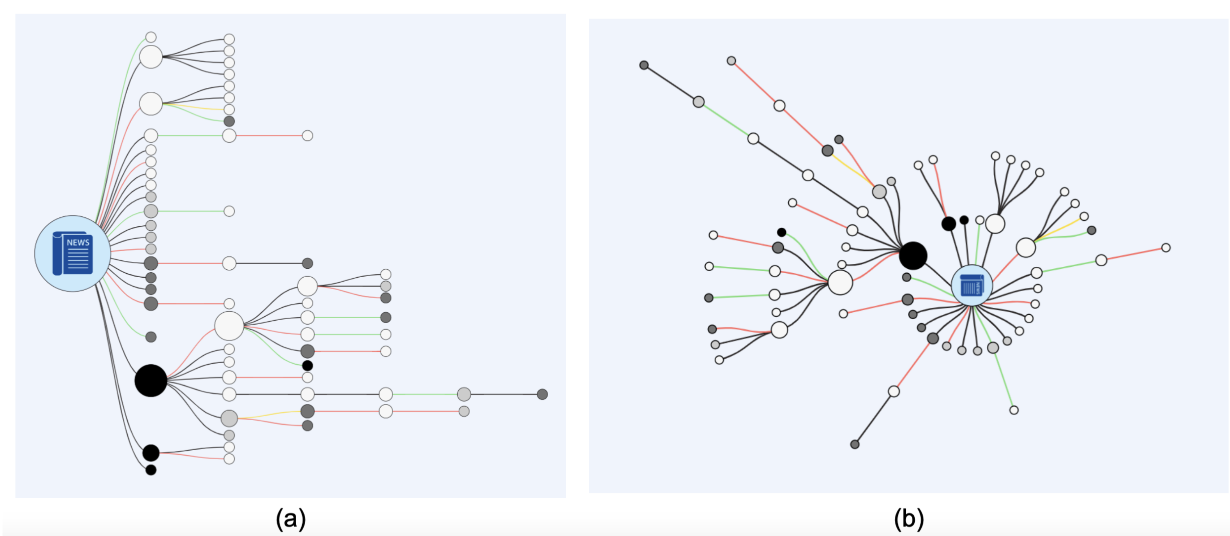
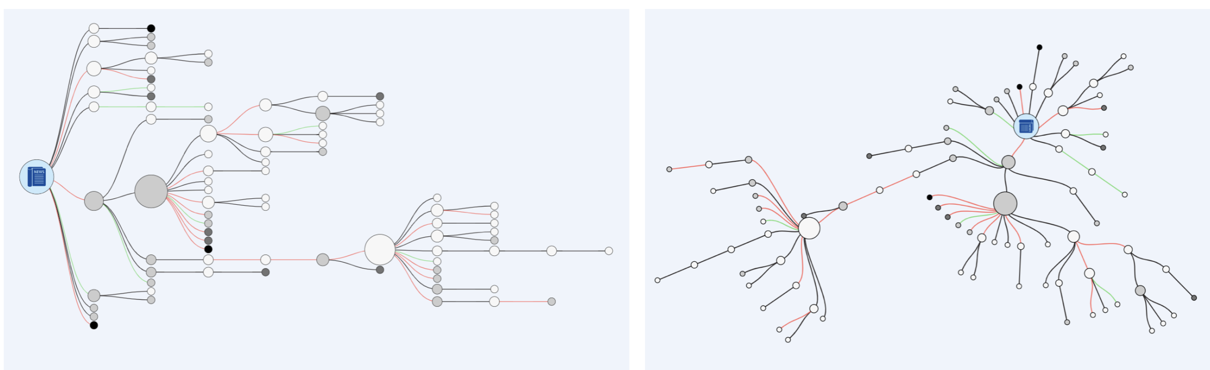


Figure 3.5: Dataset 1 displayed in (a) Tree layout and (b) Radial Layout

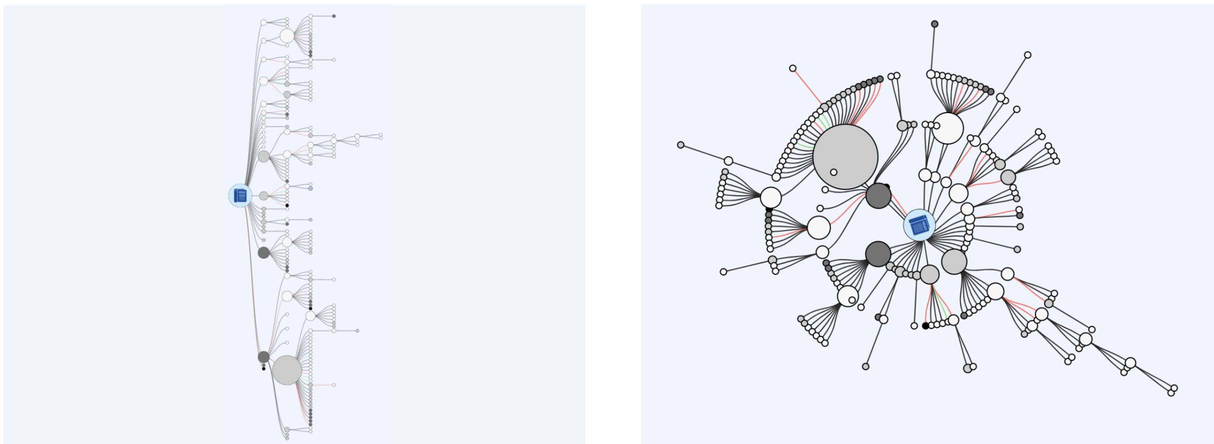
The value of N is influenced by the width, meaning that different datasets may exhibit an elongated tendency at different values of N. For instance, in Dataset 1, a threshold of $N = 21$ prevents the dataset from being classified as elongated, while in Dataset 2, $N = 9$ is the value that changes classification to compact. Therefore, we explored various datasets to determine the threshold value for N in hierarchies with a total number of nodes ranging between 65 and 265. We concluded that $N = 25$ is the critical value where the hierarchy becomes too wide to be considered elongated. As shown in Figure 3.6.a, Dataset 2 is elongated, as it has very few direct children and lacks the growth of compactness. The radial layout (see Figure 3.6.b) reveals an irregular shape, while the tree layout maintains a more structured and coherent form.



(a) (b)

Figure 3.6: Dataset 2 displayed in (a) Tree layout and (b) Radial Layout

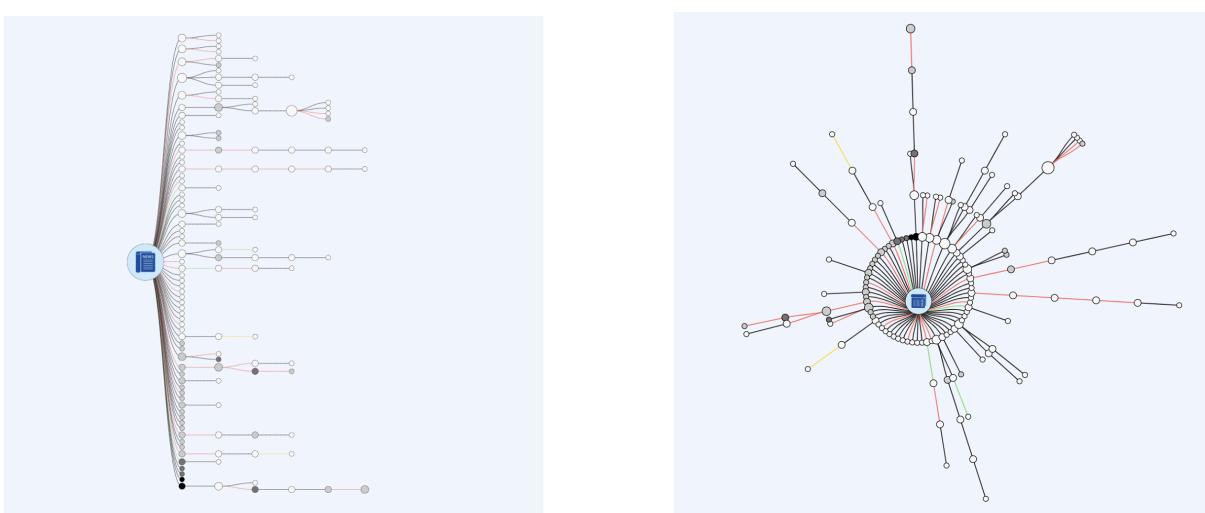
Turning to Dataset 3, with a width of 26, we observed that with $N = 25$, it is classified as having a compact tendency. This classification is confirmed in Figure 3.7. It is evident that when using the tree layout, the hierarchy becomes too wide, making it difficult to appreciate the finer details (see Figure 3.7.a). In contrast, the radial layout optimises the canvas usage, allowing for a clearer view of the continued compactness, as shown in Figure 3.7.b. The threshold value of 25 is further confirmed in this case, as setting $N = 26$ classifies the hierarchy as elongated (see 3.2), which does not accurately reflect the compact structure.



(a) (b)

Figure 3.7: Dataset 3 displayed in (a) Tree layout and (b) Radial Layout

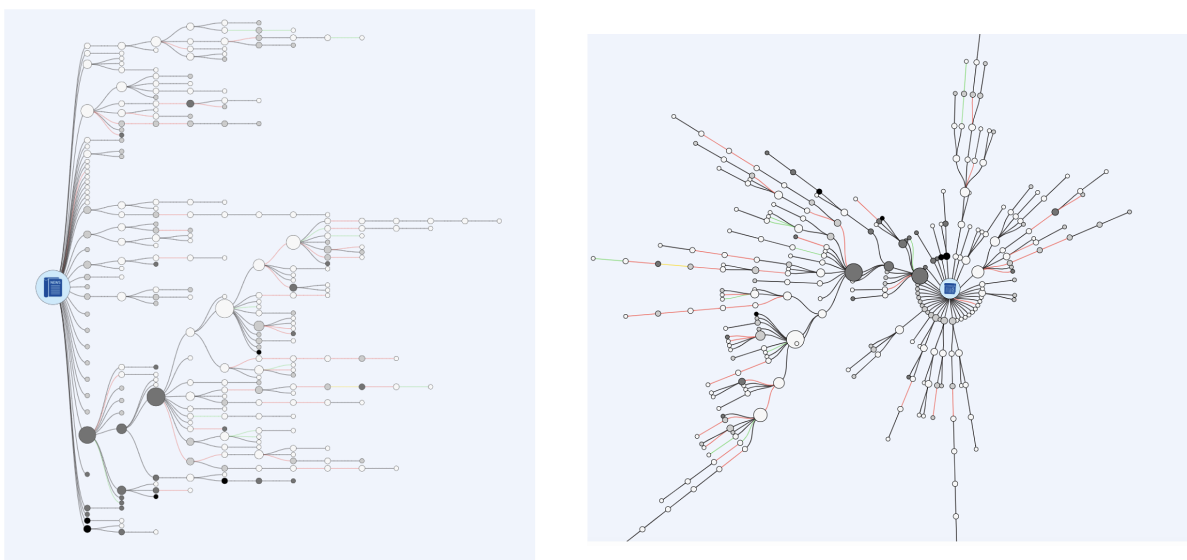
For Dataset 4, which has a larger width of 64, setting $N = 25$ correctly classifies the hierarchy as compact, as shown in Figure 3.8.b. To change the tendency to elongated, a significantly larger value of N, such as 65, would be required. However, this is an unusually high value and does not reflect the hierarchy's actual structure, as demonstrated in Figure 3.8.a. The choice of $N = 25$ is a reasonable threshold in this kind of dataset with a mean of 157 total nodes, effectively capturing the hierarchy's true structure without resorting to excessively large values.



(a) (b)

Figure 3.8: Dataset 4 displayed in (a) Tree layout and (b) Radial Layout

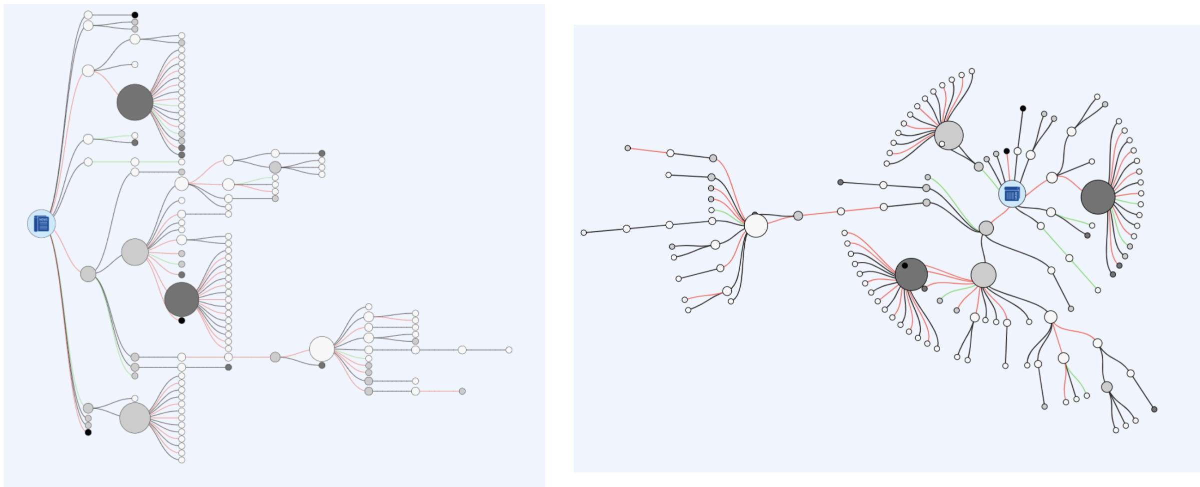
Furthermore, some datasets initially exhibit either elongated or compact tendencies but later take on a different form, such as Datasets 5 and 6, which we classify as unspecified. For example, setting $N = 25$ for Dataset 5, which has a width of 36, incorrectly classifies it as compact, as shown in Figure 3.9. Although the width is large, the hierarchy exhibits some elongated tendencies within its branches, with certain branches being very long. However, it does not maintain consistent compactness, making the compact classification unsuitable (see Figure 3.9.b). On the other hand, when we set $N = 36$, the hierarchy is classified as elongated, which is not correct either, due to the initial compactness observed at the root level (see Figure 3.9.a). As mentioned before, N is just one factor in determining the tendency.



(a) (b)

Figure 3.9: Dataset 5 displayed in (a) Tree layout and (b) Radial Layout

Lastly, for Dataset 6, which has a width of 10, similar to Dataset 2, we observed that while the hierarchy initially appears elongated (see Figure 3.10), with few nodes directly connected to the root and long branches, some nodes at the second level become more compact, making the classification uncertain. Setting $N = 9$ classifies it as compact, while $N = 25$ classifies it as elongated. However, as shown in Figure 3.9 a and b, the hierarchy is neither strictly elongated nor compact. Therefore, after a careful review of multiple datasets, we conclude that $N = 25$ provides the most consistent and accurate classification across all datasets.



(a) (b)

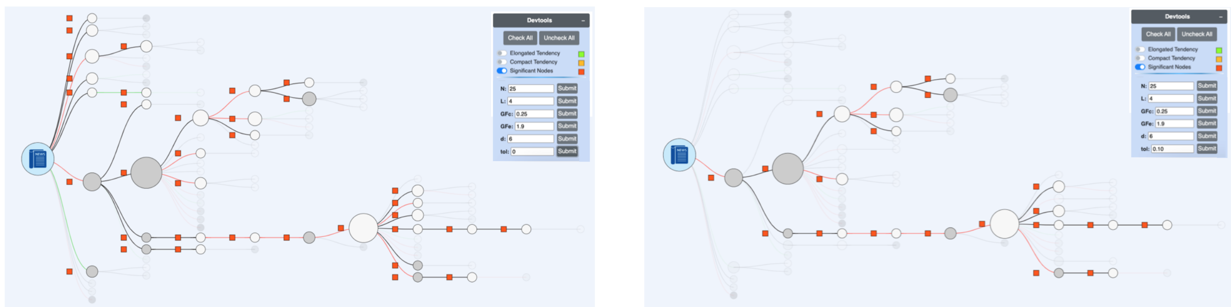
Figure 3.10: Dataset 6 displayed in (a) Tree layout and (b) Radial Layout

As we have already explored, the threshold value N can be influenced by the available datasets. Now, we also think that the canvas size may impact this value. The visual representation of the hierarchy can vary depending on the device being used—whether it's a standard laptop, a large desktop screen, or a mobile phone. While the concept of N remains constant, the way the data is displayed changes based on the screen size. For consistency, we have adjusted N to suit a medium-sized screen (28–38 cm W and 16.5–23 cm H), such as a typical laptop, ensuring that the visualisation remains clear and balanced across common devices.

Tolerance Value

The Growing Factor is another key factor used to determine the tendency of the hierarchy, as shown in formulas 3.4 and 3.5. Significant nodes denoted as $significant(n_i^k)$, are defined as nodes that have a relevant number of descendants compared to their parent's descendants. These nodes play an important role in detecting the Growing Factor (GF), as only these nodes are used to define the GF of a hierarchy. To identify the significant nodes, a **tolerance** threshold is set. In this thesis, we fixed the value of **tolerance** to 0.15. However, this value can be adjusted depending on the available datasets. To justify our choice, we take two datasets, one elongated (dataset 2) and one compact (dataset 3), and explore the impact of different tolerance values on the classification of significant nodes.

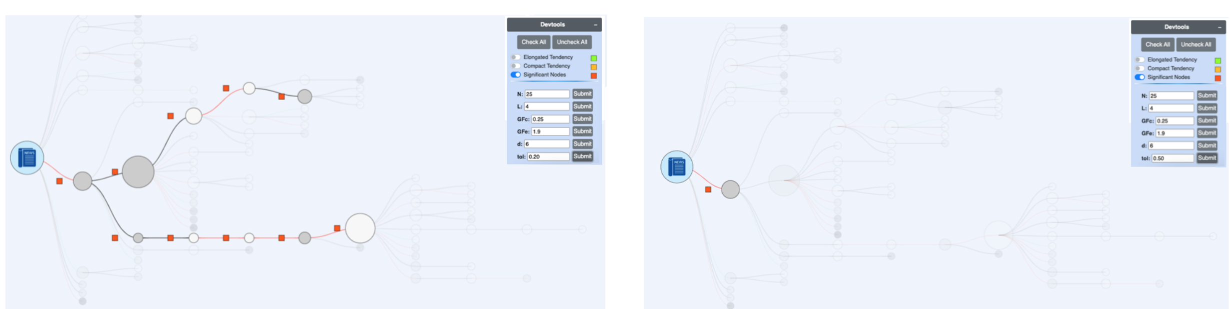
For example, for a tolerance value of 0 in Dataset-Elon, nearly all nodes in the hierarchy are considered significant, as illustrated in Figure 3.11.a. However, many of these nodes lack relevance in practice. For instance, nodes directly connected to the root without any children contribute little and can be excluded. When the tolerance is adjusted to 0.10, as shown in Figure 3.11.b, the hierarchy simplifies, retaining only the largest direct node connected to the root and some of its children. Thus, as the tolerance value increases, the number of significant nodes decreases. When the tolerance is raised to 0.20, as shown in Figure 3.12.a, some of the less relevant nodes from the larger branch are eliminated.



(a) (b)

Figure 3.11: Significant nodes for Dataset-Elon with tolerance set to a) 0 and b) 0.10

From a tolerance of 0.20 to 0.40, the number of significant nodes remains unchanged; however, when the tolerance reaches 0.50, as shown in Figure 3.12.b, only a single node remains. The final tolerance value should be chosen with care. If set too high, significant descendants at the root of the tree may go undetected, which can result in hierarchies being misclassified as compact due to the root appearing as the only significant node. Conversely, if the tolerance is set too low, such as 0.10, the hierarchy becomes cluttered with unnecessary nodes, reducing its interpretability. In this dataset, we can consider 0.20 the mid-way.



(a) (b)

Figure 3.12: Significant nodes for Dataset-Elon with tolerance set to a) 0.20 and b) 0.50

When exploring another dataset, Dataset-Comp, as shown in Figure 3.13, the effect of different tolerance values becomes clear. In Figure 3.13.a, with a tolerance set to 0.10, three nodes connected to the root remain significant along with their children. However, when the tolerance is increased to 0.15 (see Figure 3.13.b), one of the nodes is removed as it is less significant, with the remaining node being much larger. At a tolerance of 0.20 (see Figure 3.13.c), there is no change. Moving to 0.30 (see Figure 3.13.d), another child node is removed, and at 0.40 (see Figure 3.13.e), all the nodes are eliminated. It's important to find a middle ground, as selecting an appropriate tolerance ensures a balance between retaining meaningful nodes and removing redundant ones.

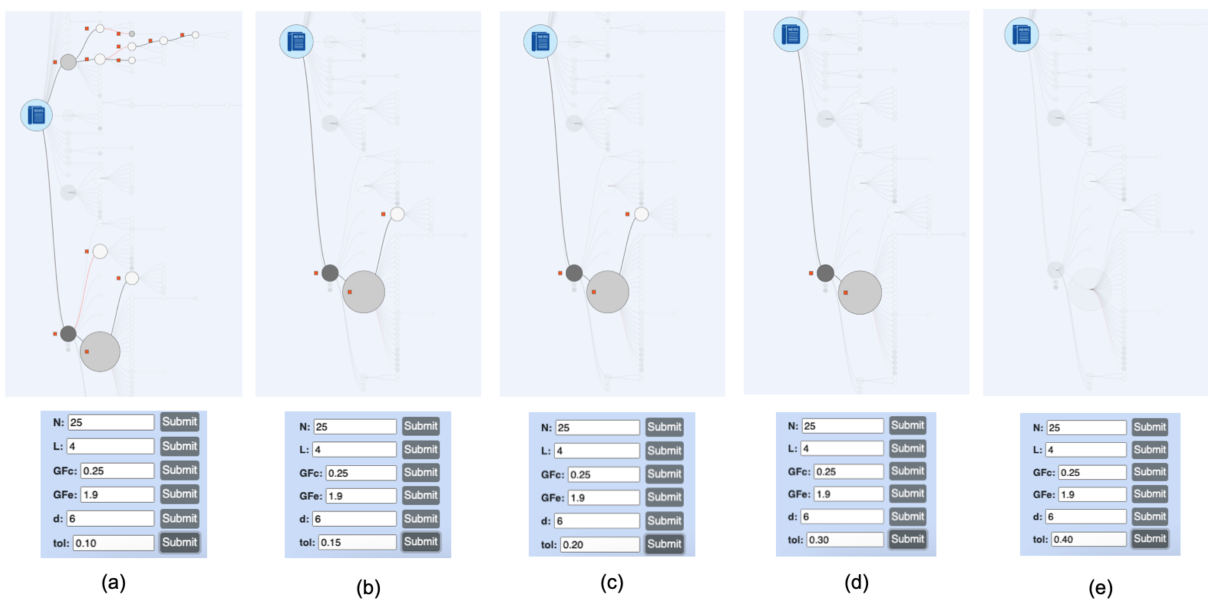


Figure 3.13: Significant nodes for Dataset-Comp with tolerance set to a) 0.10, b) 0.15, c) 0.20, d) 0.30 and e) 0.40

Upon returning to Dataset-Elon and exploring with a tolerance of 0.15 (see Figure 3.14), it became apparent that more nodes were retained compared to Dataset-Comp, with the largest node still present. Therefore, we concluded that a tolerance between 0.15 and 0.20 is appropriate for this thesis. In most cases, we used 0.15 because it provided a good balance for the types of hierarchies we were analysing.

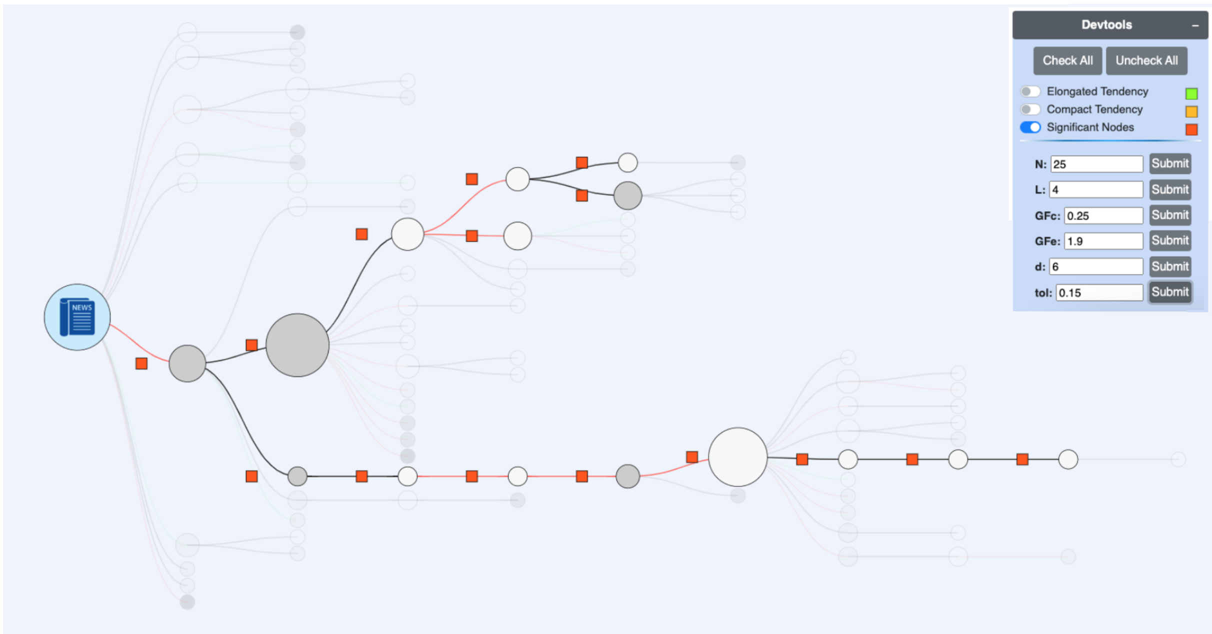


Figure 3.14: Significant nodes for Dataset-Elon with tolerance set to 0.15

$\mathbf{GF_{Elongated}}$ and $\mathbf{GF_{Compact}}$ Values

Similarly, factors such as adjusting the elongated and compact dimensions ($GF_{Elongated}$ and $GF_{Compact}$) of the Growth Factor also influence the overall structure of the hierarchy. These limits set maximum and minimum values from which the hierarchy is considered elongated or compact at a given level, respectively. In this thesis, hierarchies are considered to exhibit an elongated tendency at a given level if their Growth Factor is less than 2, and a compact tendency if their Growth Factor is greater than 0.25. In Table 3.3, we justify these numbers across four datasets, where two exhibit an elongated tendency and two a compact tendency.

It can be seen in Table 3.3 that the changes in $GF_{Elongated}$ and $GF_{Compact}$ values directly influence the tendency for each dataset. For example, in the first dataset, the Growing Factor is set to 0.09, which is smaller than the GF, resulting in a compact tendency. In the second dataset, the Growing Factor is 0.6; when $GF_{Elongated}$ is set to 2 and $GF_{Compact}$ to 0.25, the tendency shifts to elongated. Similarly, for Datasets 3 and 4, where the Growing Factor is 1, setting $GF_{Elongated}$ to 2 and $GF_{Compact}$ to 0.25 initially results in a compact tendency, but as $GF_{Compact}$ is adjusted to 0.9, the tendency shifts to elongated. These changes demonstrate how variations in $GF_{Elongated}$ and $GF_{Compact}$ influence the hierarchy's tendency within each dataset. As observed, different Growing Factors require appropriate threshold values to ensure accurate results. If 0.01 is selected, for

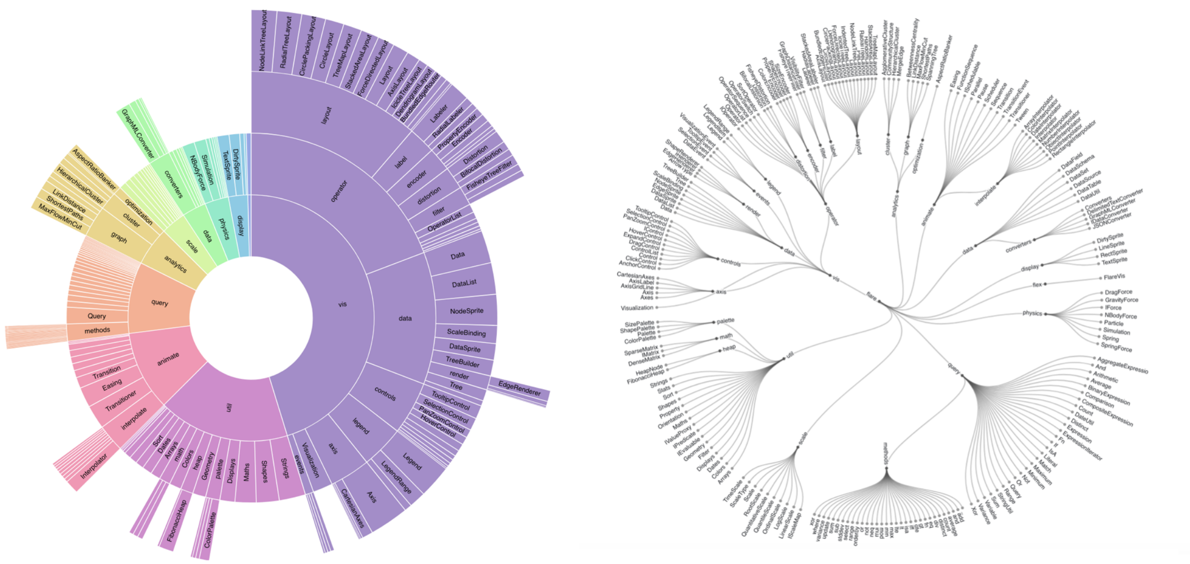
Dataset	Growing Factor	GF_Elongated	GF_Compact	Tendency
1	0.09523	2	0.25	Elongated
1	0.09523	0.09	0.09	Compact
2	0.6	2	0.25	Elongated
2	0.6	0.5	0.5	Compact
3	1	2	0.25	Compact
3	1	0.9	0.9	Elongated
4	1	2	0.25	Compact
4	1	0.9	0.9	Elongated

Table 3.3: Dataset, Growing Factor, and Growing Factors of Elongated and Compact Tendencies. Green cells represent correctly classified cases, while red cells indicate misclassified cases.

example, Datasets 2, 3, and 4 will produce incorrect tendencies. The choice of 2 for $GF_{Elongated}$ and 0.25 for $GF_{Compact}$ is deliberate. A high $GF_{Elongated}$ value (e.g., 2) emphasises the tendency for nodes to extend further, creating an elongated hierarchy. Conversely, a low $GF_{Compact}$ value (e.g., 0.25) allows for tighter grouping, resulting in a compact structure. These values were selected because they provide a clear distinction between the two tendencies, making it easier to study their effects on the hierarchy.

3.1.4 Visualising Hierarchical Data

The literature offers a huge visual bibliography of hierarchical visualisations (advanced visualisations), [159] that includes more than 341 techniques, which makes it challenging to find the best-fitted visualisation for different hierarchical structures. They are classified mainly into two categories: implicit and explicit visualisations [161]. **Implicit** hierarchical visualisations represent parent-child relationships with positional encoding using shapes within other shapes (e.g., using rectangles both in treemaps, sunburst and icicle diagrams, and circles in circle packing) (see Figure 3.15.a). While on the other hand **explicit** visualisations represent these relationships with lines, such as radial tree layout (see Figure 3.15.b). Elijah Meeks [128] proposed four types of layouts: pack layouts (circle packing), node-link layouts (tree and radial layout), partition layouts (sunburst diagram and icicle diagram), and treemaps, explaining when to use each visualisation depending on the type of the data or the VisTask to perform. For instance, they recommended using partition layouts for analysing numerical data, and node-link layouts for analysing paths.



(a) (b)

Figure 3.15: Example of (a) Implicit and (b) Explicit hierarchical visualisations

It should be pointed out that most of these techniques are derived from each other by adding additional features or advancing the existing techniques. For example, Multivariate Bubble Treemap by [213] has the same visual as Bubble Treemap by [71] but it includes additional glyphs. Moreover, some of these techniques (e.g. radial, tree, etc.) are the most used [141]. In the following, we analyse the most common implicit and explicit hierarchical layouts. The use of implicit layouts [161], such as treemaps, Circle packing, Icicle or Sunburst diagrams, where the parent-child relationships are coded using relative locations between parents and children, are space efficient due to their high compactness. However, it is harder to read huge-sized, broad and deep hierarchies with these layouts. In addition, as these layouts place nodes in a nested way without leaving empty spaces, the inclusion of icons and glyphs, to represent additional attributes in case of data is multivariate, becomes more difficult. Our emphasis lies in explicit layouts to accommodate multivariate elements that will be explained in Section 3.2.

We illustrate these ideas by displaying hierarchical data using the **Circle packing layout** in Table 3.1. This layout is the circular version of a treemap where nodes are packed in circles. The root node (e.g., the first tweet on Twitter, now X) is represented as the biggest circle that contains all the nodes (see the big blue circle in Table 3.1.a). Direct children of a node are placed inside the circle relative to their parent node. The more children a node has, the larger the circle is.

While its compactness is an advantage with small-sized data that has few levels, it can be a disadvantage, especially with broad data that has most of its nodes on the same level as it becomes overcrowded easily (Table 3.1.b). Moreover, where siblings with a different number of children, the circle packing uses different sizes in the same levels, losing the perception of the relationships between them. On the other hand, in narrow and deeper hierarchies, the nested circles make it difficult to understand the hierarchy structure (Table 3.1.a). Moreover, long spines (i.e., large narrow branches) are visualised as many concentric circles, because of the consecutive placement of children onto their parents' circles, thus, making it hard to appreciate the different levels of the hierarchy and the parent-child and sibling relationships (see Table 3.1.c). In addition, if we also add pictorial representations of the multivariate attributes, the visualisation ends up being even more crowded.

Unlikely to treemaps, Sunburst diagram [180] and Icicle plot [111] **implicit** layouts show the parent-child relationships by placing the child nodes next to their parents nodes, circularly in Sunburst and linearly in Icicle plot. While these two layouts could better show the hierarchy than treemaps, and use the space more efficiently, they will have similar problems displaying large-sized data that has long spines. Especially, when the data is big the very outer leaves on hierarchies are displayed as very thin rectangles on both layouts thus, this will make the graphs harder to analyse in a complete view. Additionally, in an Icicle plot, the same level nodes are placed next to each other in a horizontal line and when we consider broad data that has a compact tendency, the visualisation has to be either horizontally extended beyond the screen, or shrunk with zoom-out, in both ways it loses the global view. While due to its circular shape, Sunburst will not have this problem, when the data is too dense it can become crowded easily, and an overcrowded visualisation hinders the effective exploration of the data. Moreover, if multiple attributes are integrated with these visualisations, they will be harder to read and analyse on overview and it would be impossible to see multivariate attributes on the slimmer nodes.

On the other hand, **explicit** layouts[160], i.e., node-link layouts (tree, radial, force-directed), have better readability over viewing hierarchies as each node is shown individually [193]. However, they are also known for using space not very efficiently due to the lines that connect parent and children nodes occupying space and generating empty backgrounds. Indeed, this can be an advantage while visualising multivariate attributes as there is plenty of space to map additional elements such as glyphs. Also, they can visualise data both on the nodes and edges. In the

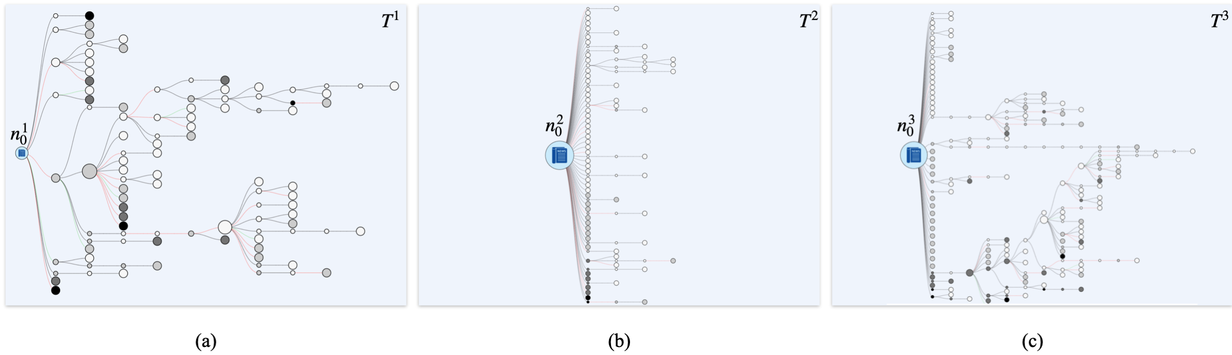
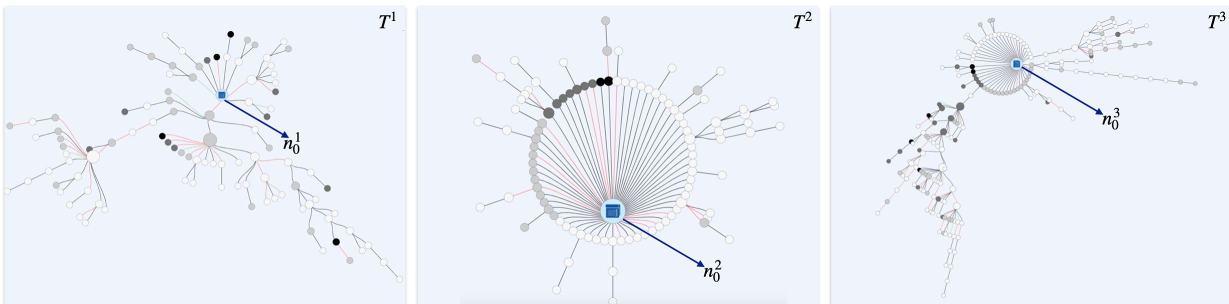


Figure 3.16: Tree layout with the same data sets shown using Circle Packing in Table 3.1. The threshold values used to compute significant nodes and Growing Factors are $N = 25$, $L = 4$, $GF_{Elongated} = 1.9$, $GF_{Compact} = 0.25$, and $tolerance = 0.15$.
(a) Data set categorised as Elongated, (b) Data set categorised as Compact, and (c) Unspecified data set.

following, we show three data sets (elongated, compact and unspecified) through a review of the most significant explicit layouts in Figures 3.16, 3.17, and 3.18.

The **Tree layout** in Figure 3.16 is laid out horizontally. Thus, the depths of the nodes are shown horizontally and, all nodes in the same depth are placed in the same vertical line. Particularly, the tree layout is well organised to visualise narrow structures, it clearly presents the relationships between siblings in different levels (see Figure 3.16.a). However, if a hierarchy has nodes with a lot of direct children on the same level, they will be placed on the same vertical line forming a very long straight column with very small nodes (see Figure 3.16.b), losing details and wasting canvas space. Especially, when the wide data is also crowded at all levels tree layout loses its comprehension and becomes difficult to visualise the entirety of the structure at once (see Figure 3.16.c).

Radial layout (see Figure 3.17) arranges nodes on concentric circles. It is better than the tree layout when the data is broad since it uses space more efficiently by arranging hierarchies circularly. Thus, the Radial layout fits larger amounts of nodes into the canvas (see Figure 3.17b). For whenever its circles are partially filled, human perception through Gestalt's principle of closure [194], could reconstruct them as long as they are sufficiently populated with nodes that are evenly distributed in each level such as in compact data. Thus, the radial layout offers a comprehensible visualisation of hierarchies with compact data as the total number of nodes in each level is in proportion to the number of nodes in other levels (see Figure 3.17.b). Otherwise, when data has elongated tendency characteristics, the perception of closure is lost (see Figure 3.17.a). Moreover, a hierarchy can initially have a perception of closure in the first few levels however, this perception can be lost if the rest of the hierarchy does not follow the initial tendency, such as by having long threads away after compact first few levels. (see Figure 3.17.c)

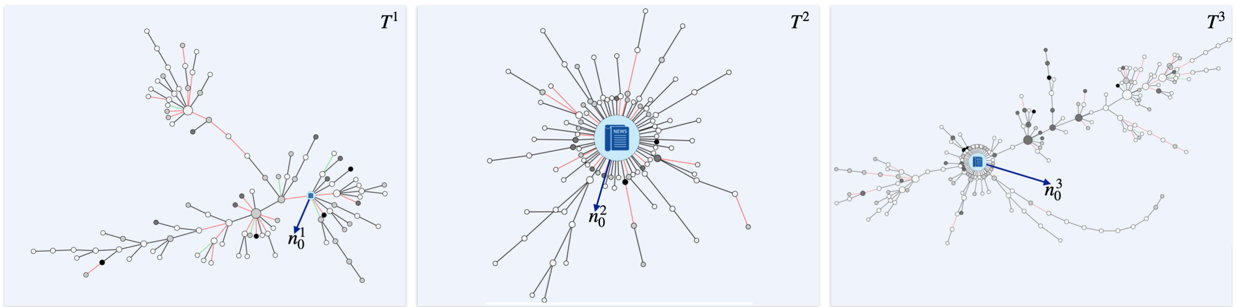


(a) (b) (c)

Figure 3.17: Radial layout with the same data sets shown using Circle Packing in Table 3.1. The threshold values used to compute significant nodes and Growing Factors are: $N = 25$, $L = 4$, $GF_{Elongated} = 1.9$, $GF_{Compact} = 0.25$, and $tolerance = 0.15$.
(a) Data set categorised as Elongated, (b) Data set categorised as Compact, and (c) Unspecified data set.

Force layout (see Figure 3.18) also displays the hierarchy somehow in a circular way. However, unlike Tree and Radial layouts, it does not place the same depth nodes in an ordered alignment, thus, it is not as effective as showing the relationship between sibling nodes but it gives nodes more freedom on the canvas. Also, due to its force-based strategy that uses energy functions to place nodes in the canvas [38], force layout visualises groups (threads) of data in clusters and places them away from each other. Thus, the Force layout could efficiently display broad (see Figure 3.18.b) and large-sized (see Figure 3.18.c) hierarchies in a global view. However, in Figure 3.18.a there is a Force layout with narrow data we can observe that it is difficult to appreciate the relationships between siblings due to the uneven distribution of nodes at each level.

While circle packing has advantages for small-sized data with few levels, the tree layout effectively organises narrow structures for visualisation. On the other hand, the radial layout utilises Gestalt's principle of closure, allowing human perception to reconstruct partially filled circles, making it suitable for compact data with evenly distributed nodes across levels. Meanwhile, the force layout efficiently visualises broad and large-sized datasets. Therefore, we believe that when visualising Elongated structures the most informative layouts are Tree and Circle. When visualising Compact structures the most informative layouts are Radial and Force, and for Unspecified structures Force is the best option. Based on these insights, we propose a rule-based automatic visual mapping system that selects the appropriate layout according to the structure, and we suggest the following



(a) (b) (c)

Figure 3.18: Force layout with the same data sets shown using Circle Packing in Table 3.1. The threshold values used to compute significant nodes and Growing Factors are: $N = 25$, $L = 4$, $GF_{Elongated} = 1.9$, $GF_{Compact} = 0.25$, and $tolerance = 0.15$.
(a) Data set categorised as Elongated, (b) Data set categorised as Compact, and (c) Unspecified data set.

appproach:

- **Elongated Structure** $\longrightarrow$ The selected layout is Tree.
- **Compact Structure** $\longrightarrow$ The selected layout is Radial.
- **Unspecified Structure** $\longrightarrow$ The selected layout is Force.

Additionally, we propose that this automatic visual mapping should be applied to subtrees when they are displayed separately, as a result of a VisQuery on the whole dataset. Since a subtree may exhibit a different structural tendency compared to the original hierarchy to which it belongs, the layout applied to the subtree will be adjusted accordingly. For example, as shown in Figure 3.19, (a) displays the entire hierarchy in a force layout as it has an unspecified structure. However, when a subtree of this hierarchy is visualised (see b), we calculated that it has an elongated structure. As a result, it should be visualised in a tree layout, as shown in (c).

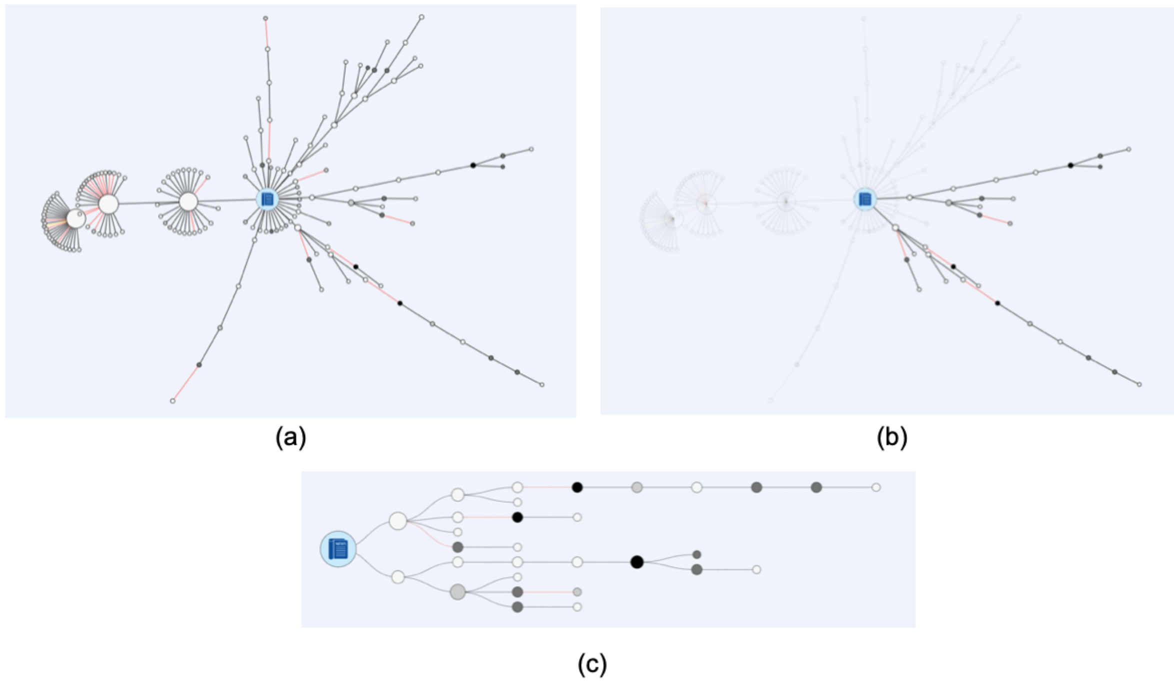


Figure 3.19: a) Entire hierarchy, b) Subtree highlighted on the entire hierarchy, and c) Subtree visualised separately.

3.2 Visualisation of Multivariate Data in Hierarchies

In this section, we first explore the concept of multivariate data to formalise it and cluster the data, helping us visualise the different types. We then delve into visualising multivariate data, discussing the most effective methods for representing each cluster.

3.2.1 Notation of Multivariate Data

In the following, we formalise the types of features contained in **multivariate data**. This formalisation will help us later in Section 3.2.2 to analyse the design elements (e.g., glyphs, icons) that best symbolise them. It should be noted that each node of the hierarchy contains data from which a set of $numF$ predefined **features**, $\mathcal{F} = \{f_1, \dots, f_{numF}\}$, will be extracted. We denote $data_{n_i^k}$ as the data relative to the node n_i^k. Analogously, we define $data_{e_{i,j}^k}$, and $data_{n_0^k}$ as the data contained in each edge $e_{i,j}^k$ of the tree, and in the root node n_0^k respectively. Thus, **the total**

information stored in a tree, T^k, is

$$\mathcal{D}ata_{T^k} = \mathcal{D}ata_{Nodes^k} \cup \mathcal{D}ata_{Edges^k} \cup \{data_{n_0^k}\} \tag{3.6}$$

being

$$\begin{aligned}\mathcal{D}ata_{Nodes^k} &= \{data_{n_i^k}\},\ \forall n_i^k \in Nodes^k,\\ \mathcal{D}ata_{Edges^k} &= \{data_{e_{i,j}^k}\},\ \forall e_{i,j}^k \in Edges^k\end{aligned}$$

Above we introduced our multivariate data categorisation neutrally without concretely basing it on any type of data. It should be noted that this categorisation can be applied to any multivariate data. Moreover, to explain our idea further we will present an example about analysing conversations in social media. As an example of a hierarchy T^k, the root node contains as $data_{n_0^k}$ the text from X (previously Twitter), a social network: *"A young North African is beaten after a violent robbery of an old woman"*. Then, a direct child n_j^k of the root node n_0^k will contain as data $data_{n_j^k}$ the message that replies to this tweet: *"A fucking piece of shit, he and those who lynch him, let's see if we understand that we live in a civilisation and not in the jungle. The thief is detained and the police are called."* In this example, the edge between both nodes n_0^k and n_j^k does not contain any related data ($data_{e_{0,j}^k} = \emptyset$). However, when a direct child n_i^k of a node n_j^k, contains data, $data_{n_i^k}$, that relates to the data in $data_{n_j^k}$, the edge between them contains the data of both nodes: $data_{e_{j,i}^k} =< data_{n_j^k}, data_{n_i^k} >$.

Based on previous definitions, we state the **labelling function**, $\mathcal{L}$, as a function that associates a list of features to each element (either node or edge) of T^k according to its information:

$$\mathcal{L} : \mathcal{D}ata_{T^k} \longrightarrow\ f_1 \times f_2 \times \cdots \times f_{numF},\ f_i \in \mathcal{F} \tag{3.7}$$

Moreover, we can define $\mathcal{L}$ separately for each type of tree element. Thus, we define $\mathcal{L}_{Nodes^k}$ and $\mathcal{L}_{Edges^k}$ with their related information, $\mathcal{D}ata_{Nodes^k}$ and $\mathcal{D}ata_{Edges^k}$ respectively.

$$\begin{aligned}\mathcal{L}_{Nodes^k} : \mathcal{D}ata_{Nodes^k} \longrightarrow\ & f_1 \times \cdots \times f_{numF_{Nodes}},\\ & f_i \in \mathcal{F}_{Nodes}\end{aligned}$$

$$\mathcal{L}_{Edges^k} : \mathcal{D}ata_{Edges^k} \longrightarrow f_1 \times \cdots \times f_{numF_{Edges}},$$
$$f_i \in \mathcal{F}_{Edges}$$

It is worth noting that each dimension, or feature, of $\mathcal{F}$, f_i, defines a variable in the domain that can be numerical (discrete or continuous) or categorical (nominal or ordinal). Some of them are independent variables, but others are dependent allowing to model cause-and-effect relationships.

Additionally, we group related features in $NumC$ clusters depending on their semantics, $\mathcal{C} = \{c_1, ..., c_{NumC}\}$, where $c_i = \{f_1^i, ..., f_{numF^i}^i\}$, being $\forall f_s^i \in \mathcal{F}, 1 \leq s \leq numF^i$.

3.2.2 Visualising Multivariate Data

As discussed, the data is hierarchical but also possesses multivariate attributes (features) that need to be visualised and integrated with these hierarchical visualisations. Multivariate data implies visualising a high amount of features without overwhelming users' perceptions. In the previous section, we formalised multivariate data, introducing equations 3.6 to capture all information on nodes and edges, and in 3.7 to include annotated labels. Furthermore, we clustered these labels based on similarity. Consequently, there are various types of multivariate attributes to visualise, and selecting the most suitable visualisation method is essential. There is a wide range of approaches in the literature proposing different visual elements and techniques to communicate a high number of features at once, such as colours, shapes, icons and glyphs. Although *one-by-one* based approaches depict features side by side [201], *all-in-one* approaches group together interrelated features [135].

Multivariate data visualisations are challenging in themselves but even more so when they should be integrated into hierarchical visualisations. On the one hand, implicit hierarchical layouts, due to their high compactness use space efficiently but are left with little empty space to integrate visual elements representing multivariate data. On the other hand, the low compactness of explicit layouts can be turned into an advantage, as the empty space, and also edges could be used for visualising features [213]. Concretely, in this research, we presented explicit layouts (Tree, Radial, Force) to incorporate the visualisation of multivariate data.

Firstly, let's consider **ordinal** data with a list of values, for instance, a three-valued size feature (small, medium, large), and each node has one of those different values. As these kinds of data should be represented in every node, the best way to visualise them is directly on the visualisations.

For example, they could be directly shown on nodes or edges as hue colours. As usually ordinal features have more abstract meanings, using icons is not ideal. Moreover, using shapes fixed next to each other on the visualisations can complicate them very easily. If the node contains more than one ordinal feature or one ordinal feature combined with other types of features then glyphs can be also an option [51].

Secondly, regarding **nominal** features (e.g., hair colour blond - yes or no -), to not overwhelm the user perception, only those nodes or edges that fulfil the property will display its visual element. For example, only nodes representing blonde individuals will display a visual representation of blonde hair. Moreover, features with concrete meanings would be well represented with icons. For example, if recycling is the tagged feature it could be easily visualised with a recycle icon. On the other hand, for **more abstract features** such as being sarcastic and intolerant, glyphs could be a good option and each feature can be mapped onto this glyph, with unique hue colours, without any additional symbol or icon, that can be visualised either as *one-by-one* or *all-in-one*.

3.3 Discussion

Our categorisation, as described in this chapter, is based on some features of the hierarchical data to detect Elongated and Compact tendencies starting from the node n_i^k, such as the significance of the nodes ($significant(n_i^k)$), how they grow ($width(n_i^k)$, and $GrowingFactor(n_i^k, s)$), and the scope of growth (controlled by the analysis of L levels). To do so, we needed some threshold values: for computing significant nodes, $tolerance$; for the number of direct children (width of n_i^k), N; for calculating the tendency along with several levels, L; and for the maximum and minimum values of the growing factor of elongated and compact structures, $GF_{elongated}$ and $GF_{compact}$, respectively. It should be noted that these threshold values may depend on the data, especially determining the N value, which indeed helps to establish some kind of borderline between Elongated and Compact structures. Although we presented an analysis of these threshold values, we think that they deserve further study, discarding the constant values such as L (in this thesis set to 4) and considering, for example, their computation based on some percentage of nodes of the tree.

In this dissertation, we mainly focused on the categorisation of hierarchical data independently of the canvas dimensions as we conducted the study based on a fixed medium-sized screen. Nevertheless, the value of N (in this thesis set to 25) could be determined by computing the canvas aspect ratio, which would categorise the same hierarchy differently depending on this aspect ratio, rather than on the inner structure of the data.

Moreover, our categorisation is based on rule-based selection, which is currently robust and reliable. However, recent developments in machine learning algorithms, specifically Large Language Models (LLMs), such as GPT-3.5 or GTP-4, have shown remarkable advancements in natural language processing. For example, a study introduced LLM4Vis [196], a ChatGPT-based visualisation recommender. While it focuses on recommending visualisations rather than creating them, it can serve as a starting point for the future. It is noted that currently it is used and tested with tabular data. Another example is Chat2Vis [122], which is another approach created with ChatGPT that creates visualisations on prompts and selects the layout itself. However, in this tool, visualisations are limited to basic ones and work with tabular data. Thus, the field still lacks more intelligent methods for selecting hierarchical graphs.

Indeed, we proposed a formalisation to categorise hierarchical structures as Elongated or Compact, however, some hierarchies do not belong to either category, which we have defined as Unspecified in our current categorisation. Thus, our classification can be further extended as we detected that some hierarchies include characteristics from both Elongated and Compact (Hybrid) and some of them may have more than one compact structure (N-compact). Indeed, we investigated layouts that fit in well with these additional categories. For example, we think that Force layout could be a good option for visualising Hybrid and n-Compact hierarchies, as it gives nodes more freedom on canvas.

3.4 Conclusions

In this chapter, we introduced a categorisation algorithm designed to classify data into either Elongated or Compact categories, with an additional category labelled Undefined if neither applies. We introduced how the shape of a tree grows, incorporating factors such as significant nodes and growth factors, and formally defined the concepts of Elongated and Compact tendencies. We analysed the threshold values used in this categorisation to justify our selections. We presented examples of various hierarchical structures to illustrate these concepts with different layouts. Moreover, we decided that Circle packing and Tree layout are the best fit for Elongated hierarchies, while Radial and Force layouts are for Compact hierarchies. In the next chapter, we will validate this selection by conducting an evaluation with users. Additionally, we proposed some suggestions for implementing this algorithm as a rule-based automatic selection, which will also be verified according to the results. Finally, in this chapter, we discussed how to visualise multivariate aspects of these hierarchical layouts, and also this will be validated in the same evaluation.

Chapter 4

DViL - A Platform for Hate Speech Visualisation

In this chapter, we introduce the DViL (Data Visualisation in Linguistics) platform, a web-based framework designed for analysing the hate speech contained in the comments of online newspapers, which can be adapted to any data domain. We use this case study to demonstrate the platform's capabilities. First, we outline the primary problems and hypotheses in linguistics aimed at studying hate speech in online newspapers followed by a description of the data linguists developed by annotating features. Next, we discuss the chosen layouts for visualising hierarchical hate speech data and also demonstrate how we selected approaches for multivariate data visualisation, based on formalisations presented in Chapter 3. Afterwards, we describe our proposal of a novel platform (DViL) to visualise the news articles, their comments, and annotated features as an example of hierarchical multivariate data. Finally, to test our categorisation of hierarchical data presented in Chapter 3, we conducted an evaluation with this case study data and the DViL platform, the results of which are also presented here.

4.1 The Context of the Hate Speech Study

The primary objective of linguistics in this context is to analyse messages to detect and classify instances of hate speech, with the ultimate goal of developing a Machine Learning model capable of automatically annotating new datasets. However, achieving this requires a high-quality, well-annotated dataset to effectively train the model and ensure reliable performance in identifying hate speech. The annotation of toxicity is a difficult task due to its inherent subjectivity [153], [45], [60].

Therefore, this subjectivity must be objectified as much as possible to carry out reliable annotation. Deciding whether the content of a message (comment, tweet, etc.) is toxic involves considering various factors, including both the content of the discourse and the extra-linguistic context. The discursive content refers to the linguistic elements, distinguishing between the information conveyed (what is being said) and the manner of expression (e.g., improper language, offensive language, rude vocabulary, belittling language, irony, sarcasm, and mockery). However, the extra-linguistic context involves real-world knowledge such as political, economic, social, and cultural events occurring simultaneously with the publication of the article and comments, which helps interpret them appropriately.

Therefore, to analyse hate speech data, the CLIC [1] Research Group constructed a corpus of annotated messages, that is described in the Section 4.1.1. To reflect the diversity in the expression of toxicity, linguists propose to assign different levels of toxicity, indicating whether the comment is 'not toxic', 'mildly toxic', 'toxic' or 'very toxic'. This classification is based on annotations that consider various features, such as sarcasm, and argumentation among others, which are described in Section 4.1.2. We explain the annotation process and also contextualise it in Section 4.1.3 regarding how visualisation can assist in the annotation and analysis processes.

4.1.1 The NewsCom-TOX Corpus

The data utilised in this application is sourced from a dataset created by the CLIC Research Group, based on online news articles [186, 187]. This data corpus is called the NewsCom-TOX and consists of 4,359 comments posted in response to different articles extracted from Spanish online newspapers (e.g., El Mundo) from August 2017 to May 2019 and annotated with toxicity [186, 187]. The news articles were selected to cover three different topics that are immigration, society and crime, and they were chosen for their likelihood to provoke controversy, aiming to identify comments with opposing opinions and instances of toxic language. The comments were taken in the same order in which they appear in the time thread on the web. Meaning, that some comments are written in response to specific or general aspects of the article in question, while others are written in response to other comments, creating threads of conversations. For example, in Figure 4.1, Comment 1 is written directly under the news article, Comments 2 and 3 are written under Comment 1 as a response, and Comment 4 is a response to Comment 3. Additionally, Table 4.1 shows the distribution of comments per topic and the number of news articles in each topic. News articles had a minimum of 60 comments and a maximum of 360 comments.

[1] Centre de Llenguatge i Computació, Universitat de Barcelona, Spain

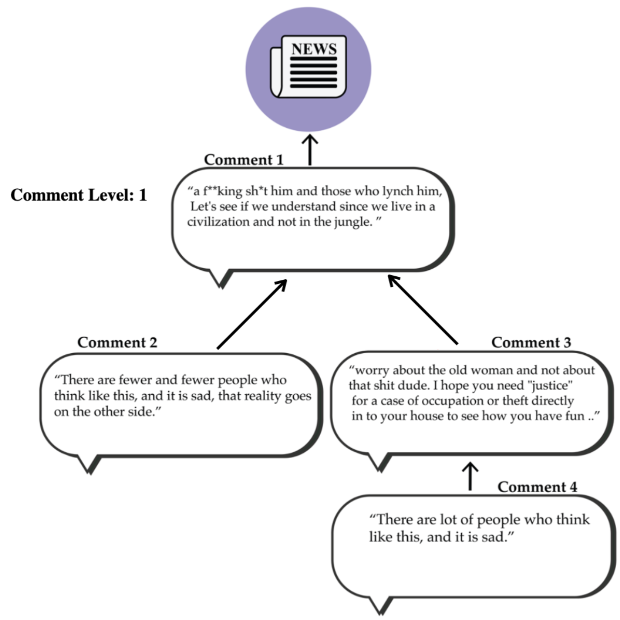


Figure 4.1: Structure of the news article and its comments displaying comment levels.

Table 4.1: Distribution of comments per topic

Area	Comments	No of News Articles (T^k)
Immigration (IM)	1,651	8
Society (SO)	866	5
Crime (CR)	1,842	8
Total	4,359	21

4.1.2 Annotation Tagset

As previously mentioned, analysing hate speech and detecting levels of toxicity is a subjective concept. To provide a reliable classification of toxicity levels, the CLIC Research Group developed an annotation tagset that captures various linguistic features, such as sarcasm, target person, insult, constructiveness, and argumentation, intending to state the level of toxicity as a combination and occurrence of these features. The goal is to standardise the process and minimise disagreement among annotators in identifying toxicity and, consequently, hate speech. These features are binary and will allow for the discrimination of the level of toxicity of the comments. Furthermore, some of these features can be correlated, for instance, argumentation and constructiveness, insult and improper language, and these correlations are also useful when assigning the level of toxicity. Therefore, the proposition is that the combination of these features enables a more objective determination of the level of toxicity. The tagset used for the annotation of comments with toxicity is as follows:

- <**constructiveness**>: a comment is constructive when it is respectful and polite (regardless of whether it is in favour or against the content of the article or another comment) , when it intends to create an enriching and useful dialogue when it contributes with new knowledge, ideas and proposals, and offers new perspectives and insights to approach the subject.
- <**argumentation**>: indicates that the comment gives arguments or reasoned explanations or grounds opinions with pieces of evidence.
- <**sarcasm**>: a comment is sarcastic when the content is ironic -that is, when the writers use words that mean the opposite of what they really want to say- and when it is accompanied by harsh, sharp, and negative criticism and made in bad faith.
- <**mockery**>: indicates that the comment ridicules, mocks or humiliates a person or group.
- <**improper language**>: indicates that the comment contains language not considered to be proper or that is vulgar and impolite which includes rude words.
- <**intolerance**>: indicates that the comment expresses intransigence and non-acceptance of difference, both in general terms (traditions, customs, religious beliefs) and at a personal level (skin colour, sexual orientation, sex, etc.), in oppressed groups.
- <**insult**>: indicates that the comment contains one or more insults or slurs with the intention to offend a person or group.
- <**aggressiveness**>: indicates that the comment expresses violence or a desire to exercise it consciously or unconsciously, without the need to include sarcasm, mockery or insults.
- <**stereotype**>: indicates that the comment contains beliefs or ideas attributed in a generalised way to a group, which characterise the individuals of this group in an undifferentiated and simplistic way based on the magnification of a characteristic.

- <**target person**>: indicates that the comment is directed to a specific person (a politician, an artist, among others).
- <**target group**>: indicates that the comment is directed to a specific group of people (immigrants, the LGBTQ+ community, among others).
- <**negative stance**>: indicates that the comment is written against the previous comment.
- <**positive stance**>: indicates that the comment is written in favour of the previous comment.
- <**toxicity**>: a comment is toxic when it attacks, denigrates or disqualifies a person or group on the basis of certain characteristics such as race, ethnicity, nationality, religion, gender and sexual orientation, among others. This attack can be expressed in different ways -directly (through insult, mockery and inappropriate humour) or indirectly (for instance through sarcasm)- and at different levels of intensity, that is at different levels of toxicity (the most aggressive being those comments that incite hate or even physical violence).

It should be noted that all these tags are either binary (value= yes/no), except for the toxicity tag, which has four values (<0= non-toxic>; <1= mildly toxic>; <2= toxic> and <3: very toxic>). The level of toxicity is determined by the presence and combination of the features presented above (see Figure 4.2). In fact, these features are different ways or mechanisms to express the toxicity and, therefore, they also help to define what is meant by toxicity. The more negative features appear in the comment, the higher the level of toxicity. For instance, annotators tag as 'mildly toxic' comments in which only one feature appears, the most frequent being <sarcasm>, <mockery> and <improper language>, whereas in comments tagged as <very toxic> the combination of features is higher than two, an especially frequent combination is <improper language>, <mockery> and <insult>. This annotation allows for the establishment of fine-grained criteria for analysing and better defining what can be considered a comment with toxic language or hate speech.

Apart from annotating to detect the toxicity of comments, additional contextual information is also collected. This information is invaluable for annotators, aiding in the accurate interpretation and understanding of the message content while also facilitating the exploration of conversation threads between different commentators. It can be helpful to see if commentators contribute more than once or not. Thus, the following is also annotated with:

- <**comment ID**>: An order number that indicates the chronological order in which comments were posted.

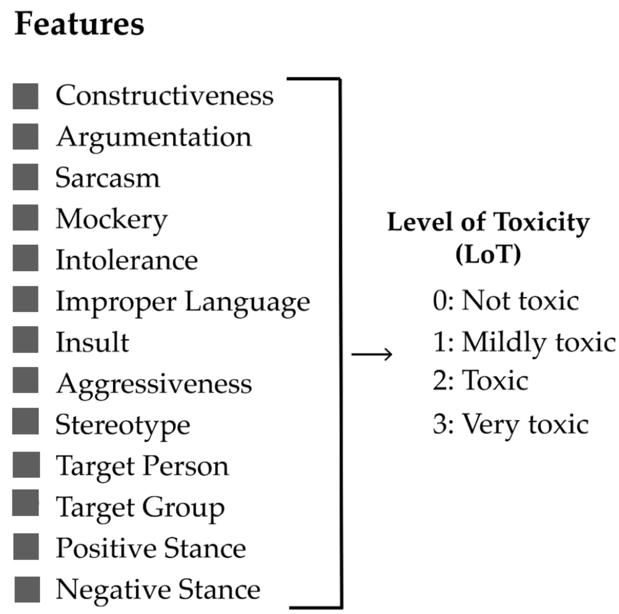


Figure 4.2: Selecting a Level of Toxicity from the combination of features.

- <**user ID**>: Name or alias of the person who wrote the comment; in some cases, these are users who have had to pre-register and may appear in more than one news article.

- <**date**>: Date the comment was posted.

- <**time**>: Time the comment was posted.

- <**thread**>: A thread is a sequence of comments connected by replies. The annotated thread number indicates the position to which each comment belongs. See in Figure 4.3 the **thread 1** consists **Comment 1** and other comments that are written as a response to it.

- <**comment level**>: It is the number that indicates whether this is a primary (1) or secondary (2) comment. Primary comments are written directly under the news article, while secondary comments are written under other comments. Secondary comments will always be marked as 2, regardless of the degree of nesting within a thread. See in Figure 4.3 example of comment **levels 1 and 2**.

- <**depth**>: It is the number that indicates the degree of nesting within a thread. See the illustration of different depths in Figure 4.3.

Thus, the NewsCom-TOX corpus is multi-level annotated with different binary linguistic categories, considering both the information conveyed in each comment and the entire discourse thread in which the comment occurs.

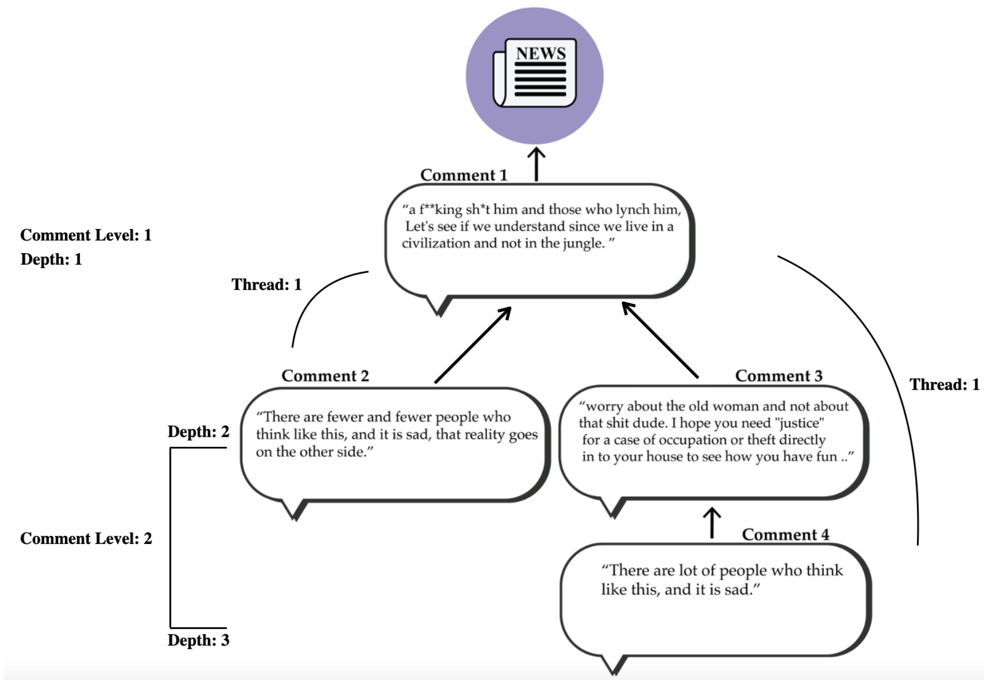


Figure 4.3: Structure of the news article and its comments.

4.1.3 Annotation Process

After establishing the annotation tagset as defined in the previous section, in the following we explore the process the annotators perform in order to annotate the corpus to feed Machine Learning Models and how the visualisations can enhance the whole process. Before starting the annotations, annotators set up the tagset and also define annotation guidelines to ensure a uniform approach to the annotation process. In the first stage of the annotation process, each comment is annotated by three different annotators working in parallel (see (1) in Figure 4.4).

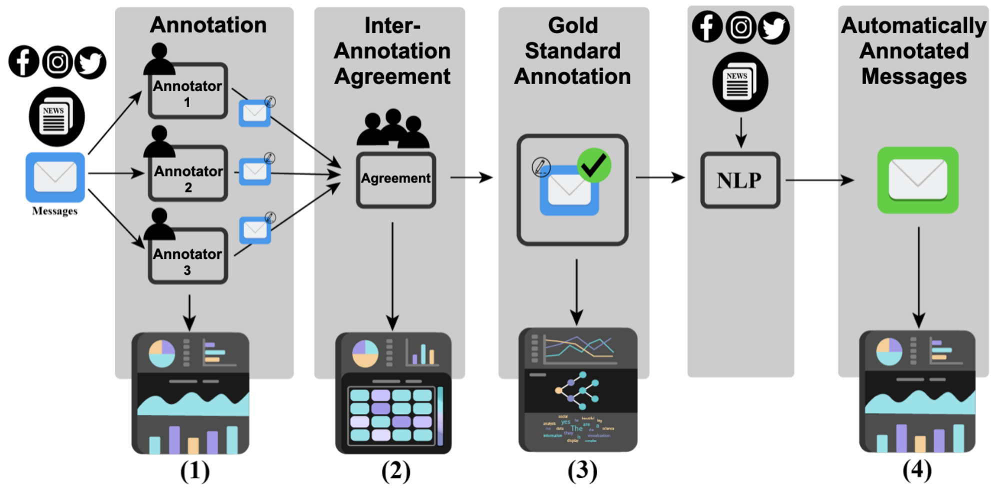


Figure 4.4: Annotation Process and the Role of Visualisations: (1) Individual Annotation, (2) Inter-Annotator Agreement, (3) Gold-Standard Annotation, and (4) Automatic Annotation

Once the annotation of all comments related to a news article was finished, in the second stage, (2) Inter-Annotator Agreement test was conducted to detect the problematic cases and solve the disagreements by consensus in meetings with the three annotators and a lead coordinator. When there was disagreement on the annotation, the final consensus annotation was reached after reviewing each case, discussing and deciding collaboratively. After the inter-annotator agreement stage comes the stage (3) Gold Standard Annotation, which represents the final agreed-upon version of annotations resulting from the inter-annotator agreement process. In the final stage (4) of the annotation process, this data can be used to train Machine Learning algorithms, enabling systems to automatically annotate messages. However, it is important to note that the quality of the automated annotations will depend on the accuracy of the manual annotations.

As depicted in the bottom of Figure 4.4, visualisation plays a crucial role in each of the four stages, adding significant value. For example, in the first stage, visualisations can track the annotation process; in the second stage, it can identify inconsistencies in the annotations; in the third stage, it can assess annotation quality and analyse insights based on the hypotheses of the annotators, which will be discussed later; and finally, in the fourth stage, it can facilitate deeper analysis of insights and monitor the automatic classifications. In this dissertation, we leverage the data from the gold standard annotation to focus on analysing insights and developing a tool to facilitate the analysis of this complex annotated data as the data has a naturally hierarchical structure and with comprehensive annotations (tagged features) it became multivariate.

4.2 Visualising Hierarchical Data

In the previous section, we outlined the challenges faced by linguists in analysing hate speech and described how they created an annotated dataset to achieve their goal. In this section, we will explore how the Visual Mappings of this dataset can be applied to the layouts proposed in Chapter 3. Thus, following the formalisation of hierarchical data introduced in Chapter 3, Section 3.1, we define the NewsCom-TOX as a set of **hierarchical multivariate** structures, denoted as $\mathcal{T} = T^1, T^2, T^3, ..., T^n$, where each T^k represents a directed rooted tree. Specifically, $T^k = \langle Nodes^k, Edges^k \rangle$, with $Nodes^k$ comprising the root node n_0^k and all other nodes N^k, and $Edges^k$ representing the directed edges between these nodes.

In the NewsCOM-TOX annotated dataset, $\mathcal{T}$ are the set of news articles with their corresponding comments. One news and all its associated comments form a rooted tree, T^k, where the news article is the root node, n_0^k. Additionally, some users reply to n_0^k, and others reply to other comments, n_i^k, then edges symbolise all these direct replies. Moreover, we introduced the concept of *subtrees*, which are the hierarchical branches of the main tree rooted at any given node, the *depth*, defined as the distance of each node from the root, and the *width*, which is the maximum number of direct children any node has in the tree.

Additionally, in Chapter 3, we presented a categorisation algorithm for identifying different types of hierarchical structures using the concepts of subtrees, depth and width, and we defined specific properties such as the growth factor, significant nodes, elongated tendencies, compact tendencies, and spines. Next, we also defined two categories of hierarchies: Elongated and Compact. Elongated hierarchies are characterised by narrow structures with nodes distributed along a single path, creating a vertically extended shape. In contrast, Compact hierarchies feature broad structures where nodes are distributed more evenly across the levels, resulting in a more horizontally spread and compact shape.

We also argued that the best layouts for visualising Elongated hierarchies are Tree and Circle Packing, while Radial and Force layouts are more suitable for Compact hierarchies (see Figure 4.5). In this chapter, we use these layouts to visualise the hierarchical NewsCom-TOX dataset introduced earlier. Circle Packing is highlighted for its distinct approach among the implicit layouts, whereas Tree, Radial, and Force layouts represent explicit layouts. Our primary focus is on these explicit layouts to effectively visualise additional multivariate data due to their spatial capacity and the complexity of implicit layouts. Multivariate data visualisation will be explained in the following section.

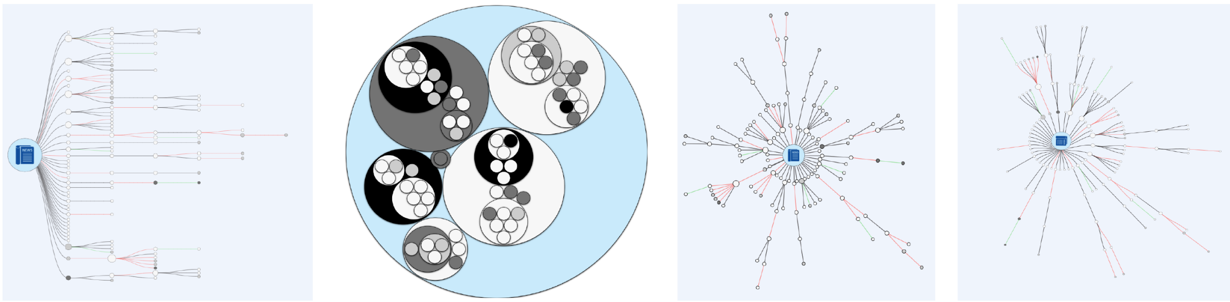


Figure 4.5: Four hierarchical layouts considered in this PhD: (a) Tree Layout, (b) Circle Packing, (c) Force Layout, and (d) Radial Layout.

4.3 Visualising Multivariate Data

Following the formalisation of multivariate data introduced in Chapter 2, Section 3.2.1 (equations about total information associated with nodes and edges 3.6, and labelling features in nodes and edges 3.7), we provide an example of how the formalisation is carried out in the NewsCOM-TOX annotated dataset. The NewsCom-TOX dataset that we presented above, is the set of features associated with the tree nodes, $\mathcal{F}_{Nodes}$, which is a set of categorical features that define the spectrum of the speech related to the comments, such as constructiveness, argumentation, sarcasm, mockery, insult, improper language, intolerance, aggressiveness, target person, target group, stereotype, toxicity, and the level of toxicity. These features correspond to $f_1, f_2, \ldots, f_{13}$, respectively. Some of these features are **nominal** features representing two values, such as f_1 (a message is constructive or non-constructive), and some others include **ordinal** features, such as f_{13} that represents the four levels of toxicity - not toxic, mildly toxic, toxic and very toxic -.

Additionally, the feature relative to the edges of the tree is related to the stance of a comment in relation to the previous one, and thus, $\mathcal{F}_{Edges} = \{f_{14}\}$, i.e., f_{14} represents the stance, that is also a **nominal** variable representing three values - the stance of a message can be positive if it reinforces the opinion of the previous message, negative, if it is against, and neutral, otherwise -.

As introduced previously, The NewsCom-TOX consists datasets that are each an online news article and a possible reply, $data_{n_1^k}$ representing the multivariate data in node n_1^k (see Figure 4.6), the labelling function of the reply produces the following result: $\mathcal{L}_{nodes}(data_{n_1^k}) =$ `<not constructive, argumentative, not sarcastic, not mockery, not intolerant, improper language, insult, not aggressive, target person, no target group, no stereotype, toxic, mildly toxic>`, and $\mathcal{L}_{edges}(data_{e_{0,1}^k}) = \emptyset$.

And, for example, when a node n_2^k supports the opinion of its father n_1^k, then the label of the edge between them is defined by: $\mathcal{L}_{edges}(data_{e^k_{2,1}}) =$ `positive stance`.

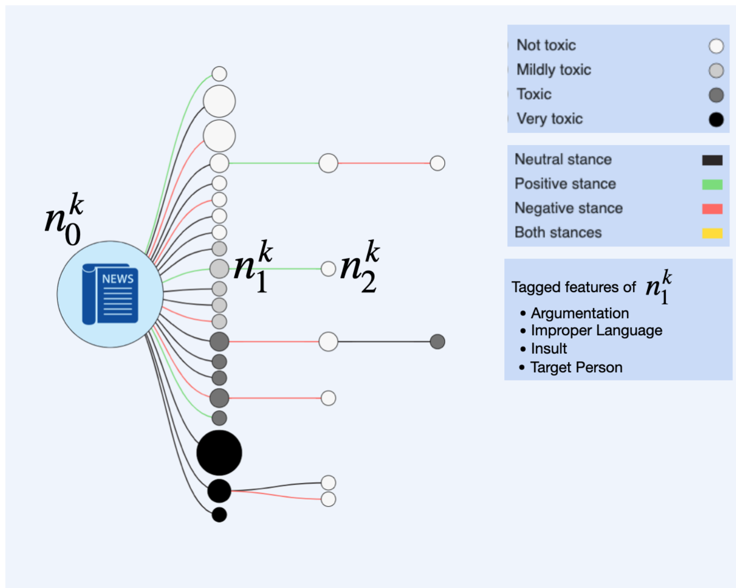


Figure 4.6: Example of a Tree layout showing multivariate nodes.

Additionally, to reiterate, we group related features in $NumC$ clusters depending on their semantics, $\mathcal{C} = \{c_1, ..., c_{NumC}\}$, where $c_i = \{f_1^i, ..., f_{numF^i}^i\}$, being $\forall f_s^i \in \mathcal{F}, 1 \leq s \leq numF^i$. For example, in our case study, the cluster c_1, includes four levels of toxicity c_1={`0 (non-toxic), 1 (mildly-toxic), 2(toxic) and 3 (very toxic)`}, c_2 three types of stances; c_2={`neutral stance, positive stance, negative stance`}, c_3 refer to the targets the comment focuses on; c_3={`target person, target group, stereotype`}, and c_4 includes nominal features; c_4={`constructiveness, argumentation, sarcasm, mockery, intolerance, improper language, insult, aggressiveness`}.

To achieve the best visualisation for each multivariate feature, we utilised our formalisation to cluster these features according to their characteristics. In the following, we describe the Visual Mapping of each cluster.

For our case study, *Cluster 1*, c_1 includes an **ordinal** feature, level of toxicity, c_1={`0 (non-toxic), 1 (mildly-toxic), 2(toxic) and 3 (very toxic)`}. As this is an **ordinal** feature and each node is tagged with one of the values the best way to visualise this feature is as hue colours on the nodes. We used a colour range from white to black to represent non-toxic to very toxic (see Figure 4.7). Moreover, the level of toxicity is the most important feature in our case study. Therefore, it should be prominently visualised in the global view of hierarchies to provide immediate information at first glance. For example, in Figure 4.8, part a) shows that the majority of nodes are toxic, while in part b), it appears that half of the nodes are non-toxic and the other half have varying degrees of toxicity.

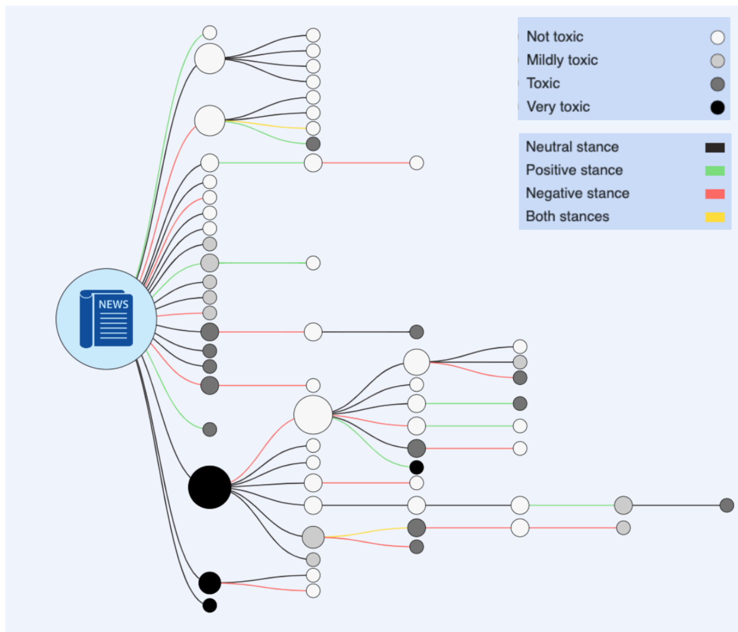
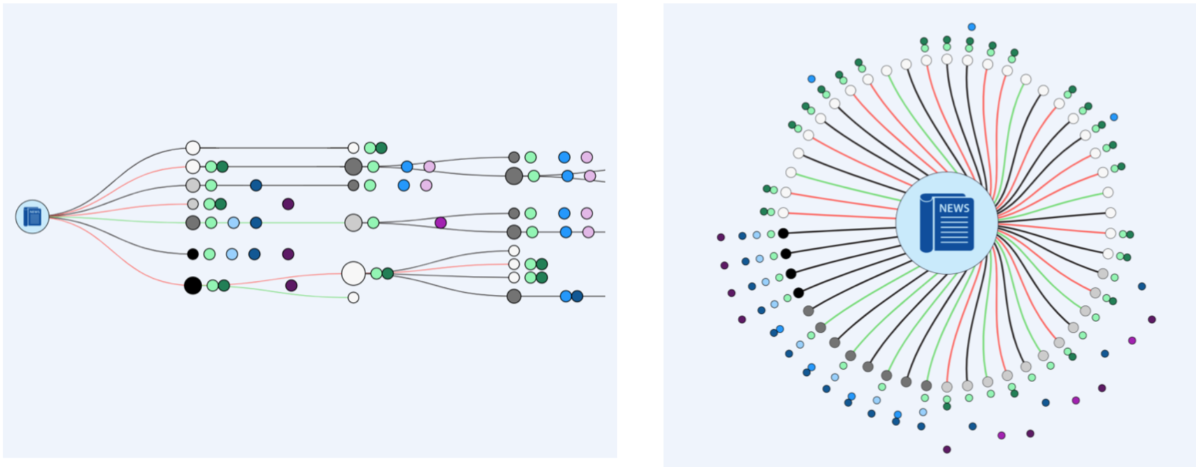


Figure 4.7: Level of Toxicity and Stances mapped on a Tree layout

In the following, we present different **nominal** clusters and the visualisation methods we used for each one. *Cluster 2*, c_2, includes **nominal** features that can be represented with edges, c_2={`neutral stance`, `positive stance`, `negative stance`}. As these features are related to the edges the best option is to directly show them on the layouts as hue colours (see coloured edges in Figure 4.7). Neutral, positive, and negative stances are represented with black, green and red respectively. Moreover, if a node has both positive and negative stances it is shown as orange.

Cluster 3, c_3, represents **nominal** features that have concrete meanings, c_3={`target person`, `target group`, `stereotype`}. Thus, we created an icon for each feature in this cluster and showed them on-demand next to the nodes that have these features (see Figure 4.9c-d).

Finally, *Cluster 4*, c_4, represents eight **nominal** features with abstract meanings, c_4={`constructiveness`, `argumentation`, `sarcasm`, `mockery`, `intolerance`, `improper language`, `insult`, `aggressiveness`}. Therefore, we created two glyphs to visualise these features: i) an *one-by-one* glyph (see Figure 4.9.a), where features are represented by coloured dots that are placed



(a) (b)

Figure 4.8: *One-by-one* glyph shown on (a) Tree layout, and (b) Radial layout

next to the nodes in an ordered row (Figure 4.8.a shows an example), and ii) an *all-in-one* glyph (see Figure 4.9.b), placed on the node, depicted as a pie chart including eight equal pieces with unique hue colours for each feature, displaying the level of toxicity of the nodes on its centre. Both glyphs used green shades for more positive features (i.e., constructiveness), blue for more neutral features (i.e., sarcasm), and magenta for more negative features (i.e., insult). Similarly to *Cluster 3*, these glyphs will be shown on-demand to avoid visual clutter.

a) c_4 b) c_4 c) c_3 d) c_3 e) c_3

Figure 4.9: Glyphs: a) One-by-one, and b) All-in-one, and Target Icons: c) Target Person, d) Target Group, and e) Stereotype

Moreover, we analyse our selected layouts against the glyphs that we will include in them. For example, *one-by-one* glyphs, that is, visualising them next to nodes either linearly, ordered one next to each other, or circularly around the node, which could be very compatible with tree and radial layouts as these layouts are more structured (see 4.8.a and 4.8.b). However, as Force layout distributes its nodes more freely, the linear placement of *one-by-one* glyphs could look confusing and hard to detect which glyph belongs to which node and, then the option to place circularly. Another option for force layout could be visualising glyphs as *all-in-one* on the nodes as shown in 4.10. Note that *all-in-one* glyphs should accommodate all the features in the cluster in less space than in the *one-by-one* case. Additionally, *all-in-one* glyphs could be either placed outside or inside the nodes. In the latter, the space devoted to each feature will be even smaller in size as it depends on the size of the nodes. Thus, *all-in-one* glyph could be more useful on a detailed view (see Figure 4.11, when zoomed in on a part of the layout), while *one-by-one* could be a more helpful method while visualising the global view of the hierarchies without losing the context and causing clutter (see Figure 4.8).

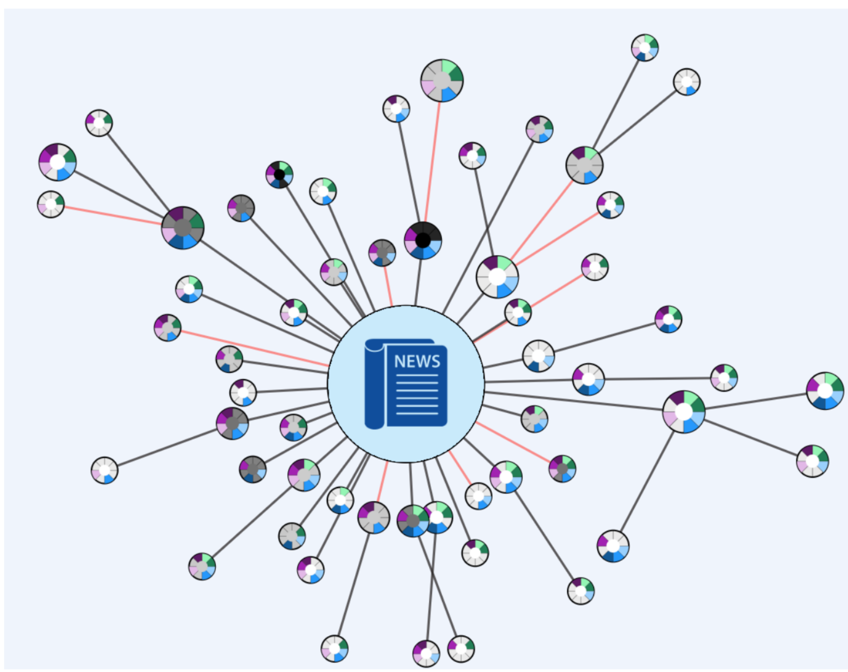


Figure 4.10: *All-in-one* glyph shown on Force layout

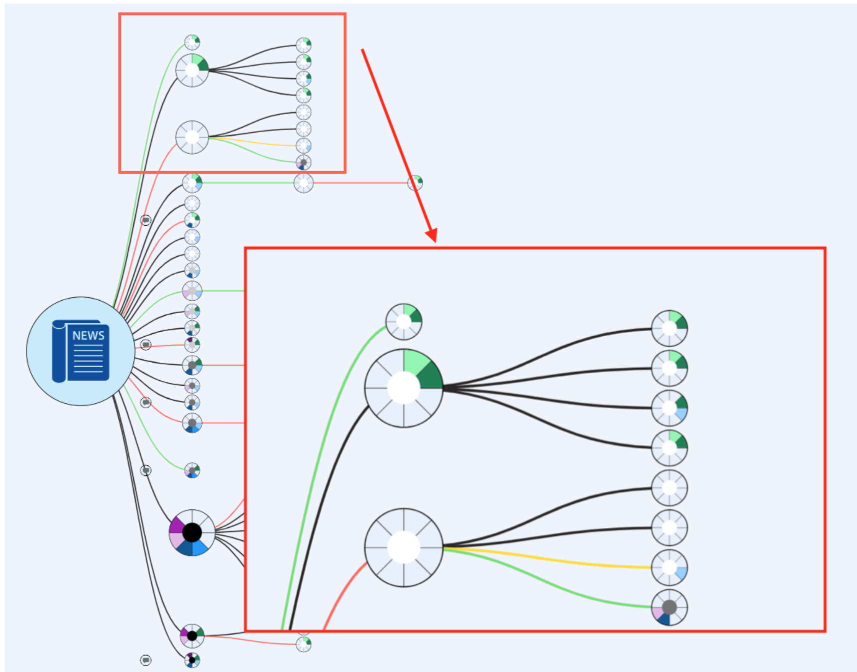


Figure 4.11: *All-in-one* glyph shown in a detailed view.

4.4 Data Visualisation in Linguistics (DViL) Platform

Having described the Visual Mappings in the previous section, this part introduces the new hierarchical multivariate data visualisation platform for linguistics (DViL - Data Visualisation in Linguistics), which provides interactive features and functionalities that will be detailed in the following.

Figure 4.12 displays the user interface of the DViL platform. Figure 4.12 **(A)** is a drop-down menu for datasets, allowing users to select a dataset from the list. The section **(B)** of Figure 4.12 shows the selected dataset's visualisation with the best-fitted visualisation layout according to the structure of its hierarchy. In the DViL platform, we incorporated the categorisation algorithm introduced in Chapter 3. In Figure 4.12, the data (one news article and its comments) is displayed with the tree layout and each comment is represented with a node. Four visualisation layouts, including Tree, Force, Radial, and Circle, can be found at the top of the screen 4.12 **(C)**. Although the platform **automatically selects the layout according to the hierarchy structure**, we also provide users with the freedom to choose their preferred visual mapping through a **user-defined visual mapping**. Figure 4.12 **(D)** displays glyph options we introduced in the previous section to visualise *Cluster 4*; Dots (one-by-one) and Circular (all-in-one), that can be used to visualise abstract tagged features of each node. On the left hand-side, (see part **(E)** in Figure 4.12), there are three sets of filters: (1) Targets (shows target icons next to associated nodes) to visualise *Cluster 3*, (2) Highlight Colour Feature (activates colours of glyphs) and (3) Select node & edge (displays only nodes & edges with selected annotations) to filter all *Clusters*. Figure 4.12 **(F)** illustrates the summary of the statistics (level of toxicity and targets) and buttons for additional charts that will open as pop-ups. One of the additional charts can be seen in Figure 4.12 **(G)**, showing the distribution of features in the whole visualisation. Finally, Figure 4.12 **(H)** shows additional charts displaying the distribution of the features in subtrees. Next, we describe the main components of the platform in detail.

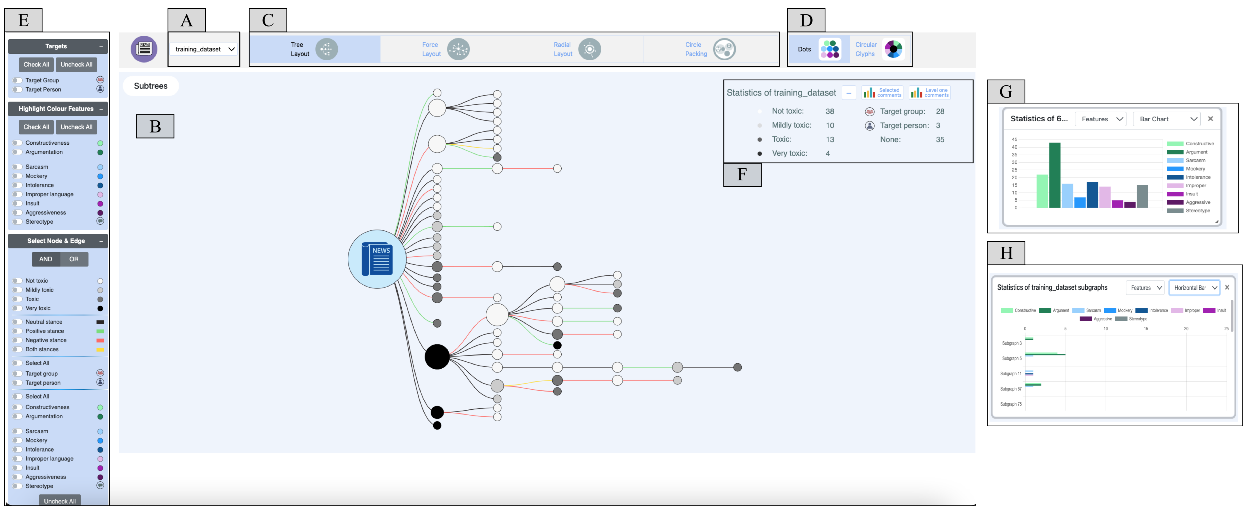


Figure 4.12: Overview of the platform (A) Datasets menu (B) Main visualisation: hate speech annotation of an online news article and its generated comments, (C) Hierarchical layouts, (D) Glyphs, (E) Filters, (F) Summary of statistics and supplementary charts, (G) Supplementary chart illustrating distribution of features in the whole visualisation, and (H) Supplementary chart illustrating distribution of features in subtrees.

Main Visualisation

The main visualisation includes four layouts: Tree, Force, Radial, and Circle. Each of these visualisations represents a single news article at a time, with the root symbolising the news article and the nodes representing the comments made on it, maintaining the hierarchical structure on layouts Tree, Force, and Radial. In the Circle Packing layout, the news article and comments are nested within each other as circles, with the root represented as a large blue circle beneath all other circles. In this layout, the nodes are depicted as packed circles (see Figure 4.5.b in Section 4.2). The node colours indicate toxicity levels, ranging from white (non-toxic) to black (very toxic), visualising the *Cluster 1*. Edges represent the stances of the comments with green being the positive stance, red negative, orange both, and black neutral, visualising *Cluster 2* of multivariate data. The node size reflects the number of child nodes. Nodes can be collapsed or expanded for further interaction in the Tree, Force, and Radial layouts. Additionally, in the Force layout, nodes can be dragged and dropped. Subtrees, which were introduced in the formalisation in Chapter 3, are considered here as well. For example, in Figure 4.12 (B), each node directly connected to the root node represents the beginning of a subtree, with the nodes branching out from it forming smaller trees within the overall structure.

Filters

As explained, the User Interface presents three sets of filters: Targets, Highlight Colour Features, and Select Node& Edge (part (E) in Figure 4.12). Our aim with these filters is to reduce visual clutter and appropriately visualise multivariate data on demand while maximising information and not losing context. Additionally, they allow us to focus our attention on specific levels of toxicity or features as needed.

For example, in Figure 4.13, we are analysing comments that are annotated with Argumentation by selecting it from the Select Node & Edge filter menu (see the activated buttons in the Menu of filters). In the main view, some nodes are greyed out while others are highlighted, indicating those tagged with argumentation. When a feature is chosen from this menu, statistics in the top-right corner are updated accordingly. Furthermore, we delved deeper into features present in argumentative nodes by activating the Dots (one-by-one) glyph option from the Highlight Colour Feature filter menu. This displays all features next to the nodes side by side. Additionally, a complementary chart is opened to visualise the distribution of features in argumentative comments, which is updated based on selections from the Select Node & Edge menu.

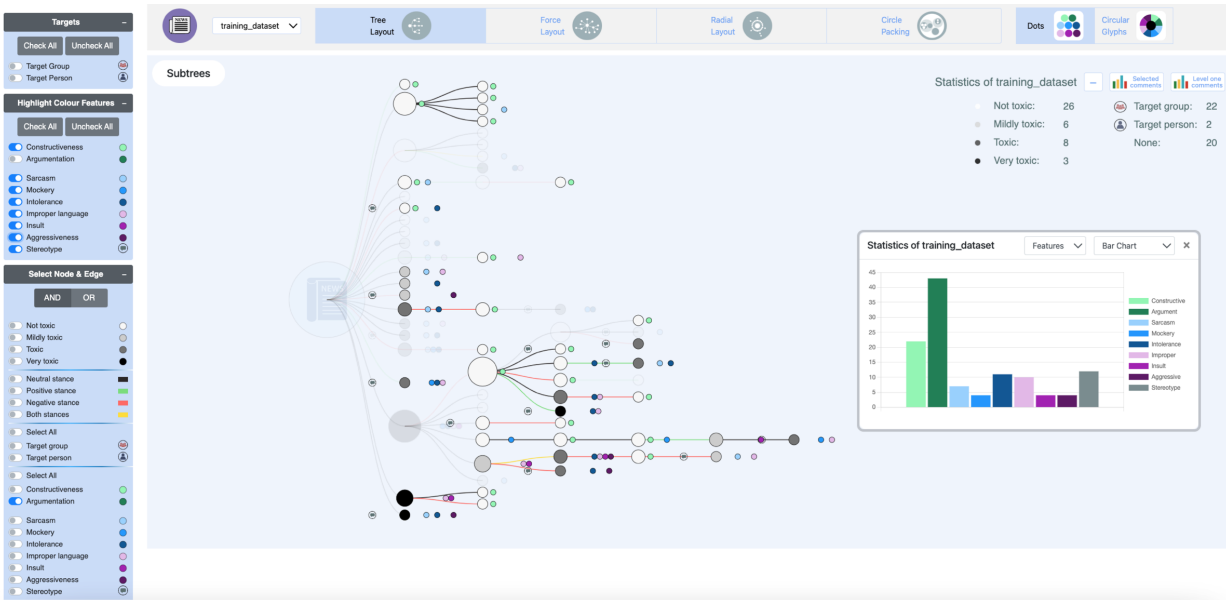


Figure 4.13: Use case of Filters and complementary chart

It should be noted that we offer our users two types of filtering for nodes and edges. One option is the **AND** option, where if two features are selected, such as Argumentation and Improper Language, only the comments that have both features will be highlighted. In contrast, with the **OR** option selected, comments that have either Argumentation or Improper Language will be highlighted. To compare the differences, refer to Figure 4.14 (a) for the AND option, and (b) for the OR option.

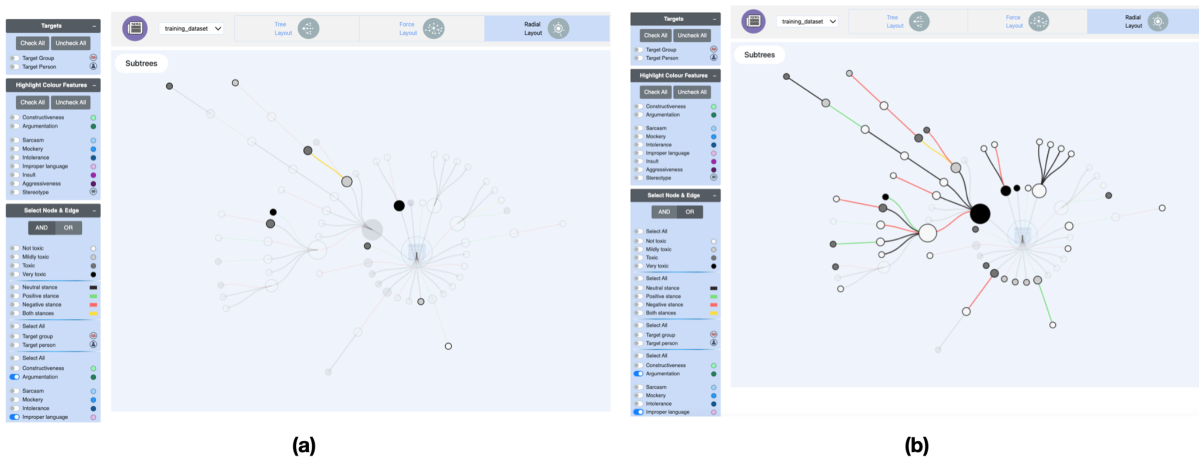


Figure 4.14: (a) Filtering Argumentation and Improper Language comments with AND option, which displays comments that are tagged with both features and (b) Filtering with OR option, which displays comments that are tagged with either feature.

Similarly, icons representing features are also shown on demand to avoid overcrowding. In Figure 4.15, we present an example showing our icons for Target Person, Target Group, and Stereotype.

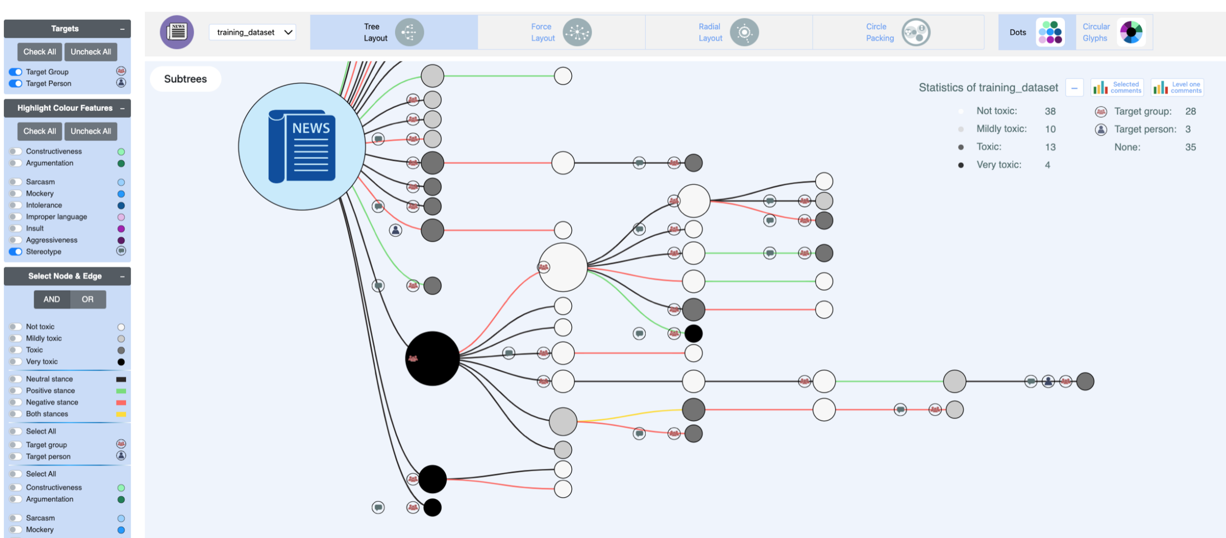


Figure 4.15: Example of Icons (Target Person, Target Group, and Stereotype) displayed on the tree layout.

Complementary Charts

The DViL platform includes complementary charts alongside the main visualisation to display the statistics of all features. This helps users view the data in another format. Like glyphs, these charts are shown on demand. Users can review the statistics of the features by clicking on the "*Selected Comments*" button or the individual subtrees using the "*Level One Comments*" button in the statistics summary section in the upper left corner (see Figure 4.16).

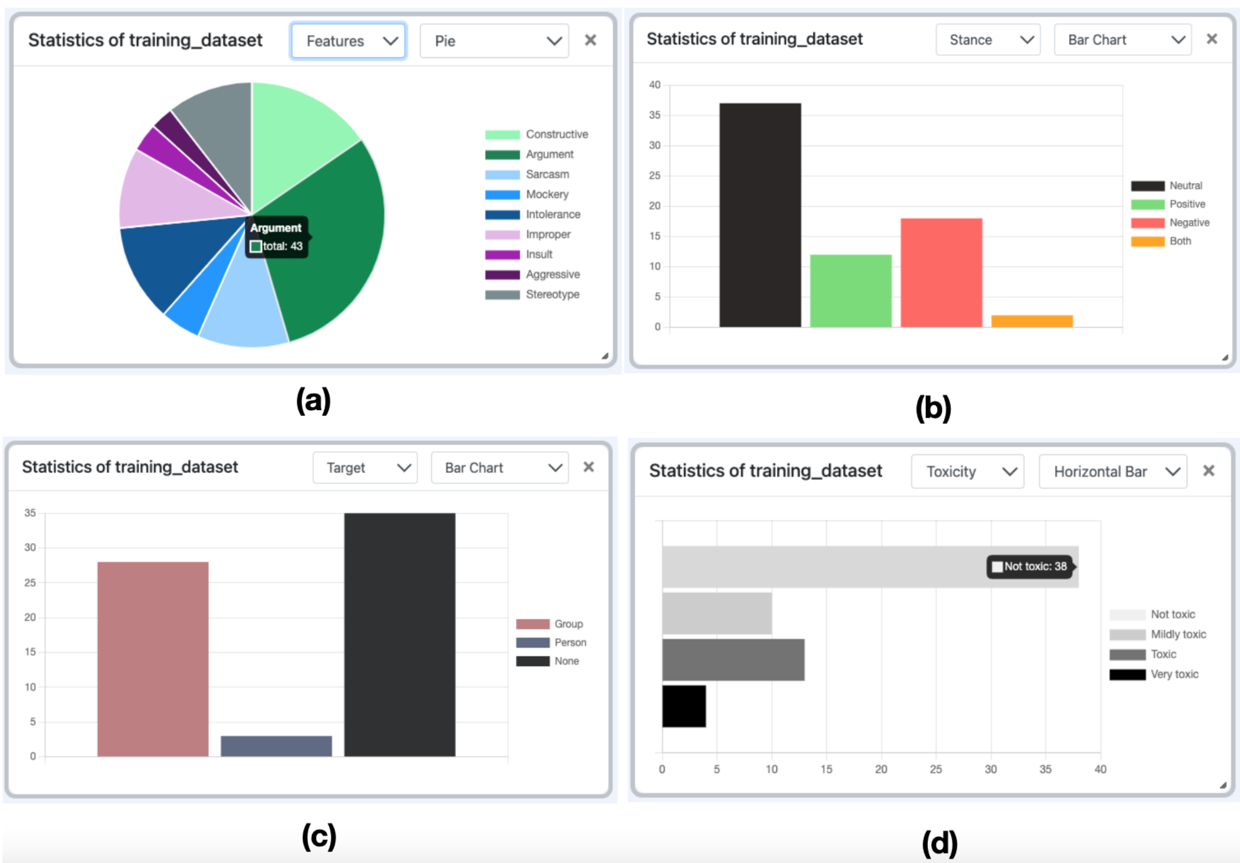


Figure 4.16: Complementary charts showing statistics of (a) Features, (b) Stances, (c) Targets, and (Level of Toxicity)

In these charts, we can visualise Features 4.16.a, Stances 4.16.b, Targets 4.16.c, and Level of Toxicity 4.16.d. Selections can be made using the first dropdown to choose the type of data to visualise, and the second dropdown to select the chart type: Bar Chart, Horizontal Bar Chart, or Pie Chart. By default, the Bar Chart is shown, as it provides the easiest data reading in most cases. It should be noted that hovering over these graphs reveals the values. Additionally, these charts are updated dynamically when a selection is made in the Select Node & Edge menu.

Subtrees in Complementary Charts

As mentioned, each node directly connected to the root, along with its children, represents a subtree. We have the option to view the statistical information of these subtrees individually. Currently, each subtree can be highlighted by clicking on its name. Figure 4.17 demonstrates this by selecting the first subtree, which highlights it in the main visualisation. Note that in this complementary chart, the subtree is highlighted while preserving the contextual information of the entire tree (i.e., a focus+context approach), which maintains the properties of Elongation and Compactness unchanged. However, if the selected subtree were visualised independently, these properties could vary, possibly requiring layout changes. This concept is explored further in the following section of the manuscript, where users perform visualisation tasks using the VisChatbot.

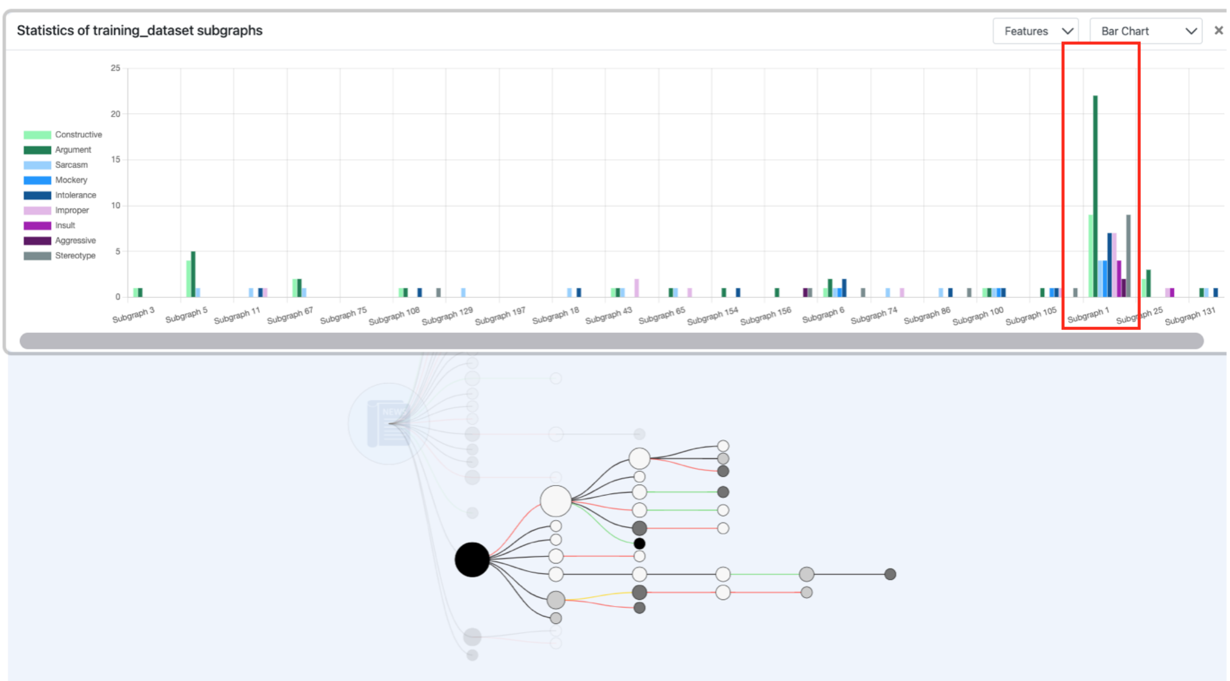


Figure 4.17: Subtrees in Complementary Charts

Tooltip

A tooltip displays tagged features and contextual information collected by the annotators, such as comment ID, comment level, comment depth, and the actual comment from the user (see Figure 4.18). This allows annotators to see the comment details. The features that are not tagged are greyed out, while the tagged features are highlighted in the tooltip. Moreover, when a node is collapsed and its children are collapsed with it, additional information, such as the number of direct children and total children, appears. This shows the number of comments in a collapsed node and its children, also known as the subtree, as well as the total number of each level of toxicity contained in that parent node. To maintain our goal of avoiding clutter, the tooltip appears only when hovering over a node.

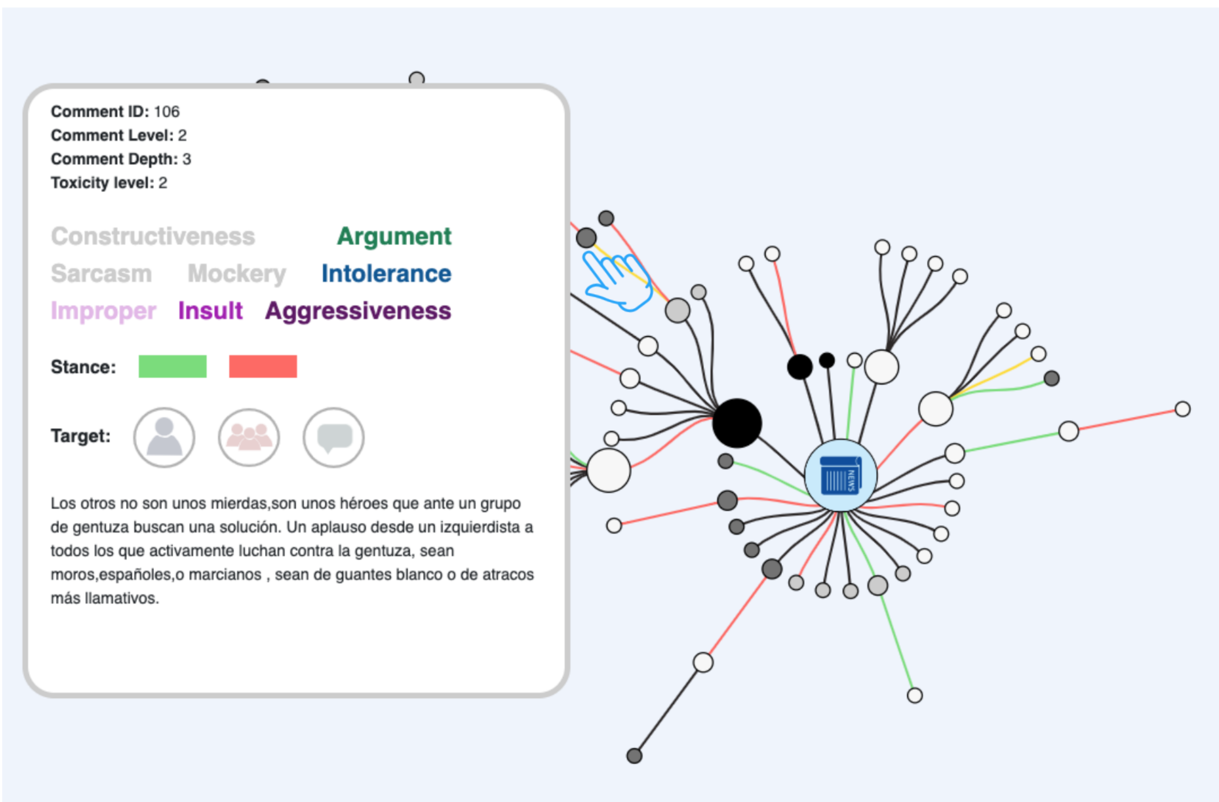


Figure 4.18: Tooltip

4.5 Evaluation

This section presents a user evaluation of the DViL platform focusing on the testing of the automatic layout/glyphs selection based on the categorisation introduced in Chapter 3. Our goal is to achieve a visualisation of hierarchical structures that allows for a clear analysis of both parent-child relationships and the distribution of features, without overwhelming the user's perception or causing clutter.

In fact, we formalised hierarchical structures with a categorisation in *Elongated (narrow)* and *Compact (broad)* structures and argued for the adequacy of a visualisation method - Tree, Circle Packing, Force and Radial (see Figures 4.19 and 4.20) - depending on defined attributes, such as the growing factor and the number of direct children of a node, which can be applied to any hierarchical dataset. We also formalised features of multivariate data and, consequently, integrated their visualisation in a hierarchical structure. Based on this formalisation, we introduced two types of glyphs: i) the *one-by-one*, where features are depicted by coloured dots placed one next to each other, and ii) the *all-in-one*, where a single pie chart represents all the features. Thus, we argued that a *one-by-one* glyph is more informative than a *all-in-one* glyph for depicting multivariate data in overviews of hierarchical structures while ensuring comprehension and avoiding complication. Therefore, in our test, we aim to validate the following hypotheses:

- ***H1:*** *When the hierarchy is categorised as Elongated (EC), the most informative methods to visualise it are Tree layout and Circle Packing* (see (a) and (b) in Figure 4.19).

- ***H2:*** *When the hierarchy is categorised as Compact (CC), the most informative methods to visualise it are Force layout and Radial layout* (see (a) and (b) in Figure 4.20).

  H3: *With a large number of features, the most informative glyph to embed them in a hierarchical layout is the* one-by-one *glyph instead of an* all-in-one *glyph* (see (a) and (b) in Figure 4.21).

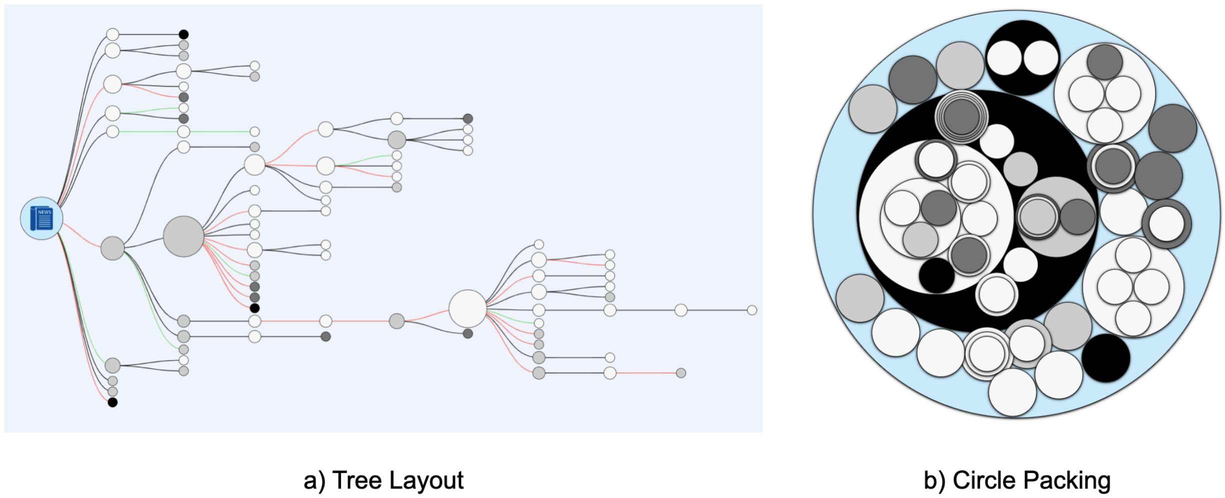


Figure 4.19: Elongated Hierarchy with a) Tree Layout and b) Circle Packing

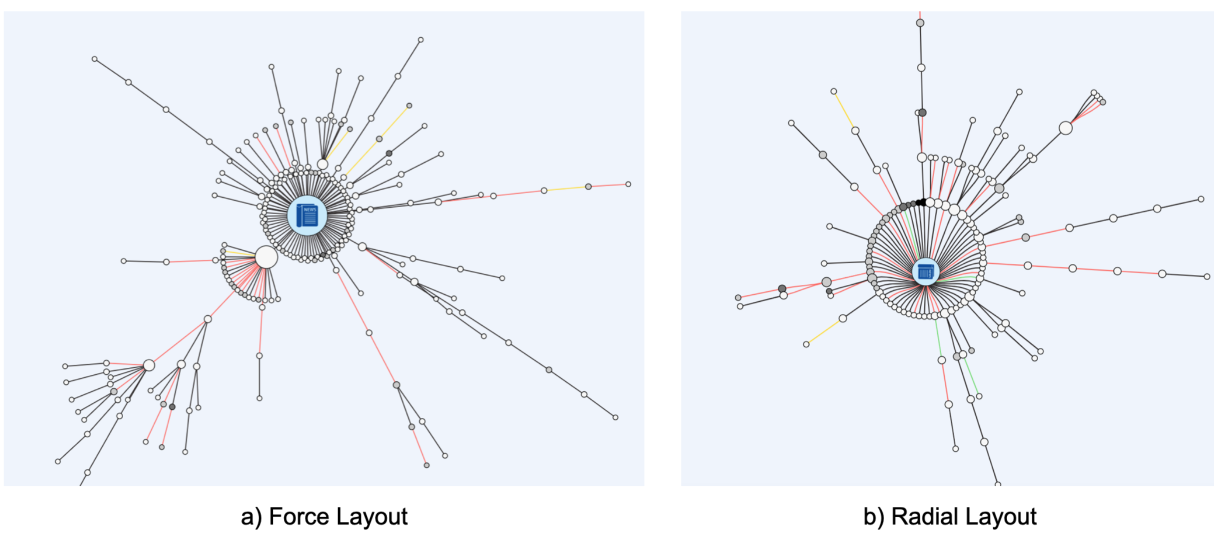


Figure 4.20: Compact Hierarchy with a) Force Layout and b) Radial Layout

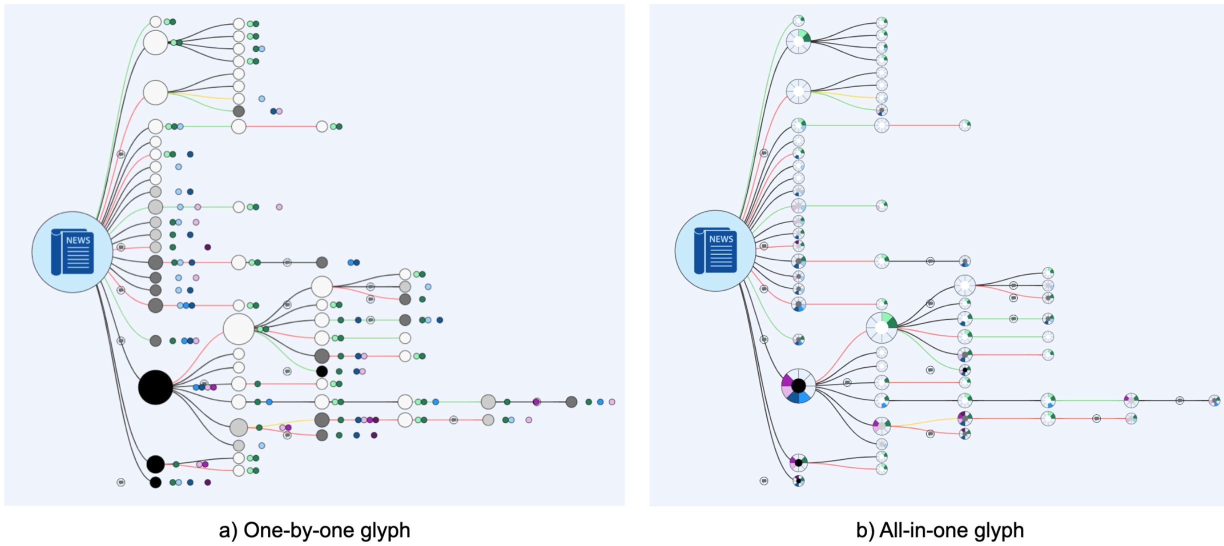


Figure 4.21: Example of a) One-by-one Glyph and b) All-in-one Glyph embedded in hierarchical visualisation with large number of features.

We conducted a user evaluation to study our hypotheses. Thus, the purpose of the evaluation was twofold: 1) to validate the best-fitted visualisation layouts according to the proposed categories of hierarchies, Elongated and Compact, and 2) to determine which glyph (*one-by-one*, *all-in-one*) is more useful with layouts while visualising data globally without causing clutter.

4.5.1 Methodology

We recruited 35 participants of which two of them were lecturers while the rest were students from the Faculty of Philology and Communication at the University of Barcelona; 60% of the participants were female, 80% were aged between 20-30 and 17% of the participants had experiences in message annotation and data visualisation but in general, participants had no experience in these fields. The study was a within-subject where participants performed several VisTasks. It was a moderated test, conducted in a classroom, with an average duration of one and half hours. It should be noted that, prior to the evaluation, the protocol we defined for the experiment was approved by the Bioethics Committee of the University of Barcelona. Before starting, participants were presented with a standard consent form, which they read and agreed to. The form informed them that the test would be anonymous, their data would be securely stored and only accessible by researchers involved in this study, and that they were free to withdraw at any time.

At the beginning of the evaluation session, we explained to students our research, the structure of conversation threads, and annotations (tagged features), and then we briefly explained how they were mapped onto our visualisation layouts. We aimed to facilitate the understanding of the tagged attributes by explaining this in more depth and to make sure that users focus on the visualisations rather than trying to understand how annotations worked. Afterwards, we gave them a couple of minutes to engage with the visualisations before we started the test to ensure that our users understood the tool. Also, before each VisTask, we gave users tips regarding how to use our visualisation tool (e.g. showing glyphs) that could help them solve the VisTasks, again, to make sure that users focus on the visualisations rather than trying to understand the tool's functioning. There were a total of 5 VisTasks (see Tables 4.2 and 4.3, where EC and CC refer to Elongated Categories and Compact Categories, respectively), to make sure that there was no learning effect in performing the first 4 VisTasks we prepared two versions, half of the users did Test A with the VisTask order T1-EC, T2-EC, T3-CC, T4-CC, T5, and the other half did Test B with the VisTask order T3-CC, T4-CC, T1-EC, T2-EC, T5. T5 was consistently placed last for all users to isolate the only glyph-related VisTask. This placement ensured it was not affected by the learning effect, as it was the first time users encountered it.

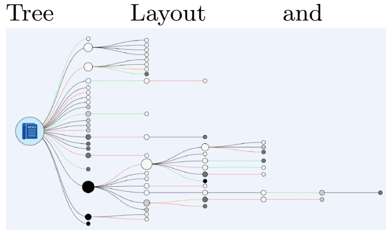
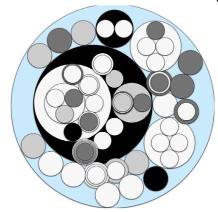
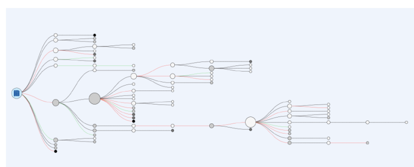
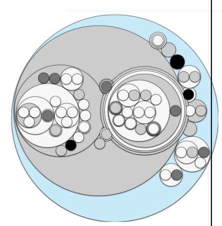
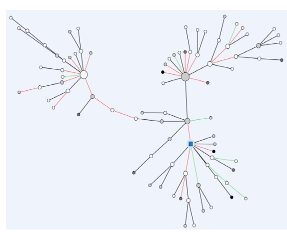
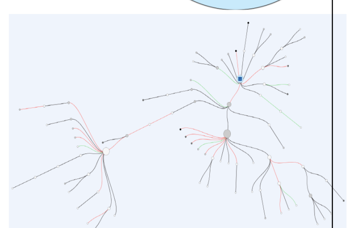
4.5.2 Research Hypotheses and Associated VisTasks

We aimed to analyse our hypotheses with VisTasks that are defined from linguists' research questions. Each VisTask is assigned a chosen dataset according to the Elongated and Compact categories.

VisTasks 1 and 2 were designed to explore ***Hypothesis 1:*** *"When the hierarchy has Elongated Category the most informative layouts to visualise it are Tree layout and Circle Packing."* To test our hypothesis, in VisTasks 1 and 2, we selected datasets that are Elongated thus we will refer to these VisTasks as T1-EC and T2-EC. In T1-EC we asked users to only engage with Circle Packing and Tree layout, and in T2-EC we asked users to play with all layouts. In this way, in T1-EC we could compare the two layouts we selected for the Elongated category to see if they are equally useful for the users to perform the VisTasks. Moreover, with T2-EC we could compare our selections (Tree and Circle) for the Elongated category with users' selections as we expect them to have higher scores than the other layouts (Force and Radial) in this category.

VisTask 3 and VisTask 4 were designed to explore ***Hypothesis 2:*** *"When the hierarchy has Compact Category the most informative layouts to visualise it are Force layout and Radial layout".* Thus, we assigned datasets with the Compact category to these VisTasks and we will refer to them as T3-CC and T4-CC. Similarly, in T3-CC we asked users to only engage with selected layouts,

Table 4.2: Evaluation: Research Hypothesis H1 with associated VisTasks, layouts, datasets, and data gathered.

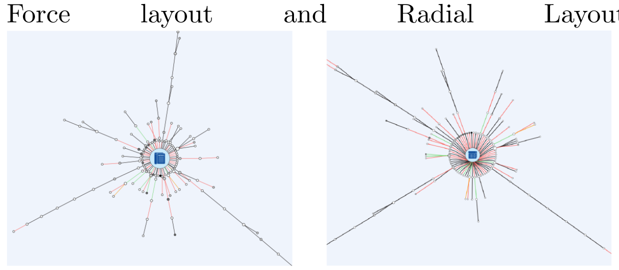
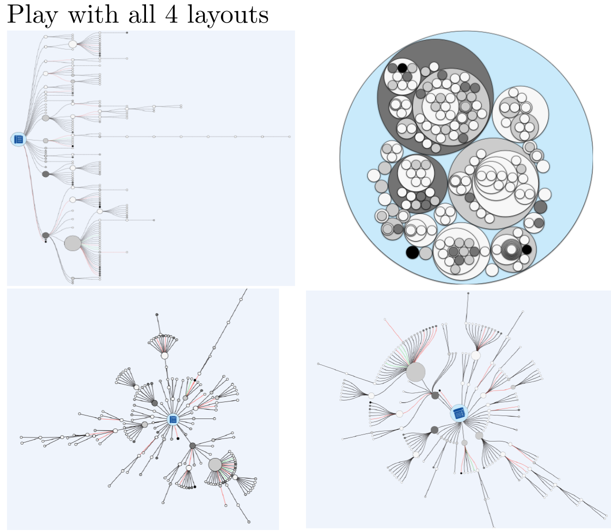
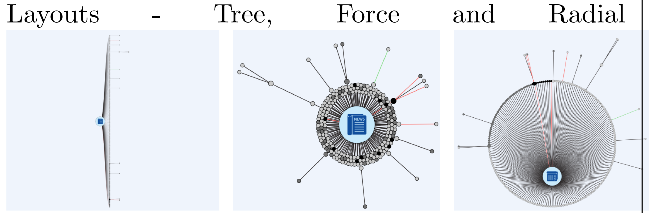
Research Hypotheses	**VisTask**	**Visualisation Layout**	**Dataset**	**Data Gathered**
H1 Elongated Category	**T1-EC:** In Comment level 2, in which Level of Toxicity does the combination of Intolerance and Stereotype comments appear more?	Tree Layout and Circle Packing  	Elong. News Article IM_1	Ratings of 2 layouts
	T2-EC: In Comment level 2, in which Level of Toxicity does the combination of Improper Language and Insult comments appear more?	Play with all 4 layouts    	Elong. News Article: SO_1	Ratings of 4 layouts

Force and Radial and in T4-CC we asked users to play with all layouts. Thus, we could compare if Force and Radial are equally sufficient in T3-CC. Furthermore, we wanted to compare if users agreed with our selections as we expected that Force and Radial would score higher than Tree and Circle Packing in T4-CC.

Finally, VisTask 5 was designed to explore ***Hypothesis 3:*** *"When a cluster contains a large number of features, the most informative representation to embed them in a hierarchical layout is an* one-by-one *instead of an* all-in-one *glyph"* . In this VisTask, we asked participants to interact with both glyphs with the layout of their choice and rate considering the global view of the visualisation. We decided to use Compact data with VisTask 5 as we wanted to see if our glyphs were useful even with broad datasets.

In all VisTasks, we asked users *"to rate the layouts they used to perform the VisTasks on a scale of very difficult (1) to very easy (5), N/A if applicable, and to write their comments, if they*

Table 4.3: Evaluation: Research Hypotheses H2 and H3 with associated VisTasks, layouts, datasets, and data gathered.

Research Hypotheses	**VisTask**	**Visualisation Layout**	**Dataset**	**Data Gathered**
H2 Compact Category	**T3-CC:** In Comment level 1, in which Level of Toxicity does the combination of Target Group and Mockery comments appear more	Force layout and Radial Layout 	Comp. News Article: SO_2	Ratings of 2 layouts
	T4-CC: In Comment level 2, in which Level of Toxicity there are more Aggressive comments?	Play with all 4 layouts 	Comp. News Article: IM_2	Ratings of 4 layouts
H3 Glyphs	**T5:** Which Features appear more with Target Group in Level of Toxicity 3 (very toxic)? In this task please use *one-by-one* glyph, *all-in-one* glyph and Target Icons	Layouts - Tree, Force and Radial 	Comp. News Article: CR_1	Ratings of 2 glyphs

have any." It should be noted that in VisTask 5 we asked participants to play only with layouts Tree, Force and Radial as these are the visualisation layouts that support glyphs. Additionally, in VisTask 5, we asked users *"to rate the glyphs they used to perform the VisTask on a scale of not useful (1) to very useful (5) and N/A if applicable."* At the end of all VisTasks we asked about their overall comments to collect some qualitative data. The evaluation VisTasks and questionnaire are available at this link.

4.5.3 Results

We considered our sample size enough for computing statistical significances, as stated by [27]. Thus, we conducted paired t-tests and computed the Effect Size (ES) after rejecting the null hypothesis to measure the magnitude of mean differences (see Cohen's d ES in Tables 4.4, 4.5 and 4.6). Notice that we obtained large effect size in all cases (Cohen's d ES > 0.8), meaning that means are likely very different. In the following, we present our results.

Hypothesis 1

VisTasks T1-EC and T2-EC were designed to study Hypothesis 1 (Elongated structure). Specifically, T1-EC was designed to compare Tree layout and Circle packing. As it can be observed in Figure 4.22 and Table 4.4, Tree layout (4.17 out of 5) received much higher scores than Circle packing (2.77), the standard paired t-test shows a significant difference between their scores ($p < 0.001$). However, this may be as a result of participants' familiarity with Tree layout as hierarchical tree diagrams are being used in a variety of fields such as management planning [8] or diagramming sentences in linguistics [123]. Circle packing has a relatively different design that participants required more explanations about how conversation thread structures are mapped onto its circular design. Moreover, the results of T1-EC and T2-EC, show that Tree layout kept its popularity and has the same mean (4.17) in both VisTasks.

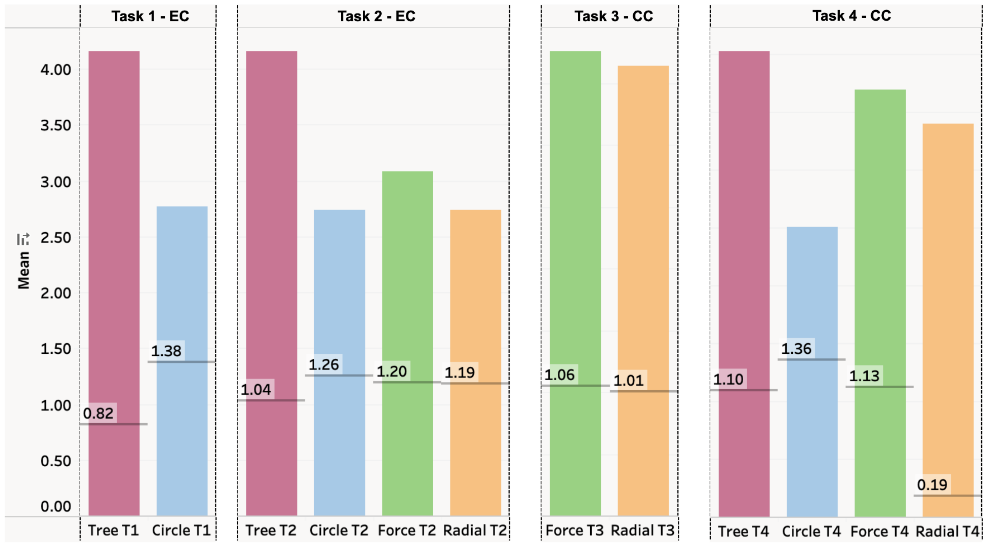


Figure 4.22: Mean values of Layouts from VisTask 1 to 4 with Standard Deviation

Table 4.4: Hypothesis 1, Elongated Categories (paired t-test)

Layouts (Mean, SEM)			**p-value**	**ES**
T1-EC	Tree (4.17, 0.14)	Circle (2.77, 0.23)	< **.001**∗	0.85
T2-EC	Tree (4.17, 0.18)	Force (3.09, 0.20)	< **.001**∗	0.74
	Tree (4.17, 0.18)	Radial (2.74, 0.20)	< **.001**∗	1.00
	Circle (2.74, 0.22)	Force (3.09, 0.20)	0.195	-
	Circle (2.74, 0.22)	Radial (2.74, 0.20)	1.000	-

According to Hypothesis 1, we expected that Tree layout and Circle packing would score higher than Force and Radial layouts in T2-EC, recall that users played with all layouts in this VisTask. When we compared the results of the Tree layout versus the Force and Radial layouts in T2-EC we found that the Tree layout (4.17) scored significantly higher than the Force (3.09) and Radial (2.74) layouts. The standard paired t-tests prove the significant difference between Tree versus Force and Radial layouts as they are both $p < 0.001$. However, when we compared Circle packing (2.74) with the Force and Radial layouts in T2-EC with the standard paired t-test we could not find any significant differences. As we stated previously, this can be because of participants' unfamiliarity with a circular design of threads of comments.

Remark 4.1. *The results indicate that* ***the most preferred layout with Elongated data is the Tree layout****. Thus, H1 is partially supported by our results.*

Hypothesis 2

VisTasks T3-CC and T4-CC were designed to study Hypothesis 2 (compact structure). As it can be seen in Figure 4.22 and Table 4.5, Force and Radial layouts in T3-CC scored relatively similar as expected, 3.77 and 3.65, respectively, and the t-test shows that their scores do not have a significant difference.

Table 4.5: Hypothesis 2, Compact Structures (paired t-test)

Layouts (Mean, SEM)			**p-value**	**ES**
T3-CC	Force (3.77, 0.18)	Radial (3.65, 0.17)	0.607	-
T4-CC	Force (3.69, 0.21)	Tree (4.03, 0.19)	0.195	-
	Force (3.69, 0.21)	Circle (2.51, 0.23)	$<$ **.001***	0.83
	Radial (3.4, 0.21)	Tree (4.03, 0.19)	**0.012***	0.45
	Radial (3.4, 0.21)	Circle (2.51, 0.23)	**0.003***	0.54

According to Hypothesis 2, we expected that the Force and Radial layouts would score higher than the Tree and Circle in T4-CC. When we conducted the standard paired t-test between the Force (3.69) and Radial (3.4) layouts against Circle packing (2.51), both showed that they scored significantly higher than the Circle packing. The results of the t-tests of the Force layout vs Circle packing is $p < 0.001$ and the Radial layout versus Circle packing is ($p = 0.003$). However, neither the Force nor Radial layouts scored higher than the Tree layout. The reason, again, for this can be that the Tree layout is a more common visualisation layout when it is compared to others. Moreover, another reason for this may be because originally conversation threads were formed from top to bottom and in the Tree layout, this is simply changed to the left to right. However, in other layouts the structure changed into circular designs, this caused participants to understand the tree layout more easily, therefore giving it higher scores.

Remark 4.2. *The results indicate that t****he most preferred graph with Compact data is Tree layout.***

Although the Tree layout was the best-scored layout in the previous analysis of Hypothesis 1 (Elongated data) and Hypothesis 2 (compact data), we believed that Force and Radial should score higher with Compact data and Circle packing should score higher with Elongated data. Then, we compared the scores of Force, Radial and Circle Packing layouts between T2-EC and T4-CC (see table 4.6).

Table 4.6: Elongated vs Compact (paired t-test)

Layouts (Mean, SEM)		**p-value**	**ES**
T2-EC	T4-CC		
Force (3.09, 0.20)	Force (3.69, 0.21)	**0.016***	0.43
Radial (2.74, 0.20)	Radial (3.4, 0.21)	**0.019***	0.42
Circle (2.74, 0.22)	Circle (2.51, 0.23)	0.530	-

The results show that Force and Radial layouts scored significantly higher in T4-CC ($p = 0.016$) and ($p = 0.019$), respectively. Even though Force and Radial layouts scored lower than Tree layout in both VisTasks, we can see that with Compact data they scored significantly higher. Therefore, our Hypothesis 2 can be partially supported. Furthermore, Circle packing received a higher score in T2-EC compared to T4-CC which is aligned with our Hypothesis 1. However, the difference is very little and we couldn't find any significant difference.

Remark 4.3. *Even though* ***Tree layout was selected best with Compact data, Force and Radial layouts showed some evidence that they are more appropriate for Compact data rather than Elongated ones****. Thus, this evidence confirms that our second hypothesis can be further explored. Even though it is not significant, Circle packing received higher scores with Elongated data as expected.*

Hypothesis 3:

VisTask 5 was designed to examine Hypothesis 3. When we compared the scores of *one-by-one* (3.91) and all-in-one glyph (3.48) we discovered that *one-by-one* scored higher, as expected. However, a paired t-test showed no statistical difference between them. Also, some users commented that the *one-by-one* option was more useful. This can be because the coloured dots used in *one-by-one* glyph can better scale with zoom-out when compared to the *all-in-one* glyph. The latter glyph can be better in a detailed view. As both received quite high scores we can conclude that our users find Glyphs useful.

> **Remark 4.4**. *The results showed that participants preferred* one-by-one *over* **all-in-one** *glyph on the global view which aligns with H3.*

Qualitative Results

We gathered some information from our open questions. Users' comments aligned with our Hypothesis 1. For example, user 20 stated that in T2-EC *"in tree layout, it is much easier to detect the comment level of each comment"*, however, he also commented in T4-CC *"it is difficult to obtain a nice general perspective from all comments from tree layout"*. As compact data has a wider and more dense set of comments it can be harder to analyse them with tree layout.

While most users commented that Circle packing can be confusing and hard to understand, users commented that it gets easier to understand as you get familiar with it. User 4 commented that *"Even though the tree, force and radial layouts seem easier to understand at first, I found that once you get the hang of it, the circle packing was the most useful layout to answer task 4"*, and user 10 stated that *"the circle packing layout seemed the most visually useful to me, but I needed a brief explanation to identify which circles were comment level 1 so as not to confuse them"*.

Also, we discovered that some participants needed more time to understand Force and Radial layouts. For example, when we follow the comments of user 11, in T1-EC and T2-EC the user commented that only the Tree layout was easy to understand and others, especially Circle packing, was very difficult. Then in T3-CC, the same user stated that *"in this task, I found it easy to use both display options (force and radial) in fact, I have a better idea of how they work"*. Finally, in T4-CC the user also found circle packing easier to use as the user got familiar with it.

> **Remark 4.5**. *Although tree layout seemed the most intuitive for the users at first glance,* ***the rest of the layouts have a potential*** *once the users know how to interpret them.*

4.5.4 Discussion

Previous studies gave some advice on the hierarchical visualisation methods suitable for some types of data and VisTasks [161] [128]. Nevertheless, they did not take into account the hierarchical shape

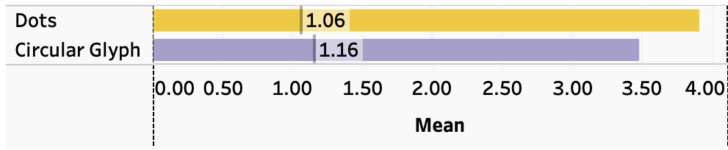


Figure 4.23: Mean values of Glyphs from VisTask 5 with Standard Deviation

dictated by the data, focusing instead on creating visualisation layouts that balance readability and compactness, without considering the data's inner structure. Other works in the literature visualised hierarchical data using a pre-defined layout. For example, Functree2 [44] visualised omics data using a radial layout, VisAC [22] used tree layout to visualise their ancestral data, and ConVis [79] and Eras [126] addressed data similar to our case study and used stacked bars and radial layout, respectively. Our findings, however, suggest that a categorisation of hierarchical data informs the visualisation method (layout) that best fits an overview of the data. Moreover, this research opens the avenue of analysing whether it is adequate to change the layout depending on the tendency of the sub-trees resulting from some queries or selections. For instance, when visualising subtrees in complementary charts, the categorisation of the entire tree may differ from that of the selected subtree, changing, for example, from a radial layout to a tree layout. This same idea can be applied to create responsive hierarchical visualisations, which would change depending on the size of the device as Hoffswell et al. [77] proposed to effectively present information based on the device context. Furthermore, this idea can be integrated when the size of the data is huge. For example, this huge dataset can be divided into regions and classified separately with our categorisation. Thus, we can visualise these regions using a combination of layouts.

Although our results showed that the Tree layout was selected by the majority of users as the most intuitive with both Elongated and Compact data, we found some evidence that Force and Radial layouts are useful for visualising broad data since users scored them higher with Compact than with Elongated data. Furthermore, we also discovered during our sessions and from user comments that some participants needed more time to understand Force and Radial layouts. Even though we gave our users explanations and exploration time, our sessions were short to fully understand how these graphs worked for users that were new to visualisations.

Moreover, the Circle packing achieved slightly higher scores with Elongated data than with Compact data, but it received the lowest scores of all the 4 layouts in every VisTask overall. This could be due to the users' unfamiliarity with the layout. It might also result from a lack of training in this type of graph, which made it less advantageous. This section of this dissertation shares some similarities with Zheng and Saldo's study [213] which explored hierarchical multivariate visualisations. Specifically, they also used Circle packing and mapped their glyphs on this visualisation. Their results aligned with our expectations and their Circle packing also received the lowest scores due to the low readability of the glyphs. Therefore, we suggest that the use of Circle packing could be further restricted to hierarchical data that are not densely crowded, i.e., trees with very few levels (maximum 3 to 4), or trees not particularly broad. In the case of broader and deeper hierarchies, Schreck et al. [157] analysed the treemap family providing global layouts for balanced and unbalanced trees. Our proposal is aligned with this work since we also try to find the best-fitted layout, and treemap among others, depending on the nature of the dataset.

Part II

Visualisation-Oriented Natural Language Interface to Analyse Hierarchical Multivariate Data

Chapter 5

Background on V-NLIs

In this chapter, we introduce the V-NLI (Visualisation-oriented Natural Language Interface) pipeline, extension of the DataVis (Data Visualisation) Pipeline presented in Chapter 2, Section 2.1, where we defined key vocabulary related to data visualisation. We now define the key terms specific to chatbot-based V-NLIs. Subsequently, we outline the categories derived from the vocabulary introduced for both fields to conduct a unified literature review. This literature review also encompasses the underlying technologies powering V-NLIs and the design methodologies employed in their development.

5.1 V-NLI Pipeline

Previously, in Chapter 2, section 2.1, we introduced the DataVis pipeline, which includes three spaces: **Data Space**, **Visual Space**, and **Interaction Space**. We discussed data visualisation characteristics in stages within each space, such as **Data Transformation**, **Visual Mapping**, **View Transformation**, and **Interaction**. Now, we extend this pipeline by incorporating a chatbot, which serves as a valuable enhancement at each step (see Figure 5.1). For instance, a chatbot can facilitate data transformations, select optimal layouts, adjust visual elements such as colours, and handle visual transformations, like executing a zoom command. Additionally, a chatbot can guide users through the interaction process by offering explanations and clarifications about the tools and features available. By integrating the V-NLI pipeline with the DataVis pipeline, we aim to provide a structured perspective for our literature review by examining the chatbot's role across various stages and properties of the visualisation process. This approach allows us to systematically investigate: (1) How do VisChatbots contribute to interactions with the Data

Space?, (2) How do VisChatbots contribute to interactions with the Visual Space?, and (3) How do VisChatbots enhance the user's interaction with the visualisation? In the subsequent sections, we will examine and define the core properties of chatbots to effectively analyse how NLIs with visualisations function, with a focus on understanding how, specifically, VisChatbots contribute to each stage of the data visualisation spaces.

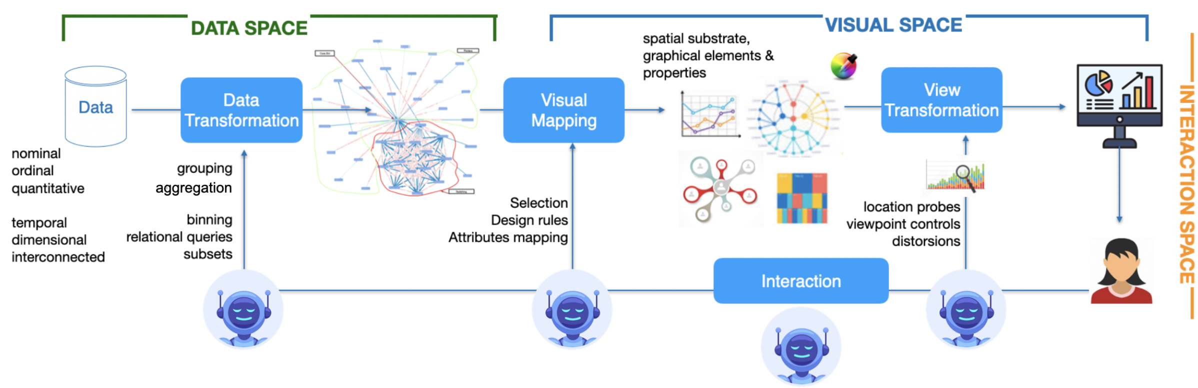


Figure 5.1: Overview of the Data Visualisation pipeline adapted from [167].

5.2 Chatbots

Chatbots are software systems that can engage in conversations with users [3], thereby representing a natural interface for them. This naturalness has favoured its spread in domains such as education [112], health [15], business [190] and, of course, fields such as visualisation analysis [133, 118].

In Figure 5.2, we propose a general characterisation of chatbots using four dimensions, named AINT, depending on how we view them. First, chatbots may have Anthropomorphic (A) properties such as appearance [36] and gender and also may be endowed with personality and emotions [189]. Second, as an Intelligent system (I), task-based chatbots can proactively make data-driven decisions to give support to users' activities, and social chatbots maintain meaningful and engaging conversations with their users. In any case, chatbots can also be enhanced through a variety of AI methods and techniques, for example predicting users' necessities and behaviours and thereby personalising the UX (User eXperience) [120]. Third, as a Natural language processing system (N), chatbots usually consist of an NLU (Natural Language Understanding) component [106], which interprets user intentions (i.e., inputs) and helps maintain the context of the conversation. Based on this understanding, chatbots provide appropriate responses, whether textual, auditory, or visual, depending on the system's design and the interaction mode. Those answer types (i.e., the outputs) can be either pre-defined or automatically generated. In the specific case of text, they are usually created by an NLG (Natural Language Generation) system [66]. Finally, as an interactive system (T), chatbots can be integrated with different interaction styles (WIMP, VR, XR) and be equipped with a multimodal interface through voice, text and gestures.

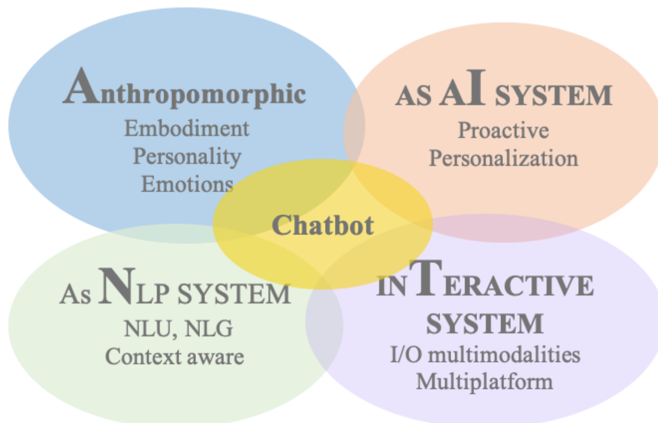


Figure 5.2: AINT—General characterization of a Chatbot based on four dimensions: A—Anthropomorphic, I—Intelligence, N—Natural Language Processing, and T—inTeractivity.

5.3 VisChatbots

Next, we focus on chatbots in the specific context of visualisation (see Figure 5.3). We analyse several aspects of the interactive space of a visualisation-oriented chatbot, including its user interface as well as its input and output mechanisms, which are listed next to them in the figure, and will be explained in the following. From this analysis, there will emerge the main V-NLI features (User Interface, Input, and Output, indicated in bold) and sub-features (indicated in italic) that will lead the analysis of the literature in this chapter.

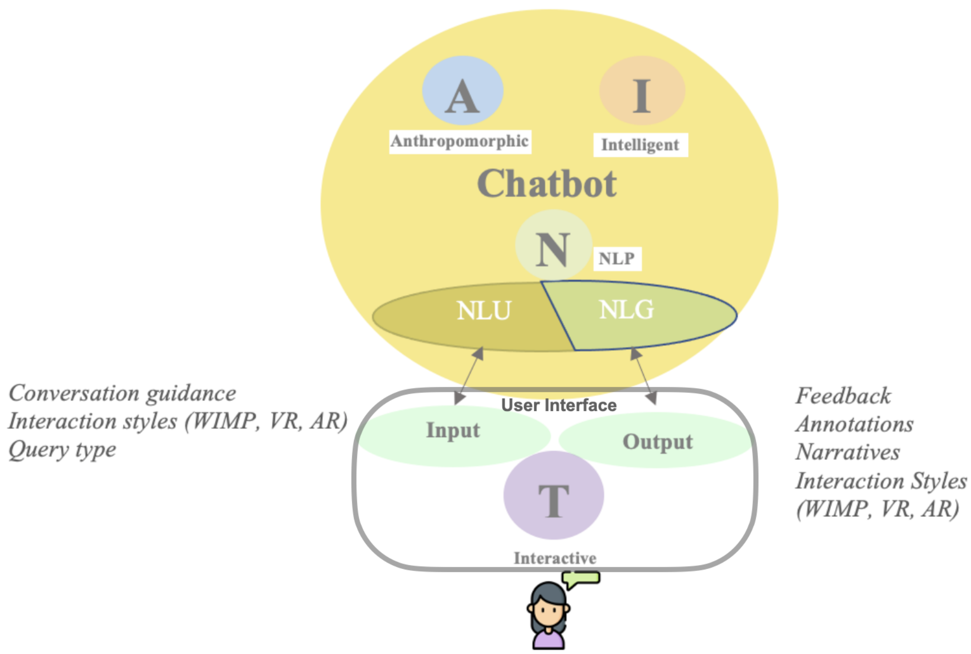


Figure 5.3: The interactive space's components of a V-NLI: User Interface, Input, and Output.

User Interface

Visualisation-oriented Natural Language Interfaces (V-NLIs) are interactive systems (AIN**T**) designed to facilitate the users' VisTasks. They can be designed using two different user interfaces (UI): a form-based interface and a chatbot-based interface. On one hand, a **form-based** V-NLI [176, 65] (see Figure 5.4) is usually composed of a text box that allows the users to introduce the VisQuery using natural language, though it also has other widgets, for example, to refine (filter) the resultant visualisation. Nevertheless, these forms are usually not designed to engage in follow-up questions with the visualisation system. On the other hand, a **chatbot-based** interface [16] (see Figure 5.5) is distinguished by a named entity (also known as an agent), with gender and appearance, as well as with the ability to recognise and express emotions, while having personality traits (i.e., empathetic, fun, neutral). Chatbots are usually presented to the users as a separate "chat window" from the visualisations. This window displays the conversation but also complementary outputs (explanation, charts, and others), as we will explore later. We can say that a chatbot-based V-NLI (**VisChatbot**) may have all of the aforementioned chatbot characteristics, i.e., AINT, meanwhile form-based V-NLI are potentially endowed with all of them except the anthropomorphic traits, i.e., INT.

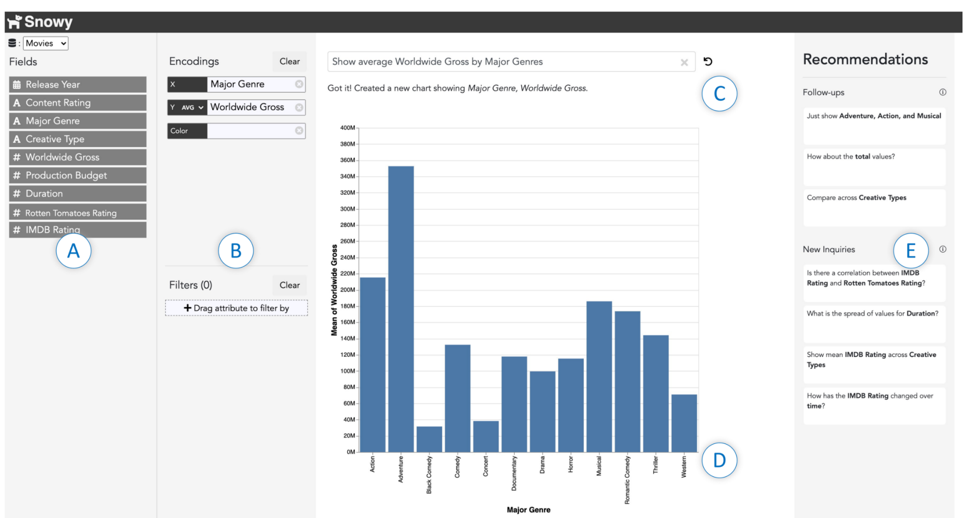


Figure 5.4: Snowy [176], a form-based V-NLI example. Dashboard including: (A) Attribute panel, (B) manual view specification and filter panel, (C) NL input box and textual feedback, (D) visualisation space, and (E) query recommendation panel.

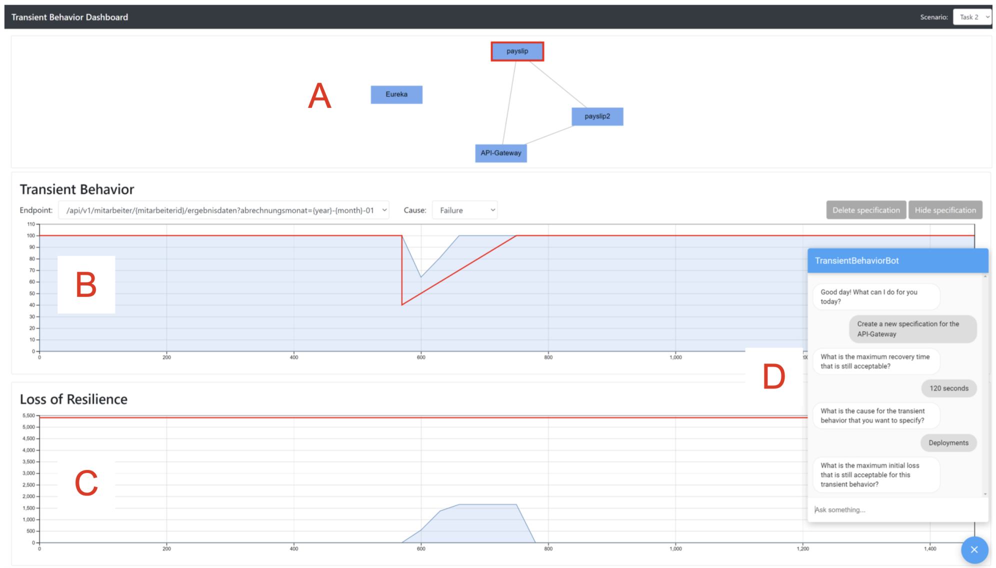


Figure 5.5: TransVis [16], a chatbot-based V-NLI (VisChat) example. Dashboard including; (A) Architecture visualisation, (B) and (C) area line graphs, and (D) chatbot window.

Input

The types of inputs (analytical questions) that a V-NLI system deals with are low- and high-level VisQueries. In **Low-level** VisQueries, the users explicitly describe their intent, for example, *"Show me action films that won an award in the past 10 years"*. Therefore, these VisQueries can be interpreted easily. In contrast, **High-level** open-ended VisQueries are naturally broader and their interpretation can be more complex [178, 4]. In many cases, these high-level analytical questions should be decomposed as a series of low-level VisQueries and be answered as such [169]. For example, to answer *"What are the trends in award-winning films?"* the system needs to infer the low-level VisQueries: first, visualise award-winning films over a certain period, and then show their relevant characteristics (genre, special effects, franchises and others). Whenever the V-NLI system is not able to answer this type of complex question, it might need to ask additional questions to the users. Note that both types of VisQueries allow the users to interact with the data through the seven-interaction methods (select, explore, reconfigure, encode, abstract/elaborate, filter, and connect) as defined in the description of the Interaction Space in Chapter 2, Section 2.1, at any of the steps (Data Transformation, Visual Mapping, and View Transformation) of the data visualisation pipeline as depicted in Figure 5.1.

Moreover, VisQueries can be One-turn or Follow-up. In **One-turn** VisQueries, the users ask the system in a single shot. Thus, even when the conversation may flow along several one-turn VisQueries, it may not be necessary for the V-NLI system to maintain the context of the conversation [104]. On the other hand, the users usually perform **Follow-up** VisQueries, which are a series of interconnected questions [181]. Therefore, the system should be able to remember the context of the conversation while answering the questions [81, 178]. For example, if the first VisQuery is *"Colour nodes by age"* and the second VisQuery is *"Now by gender"*, NLI understands that the user continues talking about nodes and wants to use the same function, colour, but now colouring them by gender.

Nevertheless, the underlying system (AI**N**T) may fail to understand the VisQueries of the users, thus, not meeting their expectations [191]. Moreover, inexperienced users may have difficulties expressing their VisQueries. Then, the design of the conversational system faces the challenge of understandability, i.e., the ability of the system to be aware of the users' intents, really knowing and grasping the nuances of users' intentions, and also the challenge of discoverability [179], i.e., the ability of the users to know what they can ask to the system. Indeed, the two properties are closely connected, as designing chatbots for discoverability can enhance their understandability.

Addressing the challenges of understandability and discoverability requires an interactive conversational system to guide users in effectively communicating their goals (or intentions). Well-known **Conversational Guidance** strategies are based on **help**—the chatbot gives the users hints on what to ask; **intent auto-complete functions**—the system makes suggestions of possible intents while the users are writing the intent [163, 171, 65, 84]; and **intent recommendations** [176]—after giving a response, the system suggests, based on data or on the previous turns of the analytical conversation, possible next intents to the users. Additionally, the understandability problem of NLIs is mainly derived from the biggest challenge that NL poses, which is ambiguity. One solution is to ask the users what they meant or to use disambiguity widgets [65, 163]. For instance, when the user's VisQuery is *"Show me medals for hockey"*, the NLI might not correctly interpret which type of hockey the user is referring to. Then, a widget may appear for the term *'Hockey'* showing two options *'indoor hockey'* and *'ice-hockey'* as both of these sports are basically called hockey. Thus, users can select the right one either through direct manipulation or by using natural language.

In the context of follow-up VisQueries, the V-NLI should help the users to transition through the different visualisation states of the analysis. Indeed, research studies concluded that users prefer to carry out *analytical conversations*, meaning users want to go beyond the first visualisation they receive when making a request to the conversational interface [74]. Nevertheless, previous Conversational Guidance strategies (help, auto-complete, and recommendations) may be sufficient for helping with the (initial) users' intent but could be insufficient for inferring the user's transitional intents (elaborate, adjust/pivot, start new, retry, and undo) throughout the different visualisation states (interaction methods such as select attributes, filter, encode, transform) of an analytical conversation [191]. Therefore, *intelligent* Conversational Guidance (A**I**NT) approaches are needed to predict users' goals based on their interactions throughout the analytical conversation and then proactively guide the user.

Another aspect of analytical visualisations is the need to handle co-reference, as users may refer to the same entity differently during a conversation, such as by using pronouns. Fortunately, nowadays the most common NLP toolkits such as spaCy and AllenNLP incorporate components for co-reference resolution in their pipelines [139]. Moreover, an interesting case of co-reference arises when natural language interfaces coexist with other interaction styles (**Multimodality**) such as menu selection (WIMP—Window Icon Mouse Pointer) and direct manipulation (XR—Virtual or Augmented Reality) [86]. That is, users' natural language (NL) VisQueries may refer to actions they performed through clicks, gestures, or eye-gaze. Consequently, the V-NLI system should track these non-textual references—what users did, not just what they said. Thus, there should be a way of translating (WIMP, VR, AR) manipulations in the visualisation to text (with named entities) and so to be ready to be solved by the co-reference model.

Output

In addition to the requested visualisation, a V-NLI can consider **Complementary Output** such as **Feedback**, either **text** or **visual**: (i) to inform about the VisQuery's success or failure, (ii) to justify relevant decisions taken by the system, (iii) to provide users with additional explanations—such as textual descriptions, oral explanations, graphs, statistics, and annotations—to help them better interpret the resulting visualisation, and (iv) to display changes in the User Interface (highlighting menus and buttons). Specifically, annotations are superposed visual elements that enhance the generated visualisation and thereby further communicate more information [115]. Another type of complementary output is a visual narrative, which is text combined with images presenting the information with narrative components (actor, plot, setting) [162]. Finally, when there are other **Interaction Styles** integrated into the V-NLI system, the output should be synchronised to help the users be aware of the operation performed (e.g., filters updated in WIMP), and it should also be enhanced to facilitate a better understanding of the required visualisation (e.g., overlying images in AR) and to better communicate the system's response (e.g., haptic feedback in VR).

5.4 Review of the Literature

We identified 20 works that explored and presented V-LNIs to conduct a thorough review of the literature. We included 10 articles that are VisChatbots [91, 214, 43, 95, 19, 57, 154, 16, 116, 122] and 10 articles that are form-based NLIs but have some chatbot characteristics such as providing feedback [178, 176, 65, 169, 108, 81, 174, 39, 84, 171]. In the following, we summarise the review categories used to evaluate these works. More information can be found in [96]

5.4.1 Review Categories

To analyse the collected works, we use the categories defined in Chapter 2, Section 2.1, exploring the three spaces involved in the DataVis pipeline (Data, Visual, and Interaction Spaces), as well as VisChatbot characteristics (the interface, the Input, and the complementary Output) in Section 5.3. Furthermore, we review the state of the art of underlying technologies of V-NLIs and chatbots' design methodologies. In the following, we summarise the categories;

Categories related to the **Data Space** described in Chapter 2, Section 2.1:

- Description of data.
- Data type (Tabular or Complex).
- Attributes (Nominal, Numerical, Temporal, Spatial).
- Data transformation.

In relation to the **Visual Space** (see Table 5.1), the characteristics presented in Chapter 2, Section 2.1 include:

- Visualisation Category (Basic or Advanced) and Type.
- Abstract (Lines, Points, Bar) and Symbolic (glyphs, icons) graphical elements.
- Visual Mapping Identification (Fixed, User-defined, Rule-based, Intelligent).
- View Transformation (Single or Multiple).
- Interaction Style (Basic—WIMP or Advanced—NL) .

In the **Interaction Space**, we collect information about the seven interaction methods proposed by Yi et al. [209]: select, explore, reconfigure, encode, abstract/elaborate, filter and connect, presented in Chapter 2, Section 2.1. These interaction methods also directly or indirectly relate to VisTasks which is explained in the same section.

Finally, regarding chatbots, we propose the following categories:

- *V-NLI Interface* (chatbot-based or form-based).
- *Input*:
  - VisQuery Type (low or high);
  - One-turn or Follow-up VisQueries;
  - Conversational Guidance (Help, Auto-complete, Recommendation);
  - Multimodality (WIMP, Touch, Gestures).
- *Output*:
  - Feedback (textual or visual): inform, justify, decisions, additional explanations (text, oral, graph, statistics, annotations), UI changes (menus, buttons) and narratives.
  - Interaction Style (WIMP, VR, AR).
- *Technology behind V-NLIs*
- *Design Methodologies*

5.4.2 Data Space

We analysed all the research works in terms of the main characteristics involved in the **Data Space**; see Figure 5.6.

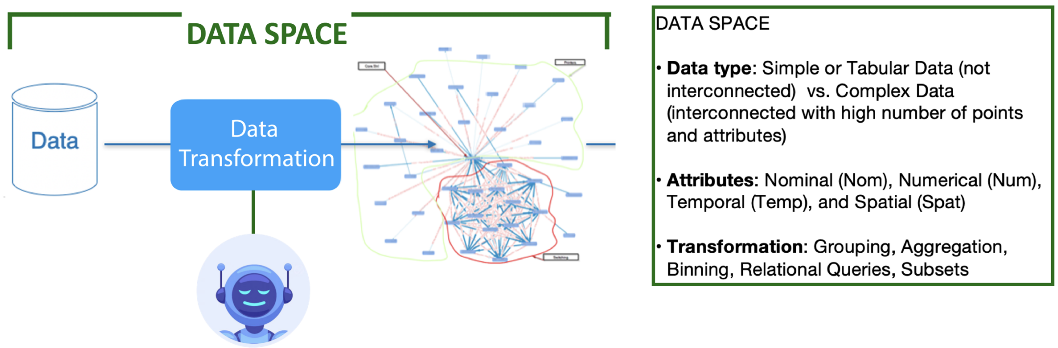


Figure 5.6: Data Space overview and the main characteristics of the data involved in the visualisation pipeline.

The analysed research works cover a diverse range of topics including movies, sports, coronavirus, finance, and more, each described through various visualisation techniques. We found that most of them used multidimensional **tabular** data [176, 39, 81, 174, 175, 122, 154, 169, 57, 116, 214, 95, 43, 91], while some of them also included spatial data [81, 39, 154, 84]. Few of the twenty visualisations systems used **complex data**. For instance, Bieliauskas and Schreiber [19] have data related to software bundles and services such as OSGi bundles, and Orko [178] uses network data displaying the relationships between football players. Furthermore, ConVisQA [171] has hierarchical data that is collected from online conversations and FlowNL [84] works with flow data such as hurricanes. Finally, Data@Hand [108] has sequential temporal data (e.g., sleep time during each night), and TransVis [16] has transient data which is a data type that is relevant to a time; in this case, it is the quality of software services over time.

Regarding the **Data Transformation** step, 10/20 V-NLIs used a kind of data transformation. For example, Ava [91] and Iris [57] are both designed to facilitate data science tasks and they transform data to perform statistical analyses, such as logistic regression and finding correlations, respectively. Similarly, Valetto [95] and Boomerang [116] compute the correlation between attributes, and the latter also finds aggregated values. Data@hand [108], InChorus [174], Evizeon [81] and Snowy [176] also perform aggregation functions such as average and sum. GeCoAgent [43] also computes aggregation functions, as well as, other data transformations such as clustering, regression, etc., while extracting genomics data. Finally, Talk2Data [169] calculates the difference between numerical attributes. In general, the reviewed methods applied basic transformations that were not highly complex, i.e., those that entail more "intelligent" data processing (e.g., PCA).

5.4.3 Visual Space

Regarding the **Visual Space** (Figure 5.7), we summarise in Table 5.1 the works in terms of Visual Mapping components (graphical elements, identification, and type of visualisation or substrate) and View Transformation details (use of single or multiple views and the type of interactions). Most of the articles employed explicitly **basic** visualisations [91, 116, 43, 108, 175, 214, 57, 39, 154, 81, 176, 95, 122, 169]. The most common methods are bar charts, line charts and scatter plots.

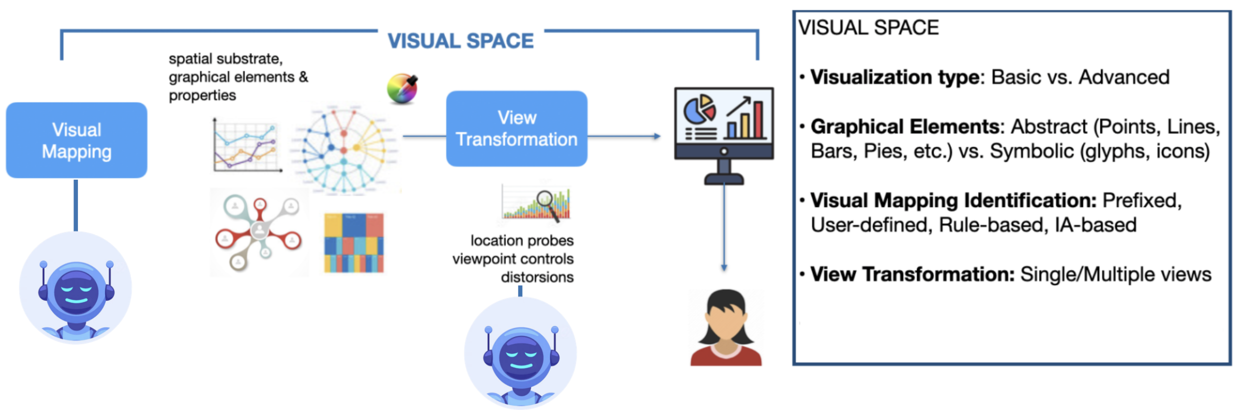


Figure 5.7: View Space overview and the main characteristics of the Visual Mapping and the View Transformation steps.

Most of the V-NLIs include a combination of these common methods [116, 176, 169, 108, 214, 122]. Additionally, Chat2Vis [122] includes a box plot, Talk2Data [169] has a pie chart, and Gamebot [214] offers users game shot charts about football and basketball games, as well as, displays games statistics in tables. Moreover, some V-NLIs include a map chart, while [39, 81, 154] have 2D maps in addition to the most popular methods, and [84] includes a 3D map. Some V-NLIs have only one visualisation method: GeCoAgent [43] has a pie chart, Ava [91] has a line chart, Valetto [95] and Iris [57] have scatter plots, and DataBreeze [175] uses dots to visualise every data point individually.

Only a small percentage of these studies used **advanced** visualisation methods. For example, Bieliauskas and Schreiber [19] and Orko [178] (see Figure 5.8) implemented network visualisations as their main visualisation. On the other hand, Tansvis [16] uses a line graph as the main visualisation for analysing transient data (quality of the software system over time), though it has a network graph for displaying the overview of the software system, where users can select

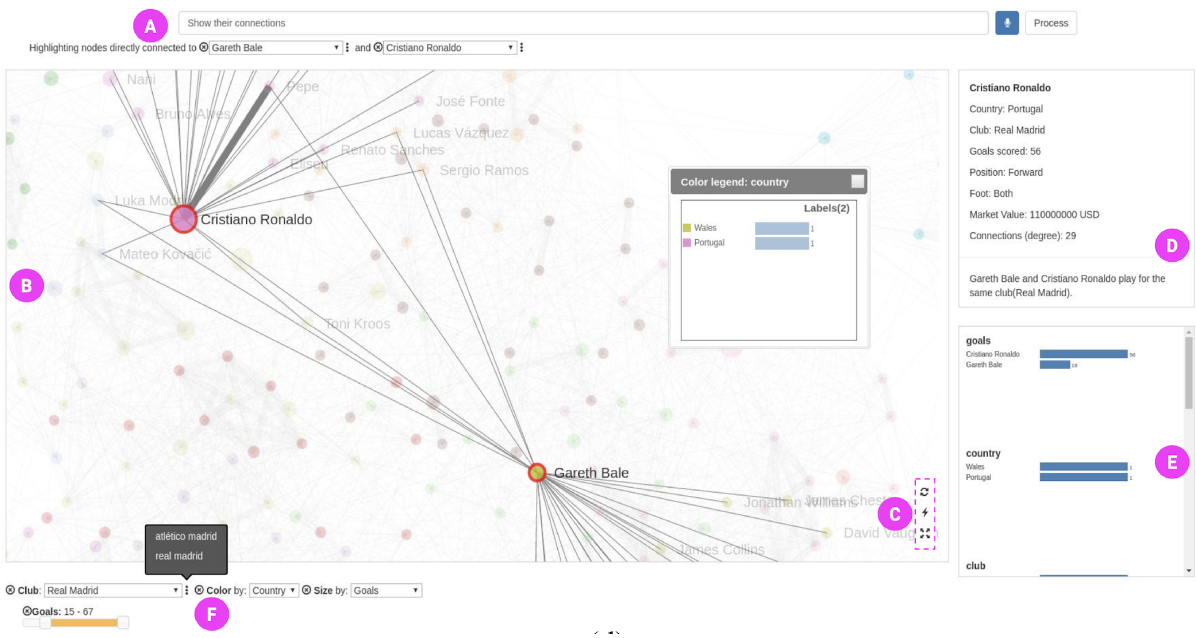


Figure 5.8: Orko [178] including A) Input box, B) Network visualisation, C) Access icons, D) Details container, E) Summary container, and F) Filter and visual encodings.

a part to explore transient behaviours. ConVisQA (see Figure 5.9) [171] created a novel design to show the hierarchical structure of conversations using stacked bar charts with indentations to show the hierarchy. ConVisQA also displays the conversations on the right-hand side of the screen. InChorus (see Figure 5.10) [174] supports popular basic visualisations such as bar, line and scatter, and they also include one complex option, parallel plots. FlowNL (see Figure 5.11) [84] used flow visualisation to show flows occurring on the earth (e.g., hurricanes). Moreover, all V-NLIs, including basic and advanced visualisations, used abstract graphical elements (lines, points, bars).

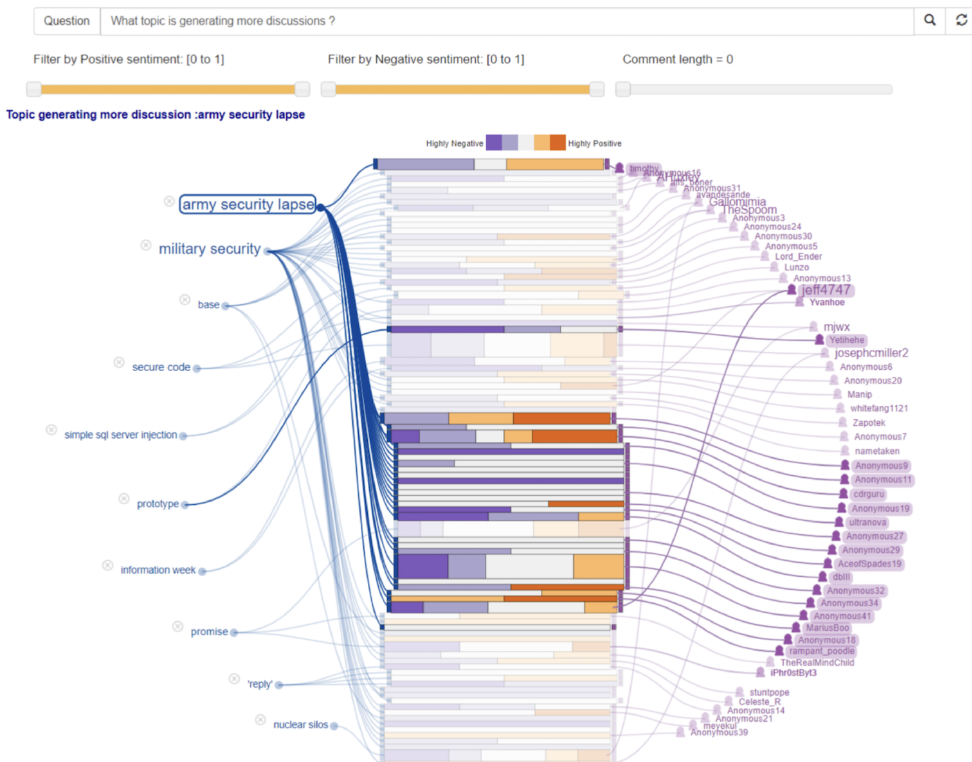


Figure 5.9: ConVisQA [171] including on top the input box, filters, and the main visualisation

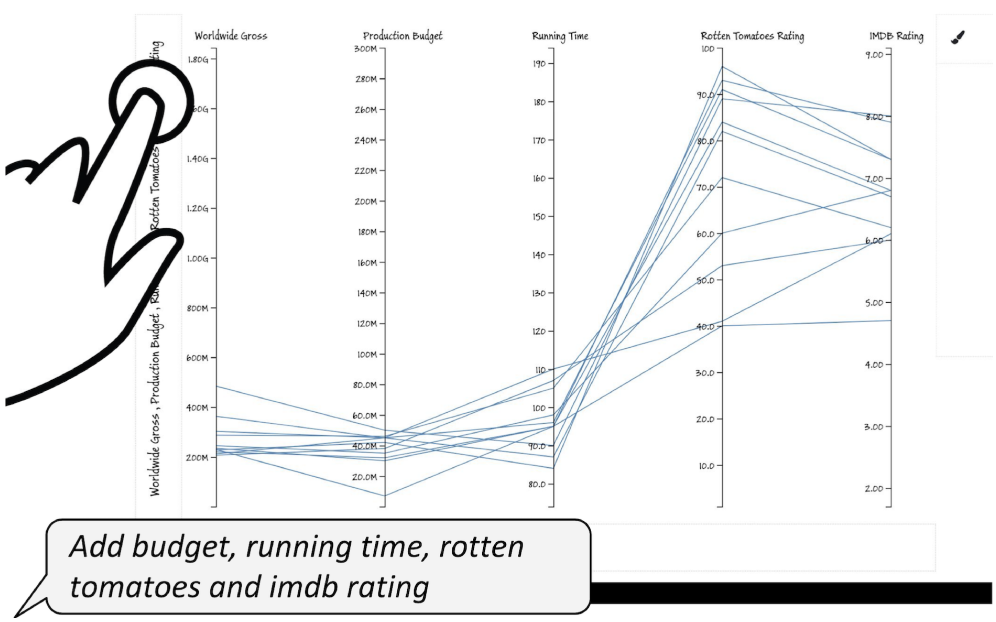


Figure 5.10: InChorus [174], Parallel plots visualisation.

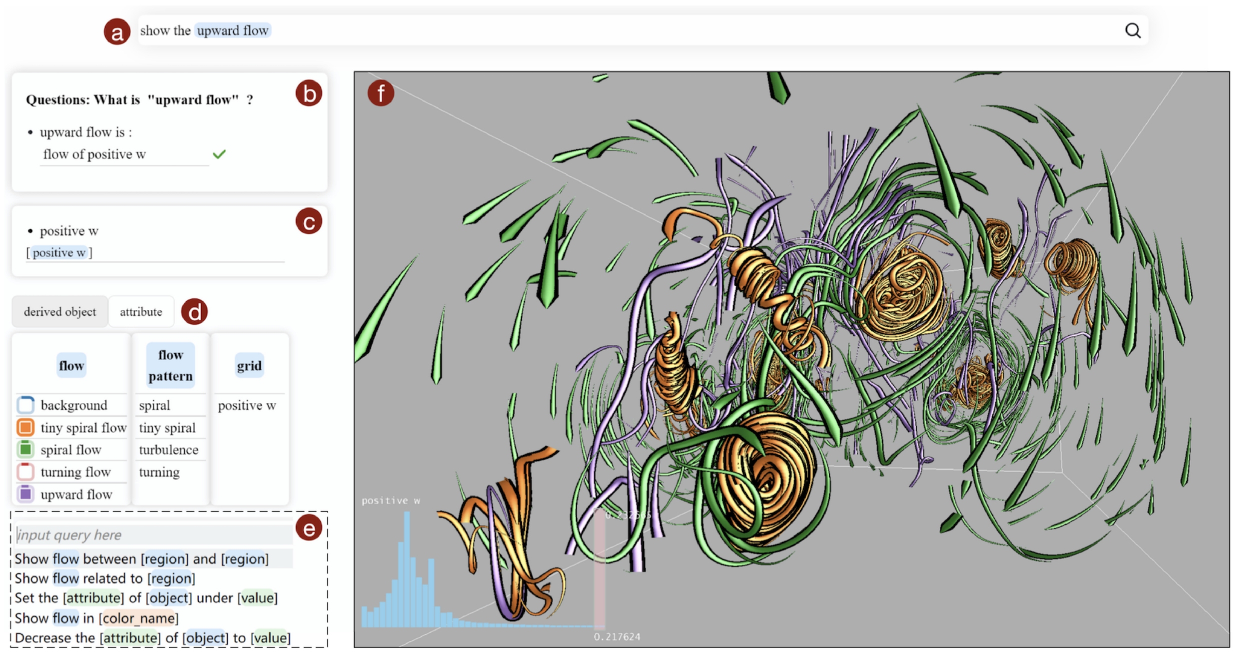


Figure 5.11: FlowNL [84] including a) input box, b) dialogue box to solve unknown terms, c) query formula, d) objects, e) suggested queries, and f) visualisation.

As defined in Chapter 2, Section 2.1 we categorised the **Visual Mapping Identification** into three types: **fixed**, **user-defined**, and **rule-based**. In the previous works, we found that half of them only have **fixed** visual mapping [171, 108, 175, 84, 178, 39, 19, 43, 16, 95]. Meanwhile, only a couple of them support **user-defined** mapping [91, 57], while a few use only the **rule-based** visual mappings [116, 176, 169, 214] method. Finally, there are V-NLIs that support a combination of two visual mappings [174, 154, 81, 122]. For example, Data@Hand [108] is a mobile application on which users can track their daily steps and sleep time, among others. It has basic **fixed** visualisations that are displayed when users open the application. On V-NLIs such as Orko [178], ConVisQA [171], Miva [39], and DataBreeze [175], when the dataset is uploaded, data are directly displayed with one **pre-defined** visualisation method. FlowNL [84] displays flow visualisations on a 3D world map with NL commands. GeCoAgent [43] and Bieliauskas and Schreiber [19] both have fixed visualisations that are updated with NL VisQueries. TransVis [16] and Valetto [95] automatically generate visualisations from natural language, though both of these systems have only one fixed visualisation. TransVis uses transient data (quality of service vs. time) that is visualised with a line area graph and Valetto uses a scatter plot to visualise tabular data (e.g., cars).

Some V-NLIs allow **user-defined** visual mapping. For instance, Ava [91] and Iris [57] use NL to perform complex data science tasks such as statistical analysis and both support visualising data with one available visualisation when asked.

Snowy [176] is one of the V-NLIs that support **rule-based** visual mapping to select layouts and graphical elements. It has three visualisation methods—bar chart, scatter plot, and line chart—and the system automatically selects and updates the visualisation method depending on the user's VisQueries and pre-defined visualisation mapping rules. They use an adaptation of the "Show Me" [121] algorithm to decide the visualisation method according to data attributes. They follow rules such as displaying a scatter plot if there are two quantitative attributes on the x and y axes or displaying a bar chart if there is one qualitative and one categorical attribute.

Similarly, Talk2Data [169] generates multiple visualisations after NL VisQueries and the system also annotates the visualisation and gives textual answers. The system follows a rule-based approach in which it associates different data facts with different visualisation methods. Data facts are extracted from the data. For example, a categorisation fact includes categorical data and is associated with a bar chart.

As we stated earlier, there are four V-NLIs that support a combination of several visual mapping strategies. For example, InChorus [174] uses rule-based mapping to automatically select a visualisation depending on the attribute type detected from users' VisQueries and it also allows users to explicitly request a visualisation method. Onyx [154] uses fixed visualisation methods but users can change these methods using WIMP or NL. Moreover, Evizeon [81] is the only V-NLI that combines fixed and rule-based mapping. It includes multiple fixed visualisations, however, if a user's VisQuery cannot be answered by the existing visualisations, the system generates a new, appropriate visualisation using the aforementioned "Show Me" algorithm [121]. Finally, Chat2Vis [122] is the only V-NLI that uses artificial intelligence (Large Language Models—LLM) for visual mapping. Moreover, users can specify in their VisQuery which type of chart they want to use to visualise the data.

Related to **View Transformation**, most of the explored works use a single view to visualise data [127, 91, 171, 175, 84, 214, 43, 174, 57, 154, 178, 95]. Nine V-NLIs have multiple views. Boomerang [116] displays multiple recommended visualisations simultaneously, while Talk2Data [169] generates multiple visualisations with annotations in a visualisation narrative style. In the case of MIVA [39], there are three fixed visualisations (bar, line, and map), which are simultaneously updated to answer users' VisQueries. Similarly, Evizeon [81] supports synchronised multiple views. Moreover, in Data@hand [108], Talk2Data [169], and TransVis [16], multiple visualisations can be observed. Finally, Orko [178] and FlowNL [84] also have complementary visualisations in addition to primary ones.

Furthermore, there are V-NLIs that also support other view transformations. For example, some systems support Focus+Context [108, 19, 171, 178, 116]. Data@hand's [108] users can analyse their sleep time across a month and they can ask the system to show the days the user woke up at 8 a.m. In this way, the system highlights the days the user woke up at 8 a.m. but also displays in the background in grey the data for the whole month. Similarly, Bieliauskas and Schreiber [19] and Orko [178] have network visualisations and users can highlight certain nodes to see in detail while viewing the whole visualisation in the background. ConVisQA [171] allows users to see the whole hierarchy while highlighting certain parts in response to users' NL VisQueries. On the other hand, Boomerang [116] uses an approach similar to small multiples (i.e., grid-like layout) on the right-hand side of the screen as recommendations while letting users ask questions on the left-hand side, as well as displaying users charts in the chat window.

Table 5.1: Summary of V-NLIs in defined visualisation categories. Visualisation Category (Basic and Advanced), Graphical Elements (Lines, Points, Bars), Visual Mapping Identification (Fixed, User-defined, Rule-based and Intelligent), and View Transformation (Single and Multiple views).

V-NLI	**Visual Space**			
	Visual Mapping			**View Trans.**
	Visualisation Category (Type)	**Graphical Elements**	**Visual Mapping Identification**	
ACUI [19]	Adv (Network)	Lines, Points	Fixed	Single
Ava [91]	Basic (Line)	Lines	User-defined	Single
Boomerang [116]	Basic (Bar, Scatter, Line)	Lines, Points	Rule-based	Multiple
Chat2Vis [122]	Basic (Bar, Scatter, Line, Box-plot)	Lines, Points, Bars	Intelligent & User-defined	Multiple
ConVisQA [171]	Adv (Hierarchical stacked bar)	Bars	Fixed	Single
Data@Hand [108]	Basic (Bar, Line)	Lines, Bars	Fixed	Multiple
DataBreeze [175]	Basic (Dots)	Points	Fixed	Single
Evizeon [81]	Basic (Bar, Scatter, Line, Map)	Lines, Points, Bars	Fixed & Rule-based	Multiple
FlowNL [84]	Adv (Flow) & Basic (Bar, Map)	Lines, Bars	Fixed	Multiple
GameBot [214]	Basic (Bar, Line, Table, Shot)	Lines, Points, Bars	Rule-based	Single
GeCoAgent [43]	Basic (Pie)	Pies	Fixed	Single
InChorus [174]	Adv (Parallel) & Basic (Bar, Scatter, Line)	Lines, Points, Bars	Rule-based & User-defined	Single
Iris [57]	Basic (Scatter)	Points	User-defined	Single
MIVA [39]	Basic (Bar, Line, Map)	Lines, Points, Bars	Fixed	Multiple
ONYX [154]	Basic (Bar, Scatter, Map)	Points, Bars	Fixed & User-defined	Single
Orko [178]	Adv (Network) & Basic (Bar)	Lines, Points, Bars	Fixed	Multiple
Snowy [176]	Basic (Bar, Scatter, Line)	Lines, Points, Bars	Rule-based	Single
Talk2Data [169]	Basic (Bar, Scatter, Line, Pie, Area)	Lines, Points, Bars, Pies	Rule-based	Multiple
TransVis [16]	Adv (Network) & Basic (Line)	Lines, Points	Fixed	Multiple
Valetto [95]	Basic (Scatter)	Points	Fixed	Single

5.4.4 Interaction Space

The **Interaction space** refers to all the interactions that users can make throughout the different stages of the visualisation pipeline (see Figure 5.12).

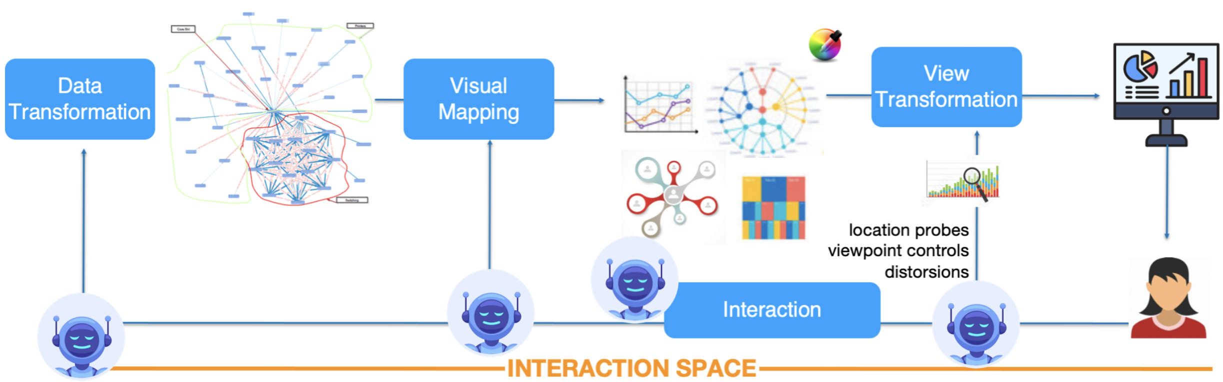


Figure 5.12: Interaction Space affects all the steps of the visualisation pipeline.

In the literature, the use of different interaction styles varies. Most V-NLIs use both Basic (WIMP) and, naturally, Advanced (NL) interactions [171, 108, 175, 81, 214, 174, 39, 178, 176, 154, 95, 84, 16], while some of them use only Advanced (NL) interactions [19, 91, 116, 57, 169, 43, 122]. Next, we analyse the interaction techniques, focusing on how different systems support both basic and advanced user interactions.

The most used interaction techniques are **select** and **filter** among all included techniques (**encode**, **reconfigure**, **explore**, **abstract/elaborate**, and **connect**). V-NLIs such as [116, 16, 108, 19] use NL to interact with visualisations selecting (marking a data point) and filtering (showing something conditionally) data according to user VisQueries. Boomerang [116] selects and filters data at the data transformation stage to create visualisations using NL. Data@hand [108] and TransVis [16] also use these techniques at the data transformation stage to update visualisations. Similarly, GeoAgent [43] and Valetto [95] use NL to update visualisations using filtering at the data transformation stage, while Chat2Vis [122] does this to generate visualisations.

Others, such as [171, 174, 39, 154], use both basic and advanced to filter visualisations at the visual mapping stage. Also, Evizeon [81] and Snowy [176] use both basic and advanced interaction techniques at the visual mapping stage to filter visualisation and [81] uses advanced interaction while using the select method. Orko [178] and Databreeze [175] use both NL and direct manipulation to filter and select data on visualisations. Moreover, Gamebot [214] asks users if they want to see a visualisation related to their VisQuery. Before displaying the visualisation, the chatbot asks questions to filter the data and customises it based on the user's responses, providing options with buttons. Similarly, Ava [91] uses NL to filter data and not visualisations. It uses NL to perform complex data science tasks such as statistical analysis and generating visualisations from libraries. Finally, Talk2Data [169] uses advanced NLP-based interaction techniques when labelling selected data visualisation (maximum sale), and FlowNL [84] uses both interaction styles.

The next most used method is **Encode** [175, 178, 176, 174, 154, 16, 95, 57, 169, 122]. For example, [175, 174, 178, 176, 154] allow users to colour and size data points and add/remove attributes by using Basic (WIMP) and Advanced (NL) interactions at the visual mapping stage. On the other hand, Valetto [95] and TransVis [16] use NL commands to add or remove attributes at the data transformation stage. Similarly, Iris [57] uses NL to interact with data in Visual Mapping (i.e., users can select different attributes for axis), but not with View Transformations. On the other hand, Talk2Data [169] and Chat2Vis [122] only interact at the Data Transformation stage, allowing NL VisQueries to colour visualisations.

The **reconfigure** method is supported by four V-NLIs [175, 174, 95, 81], which are used to change the visual perspective of the data in the visual mapping. For instance, Valetto [95] uses gestures (a basic interaction) to flip the axis in the visualisation mapping stage. InChorus [174] uses both basic and advanced interaction methods, such as re-ordering data in the step of data transformation, to reconfigure the visualisation. Similarly, in the same step, Databreeze [175] uses a combination of basic and advanced interactions to rearrange data points and Evizeon [81] uses advanced interactions for this task.

Furthermore, the **explore** method, which is considered to be zooming and panning in the View Transformation stage, is used in four V-NLIs [178, 174, 16, 81], all with basic interactions. It should be noted that Evizeon [81] and Orko [178] also automatically zoom in/out to the part of the visualisation that is related to users' VisQuery, though users cannot ask it to zoom in directly using NL. The **abstract/elaborate** method is used in four V-NLIs [108, 174, 176, 16] to drill down to show more details. For example, Data@Hand [108] transforms data to show average hours of sleep over various months, and users can choose the visual mapping to see each month separately in more detail using NL. Similarly, Snowy [176] uses NL to do drill downs, while, on the other hand, TransVis [16] uses direct manipulation. InChorus [174] uses both modalities.

Finally, the **connect** method is only used by two V-NLIs [19, 178]. Both of these V-NLIs have network visualisation and use the connect method to highlight the relationships between links using Advanced interactions (i.e., using Focus+Context visualisations). While Bieliauskas and Schreiber

[181] perform this at the data transformation stage, Orko [178] does this at the visual mapping stage.

5.4.5 V-NLI Interface

In the following, we analyse V-NLI interfaces, focusing on whether they are chatbot-based or form-based. We also examine the input characteristics, including the type of VisQuery (low or high), the interaction style (one-turn or follow-up VisQueries), and the presence of conversational guidance features such as help, auto-complete, and recommendations. Additionally, we explore the multimodal aspects of these interfaces, considering various input modalities like WIMP, touch, and gestures. Table 5.2 summarises the chatbot input categories in related work. Among the reviewed works, half of them, integrated **VisChatbots (Chatbot-based V-NLIs)** [91, 57, 116, 214, 43, 19, 154, 95, 16, 122]. These V-NLIs have a chat window in which users can engage in conversations with a bot to analyse data visualisations. In some tools, the chat window is separated from the main visualisation dashboard [19, 43, 154, 16, 95], and in others, the visualisations are displayed in the chat windows [91, 57, 214, 116, 122]. For instance, both Iris [57] and Ava [91] were developed to help users perform complex data science tasks such as statistical analysis. While Iris [57] displays visualisations in a single chat window, Ava [91] has two windows, one containing the chatbot and the other showing the actions the chatbot performs, such as displaying visualisations. Moreover, we consider the other half of the approaches to be **Form-based V-NLIs** [108, 175, 84, 174, 81, 176, 169, 171, 178, 39].

When we explored different **Low and High VisQuery Types**, we found that most of the previous research presented V-NLIs that support only low-level VisQueries [19, 57, 91, 171, 214, 16, 95, 116, 108, 175, 84, 178, 81, 43, 174, 176, 154, 39]. For instance, in [171, 175, 174, 176, 108, 154, 39, 19, 84, 116, 95], users can ask direct VisQueries and receive answers such as filtered or highlighted data points on visualisations or new visualisations. Finally, there are two V-NLIs that support both low and high-level VisQueries, Talk2Data, which is form-based [169] and Chat2Vis, which is chatbot-based [122]. Specifically, Talk2Data [169] uses high-level questions to interact with data using basic interaction techniques such as filtering, and they split high-level VisQueries into smaller sub VisQueries to find answers. An example from Talk2Data is, *"Which genre has more user reviews, fiction or non-fiction books?"* They break down this question into two: *"How many reviews does the fiction book category have?"* and *"How many reviews does the non-fiction book category have?"*. On the other hand, Chat2Vis [122] is able to understand more complex VisQueries such as *"Show the number of products with a price higher than 1000 or lower than 500 for each* *product name in a bar chart, and rank the y-axis in descending order?"* using several LLMs, which generate correct visualisations. Nevertheless, these models require some refinements because they may generate unnecessary extra information.

Table 5.2: Summary of input chatbot categories of V-NLIs. V-NLI interface (chatbot-based or form-based), VisQuery Type (low or high), Follow-up VisQuery, Conversational Guidance: Help (data-based, user-based: based on what the user can ask), Auto-complete and Recommendation (recommend next action from D: Data, N: previous NL intent, W: previous WIMP interaction), and Input Modality.

V-NLI	V-NLI Interface		Input					
	Chatbot	Form	VisQuery T.	Follow-Up	Conversational Guidance			Multimodal.
					Help	Autocom.	Recom.	
ACUI [19]	✓		low					
Ava [91]	✓		low		Hint/help		D, N	
Boomerang [116]	✓		low					
Chat2Vis [122]	✓		low & high					
ConVisQA [171]		✓	low			✓		WIMP
Data@Hand [108]		✓	low				D	Touch
DataBreeze [175]		✓	low	✓				Touch
Evizeon [81]		✓	low	✓		✓		WIMP
FlowNL [84]		✓	low			✓		WIMP
GameBot [214]	✓		low					WIMP
GeCoAgent [43]	✓		low			✓		
InChorus [174]		✓	low					Touch
Iris [57]	✓		low					
MIVA [39]		✓	low					WIMP
ONYX [154]	✓		low		data-based			WIMP
Orko [178]		✓	low	✓			N	Touch
Snowy [176]		✓	low	✓	data-based		D, N, W	WIMP
Talk2Data [169]		✓	low & high				D	
TransVis [16]	✓		low		user-based			WIMP
Valetto [95]	✓		low		user-based			Gestures

Additionally, these VisQueries can be only One-turn or **Follow-up**. There are only a few V-NLIs that support follow-up VisQueries [81, 176, 175, 178] and all of them support only low-level VisQueries. After each VisQuery, Snowy [176] recommends follow-up VisQueries on a list. On [178, 175, 81], users can refer to entities using determiners and pronouns.

One of the important characteristics of chatbots is having **Conversational Guidance**. In the visualisation context, chatbots can help users to ask the right questions, suggest possible VisQueries, navigate them through visualisations, and explain the tool operations that chatbots can perform. According to the our review, eight of the existing tools do not provide [19, 39, 175, 174, 116, 214, 57, 122] the user with any conversational guidance, while the rest of the tools (12/20, five of them chatbot-based) recommend VisTasks or VisQueries [176, 91, 178, 108, 169], help users [154, 91, 176, 16, 95], or auto-complete VisQueries [43, 171, 84, 81] designed to increase the discoverability of the NLI, helping users to understand what the NLI is capable of doing.

For example, users can ask for help from the chatbot in Valletto [95] and TransVis [16] regarding what users can ask the chatbot. Ava [91] gives hints on how to execute actions based on previous interactions. Onyx [154] helps with what it is able to do, and when something is not clear, it gives users instructions to go into the training interface and teach the system. Snowy [176] supports users by providing possible intents based on data before starting the analysis.

Moreover, Ava [91] gives users recommendations about how to continue the analysis, i.e., which actions it can do next. It also gives the users choices and asks them follow-up questions about whether they want to perform the action that the chatbot recommended. These recommendations are based on data and previous users' intents expressed in natural language. Data@Hand [108] and Talk2Data [169] recommend intents to users according to the data, and Orko [178] suggests to users possible operations on tool-tip when the system is not sure about a user's VisQuery. Snowy [176] offers three different kinds of recommendations. The first one is recommendations depending on the data, which are displayed at the beginning to start the analysis since users may sometimes be new to the dataset and do not know what to ask. Moreover, it offers users recommendations as a follow-up intent depending on previous NL intents and WIMP interactions. Furthermore, some V-NLIs are designed to collect specific information from users in a structured format in which chatbots ask questions or give the users prompts to complete the analysis [214, 57, 91, 43].

Finally, 13 of the V-NLIs have additional **Multimodality** to Natural Language (NL). For example, [19, 81] have ambiguity widgets, which serve as slider filters, allowing users to interact and refine their selections. In addition, with V-NLIs [108, 174, 178, 175], users can interact with the user interface using touch. Users can also select filters and interact with data without using NL. It should be noted that these systems have synchronised input modalities. For example, in [178], users can select a node with touch and ask a VisQuery about that node. Moreover, in [175], users can select data points and ask the system to move them, for example, to the left-hand corner. Similarly, [39, 176] have synchronised input modalities for, when a user selects a part of the visualisation using the mouse while answering the VisQuery, the system remembers this selection. FlowNL [84] and ONYX [154] have filters through which users can interact with them using WIMP. In TransVis [16], users can employ the WIMP to select a part of visualisation to explore in depth, while Gamebot [214] offers the users buttons during the conversation and Valetto [95] uses gestures to change visual encoding such as flipping the axis.

5.4.6 Chatbot Output

Finally, we explored the **Output** categories of the chatbot. Giving **Feedback** is one of the most important qualities of chatbots. All of the works in this literature review give the users **textual feedback** and some of them give **visual feedback** as well. The only exception is Chat2Vis [122], which, probably due to its recentness, is not yet integrated into a visualisation platform. Basically, textual feedback is used to inform or justify chatbot decisions to the users. Works such as [19, 39, 176, 178] inform users about the success or failure of their VisQueries. Moreover, [91, 214, 57, 43, 16] provide the users with informative feedback, additional explanations and follow-up questions to users to carry on the analysis. For example, after creating a decision tree, Ava [91] can ask users if they want to see another plot. Iris [57] and TransVis [16] ask users questions to continue the analysis, such as *"Which column should I use on the x-axis"* and *"What is the recovery time you want to use?"*.

Works such as [108, 175, 81, 174, 178] proposed different informative feedback types. For example, Data@Hand [108] gives users three types of textual feedback: to confirm that it had applied the command to visualisation, to inform users that the command is not valid, and when it fails to understand. Databreeze [175] also has three available textual feedback types: to confirm successful action, after a follow-up command, and after partially understanding a command. Evizeon [81] has five types of textual feedback: (i) when the intent is understood and the result is shown, (ii) when it does not understand the request but the system guesses the nearest operable result, (iii) when the VisQuery is partially understood feedback appears with highlighting the unknown word, (iv) when it understands the VisQuery but cannot find any result, and (v) when it does not understand the intent. InChorus [174] has three different feedback styles, after a successful operation, after completing a successful operation but not having an effect on the visualisation (e.g., asking to sort by date but the data are already sorted by date), and after an invalid command. Orko [178] is the only one that gives informative feedback using speech and it supports giving feedback after successful and unsuccessful commands.

Moreover, Boomerang [116] informs users about the insights of the data and additionally gives answers to direct questions such as *"Is there a correlation between sales and profit?"*. Similarly, ConVisQA [171] gives answers to direct questions such as *"what is the most negative comment?"* while displaying the textual answer with updated visualisation. Moreover, ONYX [154] informs users about the action it has performed and gives instructions to users to teach the meaning of the unknown commands using WIMP. Valetto [95] provides feedback to inform users when there is a misunderstanding and provides additional information to users such as stating the correlation of two attributes. Finally, Talk2Data [169] provides explanations about visualisations for creating narrative storytelling.

Furthermore, we explored related work that provided users with additional visual feedback, such as supplementary graphs with main visualisation or changes on filters on the UI that have been applied by the chatbot. For example, Boomerang's [116] main goal is to show users multiple recommended visualisations related to users' VisQueries on the right-hand side of the screen; however, relevant graphs are also displayed in the chat window when required. FlowNL [84] presents users with an ambiguity widget and has two auxiliary charts, one being a histogram displaying the velocity magnitude of hurricanes, while the other is a 2D map chart that is used to signal to specific regions. Additionally, visualisation is synchronised with a table. ConVisQA [171] visualises a hierarchical structure of comments on the main visualisation that is synchronised with actual comments displayed on the right side of the screen. Moreover, Orko [178] visualises additional charts (e.g., bar) and shows on the user interface whose filters are activated and display widgets in response to VisQueries. Similarly, Evizeon [81] presents related widgets after each VisQuery. Gamebot [214] displays buttons to assist the conversation.

V-NLIs such as [174, 176, 39, 154, 175, 95, 108, 16] have visual feedback on the UI. For example, Data@Hand [108] displays an 'undo' button after every VisQuery; further, the user interface changes according to VisQueries such as displaying related filters. InChorus [174] and Snowy [176] show applied filters on the WIMP; additionally, in Snowy, selected attributes can be seen as well. Filters and attributes shown on the UI are updated after each VisQuery in [39, 154]. Moreover, Valetto [95] highlights the recognised text in the chatbot's UI. For example, when a user asks to *"Add acceleration to the graph"*, it changes the colour of the 'Add' token in the user's sentence. Finally, Talk2Data [169] shows annotations with visualisations, and Chat2Vis [122] titles the visualisations from the users' VisQuery.

Furthermore, we examined the **interaction styles of the output** across the analysed systems. A significant number of these systems support WIMP [116, 171, 108, 175, 81, 84, 214, 174, 39, 154, 178, 176, 169, 16, 95], where filters and sliders dynamically update in response to NL commands. For instance, in Snowy [176], completing a VisQuery not only updates the graph but also adjusts the encoding parameters, such as the x and y axes on the left panel (see Figure 5.4.b). Similarly, in FlowNL [84], when a user requests an attribute through the chat, it is automatically added to the attributes table and assigned a distinct colour, seamlessly integrating into the visualisation (see Figure 5.11.d).

5.4.7 Technology behind V-NLIs

In this section, we briefly explore the software technologies used in the related works. We can distinguish between those that directly use **NLP-toolkits** and those that use **chatbot frameworks**. For the former, we found multiple examples. The most used NLP-toolkit is open-source CoreNLP in Java [124]. For instance, Snowy [176], Miva [39] and Evizeon [81] all use it. Others use CoreNLP in combination with other toolkits, such as [178], which combines CoreNLP with NLTK [20] and AIML [29], and ConvisQA [171], which integrates CoreNLP with an ANTLR parser [143]. Some works use other NLP-toolkits; for example, Valetto [95] uses spaCy toolkit [78]. Finally, Data@Hand [108], which focuses on speech recognition, uses Apple speech framework [9] and Microsoft Cognitive Services [129] for IOS and Android devices, respectively, and Compromise NLP toolkit [101] to perform part-of-speech tagging. Among the V-NLIs that use chatbot frameworks, running independently from the visualisation module, we find: ACUI [19] using Rocket Chat open-source software [151]; Boomerang [116] based on IBM Watson Assistant [1]; GeCoAgent [43] based on Rasa [146]; and TransVis [16] employing Google Dialogflow4 [67].

Moreover, other works proposed customised solutions. Gamebot [214] uses rule-based word matching. Iris [57] uses domain-specific language that transforms Python functions into a finite state machine. Ava [91] employs a state machine to control natural language conversations. FlowNL [84] uses a declarative language to filter and combine data to derive structures and translates natural language VisQueries into declarative specifications to render visualisations. Finally, the latest contributions to the field: Chat2Vis [122] uses LLMs, while Talk2Data [169] uses a novel decomposition model that is extended from sequence-to-sequence (deep neural networks) architectures.

Furthermore, recent advances in chatbot technology, such as ChatGPT-4 [138], and Gemini[188] demonstrate its ability to respond to VisQueries. These advances already started to be applied to the field of data visualisation. For example, users can ask the chatbot to show them a visualisation of a particular layout by sending an image showing the desired layout. In fact, a recent study has focused on creating data visualisations using Natural Language with ChatGPT-3 and GPT-3.5 [122]. The study proposed using Large Language Models (LLMs) to create data visualisations from tabular data with basic visualisation methods. The system is able to select the appropriate visualisation type based on user VisQueries. In another example, ChatKG [40] employs LLM using ChatGPT for analysing temporal sequences. While the study reported positive results, it also recognised certain constraints, such as the difficulty in effectively comparing extensive sets of data simultaneously. It is particularly not suitable for analysing non-temporal or multivariate data and can become visually overwhelming when dealing with numerous patterns simultaneously. Moreover, these advances come with several challenges, such as difficulties in specifying refinements to plotting elements, variability in the type of plot generated, and their non-deterministic nature. Given the fact that none of the works reviewed in this literature review use NL interactions to change symbolic Graphical Elements, such as glyphs and colour palettes, these generative approaches can potentially be used to generate them during analytical conversations.

5.4.8 Design Methodologies

In the previous sections, we explored V-NLI interfaces that contributed to the visualisation chatbot literature. As explored, these works presented V-NLIs in health [108] or business [89] domains, and also delved into specific aspects of VisChatbots such as queries' recommendation [176] and ambiguity resolution [65]. However, it should be noted that these works did not mention general design methodologies focused on VisChatbot, and neither does the literature present contributions in this aspect. However, the creation of a VisChatbot presents numerous challenges due to its multifaceted nature; integrating visual elements into a conversational interface requires careful planning to ensure a seamless user experience; while the specifications need to be broad enough to encompass factors such as target user profile, goals tasks, preferred visualisations, and input/output modalities.

It should be mentioned that, in the field of text-based chatbots, there have been some efforts to introduce design methodologies, following well-known steps in software development but tailored to specific fields like health [30] and education [13]. Additionally, Moore et al. [131] provided the Natural Conversation Framework (NFC) wherein they presented 100 conversation patterns, among other artefacts, to guide the conversational UX design process. The framework proposes the phases of the design thinking methodology [75] for crafting effective chatbot interactions. However, these methodologies fall short of meeting VisChatbots requirements such as being aware of the nature of the data, visualisation types and interaction modalities.

Feine et al, [58] proposed three design principles for interactive chatbot creation. Firstly, they advocated for direct manipulation of objects of interest. Secondly, they stressed contingent responses to engage users further. Lastly, they highlighted the importance of collecting and visualising interaction metrics. These principles were then integrated into specific design features for their chatbot. Although their approach emphasises interactivity, it focuses primarily on chatbot functionality rather than visualisation aspects.

While these works are well-established methodologies for visualisation, HCI, and chatbot design, they are not directly applicable to VisChatbots. Actually, proposing a new methodology for developing visualisation chatbots (V-NLI) is essential because it addresses the unique challenges and requirements that traditional HCI design methodologies and textual chatbot frameworks fail to cover adequately. Although the phases in these methodologies may be similar, the artefacts produced and the steps followed within each phase must be specifically tailored to the context of a V-NLI. HCI methods are primarily focused on user interfaces [131], [30], [58], and did not provide guidelines for integrating NL interaction with complex data visualizations, while general chatbot frameworks are optimised for text or voice communication without addressing the intricacies of presenting and interpreting visual data. That is, the integration of visual elements into a conversational framework demands more than the simple addition of graphics to dialogue— it requires a careful rethinking of how users interact with and explore data through conversation. Existing methodologies fail to account for the multidimensional interaction, where both visual and verbal cues must work together to guide the user through complex data.

In summary, while several studies have explored methodologies and tools for designing text-based chatbots, a significant gap persists in the literature regarding the development of methodologies specifically tailored for VisChatbots. Indeed, VisChatbots introduce unique considerations involving the visual mapping of data (generating or updating visualisations) and the subsequent transformations of these visualisations(zoom in, panning, changing colour, among others). Moreover, designing a VisChatbot requires understanding why users engage with visualisations, their characteristics (demographics, familiarity with visualisations, and the data domain), their objectives, and the complexity of the target tasks extracted from their hypotheses. Additionally, the different nature of the inputs and outputs (text and visual) means putting the focus on the modalities (e.g., natural language, WIMP, sound, touch, gestures) that need to be managed and synchronised by the

VisChatbot. Finally, in this novel paradigm, VisChatbots designers need new proposals for UX evaluation metrics.

5.5 Summary

In the V-NLI literature discussed above, most works focused on **tabular data** with **basic** visualisations, rather than exploring more **complex data** with **advanced** visualisations. This trend continues in the selection of visual mapping techniques, where the emphasis remains on simple data and visualisations. Moreover, they typically allow users to express only **low-level VisQueries**, while those that consider **high-level VisQueries** still do so using simple data types and attributes, such as tabular data with numerical and nominal attributes. Indeed, Talk2Data [169] and Chat2Vis [122] are the only related works that used high-level VisQueries, both with tabular data. However, the former has a **form-based interface**, and although the latter is **VisChatbot**, it lacks chatbot qualities such as **conversational feedback** and viewing conversation history. As a result, a gap exists in the use of natural language for analysing more complex data types, such as network and hierarchical data.

> **Remark 5.1**. *Designing **VisChatbot** should account for **complex** data and **advanced** visualisations that support **high-level** VisQueries.*

Furthermore, when we explored **conversational guidance** strategies (auto-complete, help, recommendation, and follow-up), we came across works that include different kinds of conversational guidance [91, 154, 176, 95, 16, 108, 178, 169, 171, 81, 43, 175], few of them supporting multiple types [176, 91]. Conversational guidance strategies enhance the user experience by helping users navigate the V-NLIs, understand intricate components, can guide them in formulating effective VisQueries, and supporting their data analysis. Therefore, guidance strategies should be thoroughly explored and integrated into V-NLIs to enhance user experience and simplify the analysis process.

> **Remark 5.2**. *Chatbots' **guidance strategies** are crucial in VisChatbot design, as they improve the **usability** of V-NLIs, ultimately enhancing the overall user experience.*

A significant development that can enhance guidance strategies is the implementation of a passive listening mode, which allows the chatbot to observe conversations between users and respond accordingly [19]. In line with this, a recent study explored an always-listening agent that acts as a third collaborator in a multi-person visual analysis. The agent generates visualisations based on observations it makes from users' conversations [185].

Generally, V-NLI research examines various aspects of VisChatbots, such as widgets for managing ambiguity [65] [163], intent recommendations for data analysis [176], automated chart selection for accessibility [169] [116], and multimodality (touch and gestures) to improve user experience [95] [174]. However, these studies frequently **lack a structured design methodology**, complicating the integration of user needs, natural language understanding, and conversational flow. This lack of methodical planning can lead to inefficiencies, inconsistencies, and challenges in delivering a cohesive and effective VisChatbot solution. Although some toolkits, such as NL4DV [133] and ncNet [119], facilitate V-NLI creation, they fail to guarantee their robustness and efficacy, as revealed by Feng et al. [59].

> **Remark 5.3**. *There is a notable gap in comprehensive* ***design methodologies tailored specifically*** *for* ***VisChatbots****. Addressing this gap is essential for developing effective and cohesive VisChatbots.*

5.6 Conclusions

In this Chapter, we brought together the fields of data visualisation and chatbot-based interaction to study the body of literature on Visualisation-oriented Natural Language Interfaces (V-NLIs). Our aim was to provide an overall picture of the current state of V-NLIs, especially VisChatbots, and to identify and highlight future research directions. We provided a summary of the aspects that are currently focused on and supported by V-NLIs, as well as their limitations. Specifically, the limitations found are related to the complexity of the analysed data, advanced type visualisations, the type of queries supported by the chatbot, the lack of visual mapping automatisation, and the supported interaction styles. Moreover, after this thorough review, we emphasise that these V-NLIs focused on only specific aspects and did not explore a comprehensive methodology for creating VisChatbots. In the following chapters, we will use this information as guidance to develop a comprehensive methodology for designing VisChatbots and for creating a VisChatbot for our visualisation platform, DViL using this methodology.

Chapter 6

VisChat Methodology

With the exponential growth of data from diverse sources, effective data analysis tools have become increasingly crucial. V-NLIs, also known as VisChatbots, have emerged as promising solutions to address this challenge. As explored in the previous section, numerous research efforts have significantly advanced the literature on VisChatbots. These studies have applied V-NLIs in various domains, including health [108] and business [89], and have investigated specific aspects such as query recommendations [176] and ambiguity resolution [65].

However, as stated in Remark 5.3 in Chapter 5, these works have not addressed design methodologies specifically focused on VisChatbots, and the literature lacks contributions in this area. Indeed, creating a VisChatbot involves numerous challenges due to its multifaceted nature. Integrating visual elements into a conversational interface requires meticulous planning to ensure a seamless user experience, while also addressing broad specifications such as the target user profile, goals, tasks, preferred visualisations, and input/output modalities. In the broader context of conversational user interfaces, some research has proposed natural conversational frameworks [131] [30], while others have outlined principles to guide chatbot design [58].

To tailor these design frameworks to the context of VisChatbots, in this chapter, we propose a new Visualisation Chatbot methodology (VisChat), which aims to guide the incremental creation of VisChatbots for visual analytics processes. This iterative process includes the three main classical phases of a development process: Analysis, Design, and Development. In our case, however, each phase introduces various stages that deal with specific characteristics of VisChatbots.

6.1 Methodology

VisChat Methodology is inspired by the well-known Behaviour Driven Development (BDD) from agile software development [172], which is especially suitable when a project is complex. BDD ensures that the software meets the customer's precise requirements and breaks down complex requirements into smaller, manageable pieces (features, scenarios, and use cases). Particularly, we focus on describing VisChatBot behaviour through use cases and encouraging collaboration among stakeholders (users, designers, and developers). Our VisChat methodology proceeds through three phases - Analysis, Design and Development -, where the **Design** phase is the centre of the process on which the **Analysis** and the **Development** phases will pivot in successive iterations, not only ensuring the VisChatbot's design evolves cohesively with the demands of users but also empowering continual enhancements in its design. Figure 6.1 shows these phases, depicting the **Design** phase as a big turquoise square, and the **Analysis** and **Development** phases as blue and green circular shapes, respectively. Circular arrows cross the main **stages** (coloured small squares) of each phase, allowing for revisiting and iteration over phases.

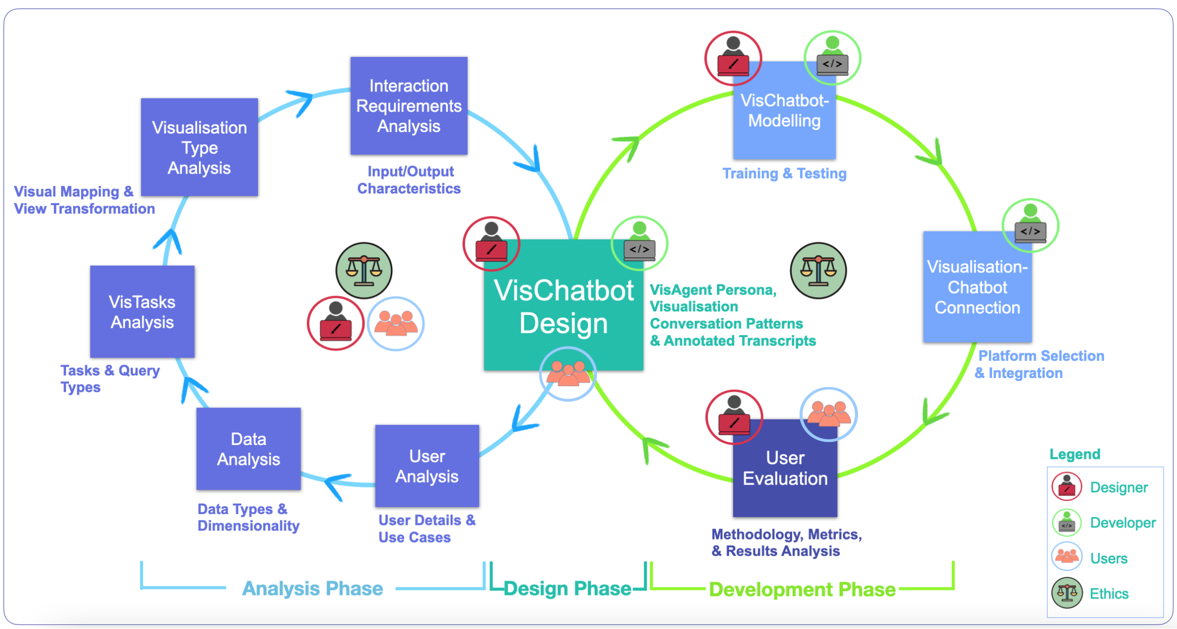


Figure 6.1: The 3 Phases of VisChat Methodology. Squares indicate stages within phases. Icons show Ethics requirements and stakeholders' roles (see the legend on the right bottom side). Role icons at the centre of a circle indicate the roles involved in all stages.

Each iteration incorporates one use case, i.e., a research hypothesis, to be analysed along the visual analytics workflow. Each research hypothesis involves several elemental VisTasks. For example, in a visual analytics process dealing with urban mobility patterns, researchers formulate the research hypothesis: *"The larger the population in an area, the more pronounced the peak hours of traffic congestion"*. Then, the first iteration incorporates into the design two tasks: (1) an initial exploration of the data which is to get familiarise with the visualisation and gain a general understanding of it (first VisTask), and (2) discovering patterns inside the data which is to test the hypothesis (second VisTask), ending with an evaluation stage to ensure that the VisChatbot can assist users in the validation of the hypothesis. This evaluation allows for gathering users' feedback to refine either the requirements of the VisChatbot (see blue circle in Figure 6.1), or VisChatbot's design (see green circle in Figure 6.1). Finally, after addressing all the refinements, the design is prepared to integrate a second research question into the visual analysis.

The design of a VisChatbot requires the collaborative efforts of key participants: users, developers, and designers. In the **Analysis Phase**, users and designers collaborate to gather and evaluate user needs, with designers focusing on capturing detailed requirements and preferences related to users, their data, and the visualisation platform on which the analysis will be performed. During the **Design Phase**, all three roles are actively involved: users offer additional details as needed based on insights from the analysis phase, while designers refine the design using the collected information to balance user needs with technical feasibility. Designers work closely with developers to ensure the design can be effectively implemented. In the **Development Phase**, collaboration persists as users engage in testing and provide feedback, developers turn the design into a functional system, and designers oversee the integration to ensure the final product aligns with both the initial vision and user expectations. The specifics of the roles will be explained in each phase below.

Moreover, it should be noted that during the design of a VisChatbot, we recommend the consideration of ethical and privacy issues throughout all phases, especially in the stages of user analysis, data analysis, in the generation of Annotated Transcripts in the VisChatBot Design —where the fundamental logic and decision-making processes of the chatbot are established, ensuring ethical interaction between the chatbot and the user—, and during the user evaluation stage. This means double-checking data sources (authenticity and reliability), avoiding bias both in the data and in the visualisations by preserving sufficient context, encouraging feedback for transparency, respecting privacy rights and obtaining consent before collecting personal data during testing, among others. The following sections introduce each phase of the VisChat methodology (Analysis, Design and Development), describing the stages involved in each of them.

6.1.1 Analysis Phase

The aim of this phase is to investigate users' requirements (use cases), describe data characteristics to comprehend its structure, perform the VisTask analysis, and set the visualisation demands and interaction needs. Note that a VisChatbot is naturally aware of the textual conversation, but it also has awareness of the visualisations, the data it manages and the user interface (UI) in which it is embedded.

Gathered insights will be used to guide the design of the VisChatbot, ensuring it aligns with users' requirements for seamless data exploration and interpretation. This phase encompasses the following stages: User, Data, VisTask, Visualisation Type, and Interaction Requirements Analysis (see Figure 6.2).

In this phase, users and designers collaborate to create a user-centric design. Specifically, users actively collaborate with designers by sharing valuable insights regarding their objectives, preferred visualisations, and data interaction methods. This input allows designers to better understand user needs and preferences. In turn, designers synthesise this information to inform the next phase of development, ensuring that the resulting chatbot design effectively supports users' goals and tasks while seamlessly integrating with the visualisation platform.

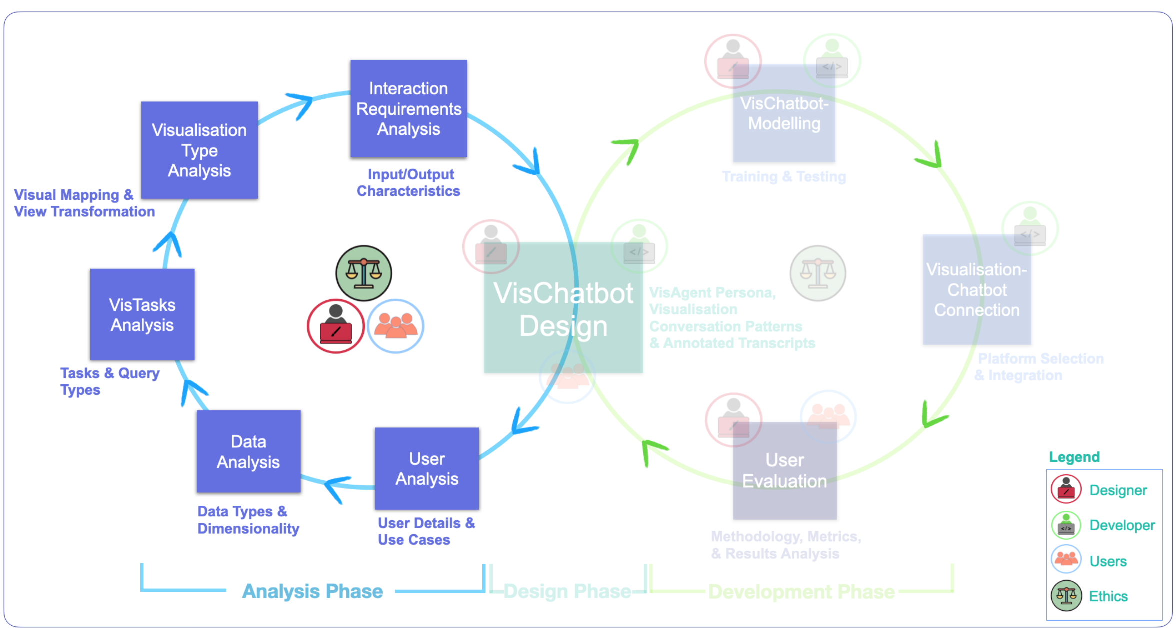


Figure 6.2: Analysis Phase of the VisChat Methodology

User Analysis Stage

This stage involves a comprehensive exploration of the target audience. The primary focus is on acquiring knowledge regarding the users and their goals with the conversational agent.

In the HCI field, there are general user research methods to emphasise with users and gather data from them (observations, interviews, questionnaires) [113], and others based on analysing the visual analytic activity [68]. When it comes to chatbot design, Wizard of Oz studies (mimicking a chatbot) is the most suitable method [12]. Collected data such as users'**demographics and background** will undoubtedly affect the vocabulary and the jargon of VisChabot's design. For example, if users are children who are learning data visualisations, they might require designing a different jargon than adult users who are analysing a company's sales [70].

Moreover, as users interact with the VisChatbot using natural language, it is crucial to know their **language proficiency** in order to ensure the effectiveness of the conversation. For example, English may not be the users' mother tongue and they may not have the grammatical mastery required to talk to the chatbot. Finally, understanding the extent of **users' expertise** regarding the data domain, visualisation terminology, and the confidence needed to manage the UI helps determine the appropriate level of support provided by the VisChatbot.

In summary, in this stage, designers gather information about the target user profile, including their goals, demographics, background, language proficiency, and expertise with data and visualisations. These data will inform design decisions in the Design Phase.

Data Analysis Stage

This stage explores the characteristics of the data the VisChatbot will deal with. The literature provides a myriad of data types for visualisation [170], although for the sake of simplicity here we consider two main **Data Types**: tabular and complex data as we introduced in the Chapter 2, Section 2.1. Indeed, user queries are simpler with tabular data (e.g., requesting to plot a bar chart or select the peak point) than with unstructured or complex data, which requires more sophisticated visualisation (e.g. network layouts), and consequently, longer conversations and more difficult language understanding are needed to interpret user queries. This situation worsens when dealing with high **multidimensional data**, which brings more aspects to analyse, including complex relationships between features.

Whatever the case, whether the data is simple or complex, users need to prepare and restructure raw data into a format suitable for visualisation. Thus, designers can consider that NL can help to improve the accessibility of average users to perform data transformation and facilitate intuitive visualisations [85].

Another aspect is to determine the vocabulary used during a VisTask. On the one hand, the chatbot's designers must define general terms commonly used in analytics (e.g. *"Are there outliers?", "Which category has the lowest values?"*, etc.). On the other hand, specific jargon of the data domain should be included in the VisChatbot's vocabulary (e.g., sports, healthcare, business).

To sum up, in this stage, designers should document key information about the target data type and domain. This includes clearly defining the data domain, specifying the type of data, and recording the relevant jargon and terminology. For example, if dealing with hierarchical data, designers need to ensure the VisChatbot understands terms like "thread," "depth," or "node." Similarly, for other types of data, the chatbot should be able to comprehend relevant terms, such as "trends", "axis", "frequency", and "time range" for time series data, or "category", "values", and "trends" for categorical data.

VisTasks Analysis Stage

At this stage, it is essential to understand what motivates users to interact with the VisChabot (the 'why' of usage). This involves analysing the VisTasks that are useful when solving a specific research hypothesis. The selection of the VisTasks will depend on the research hypotheses and goals. To do so, we leverage the main **VisTasks Types** presented in the Chapter 2, Section 2.1: LookupT (known Location and Target), LocateT (unknown Location and known Target), BrowseT (known Location and unknown Target) and ExploreT (unknown Location and Target). Moreover, we introduced VisQueries; IdentifyQ, CompareQ, and SummariseQ which are utilised to execute sequences of VisTasks. By the end of this stage, designers compile a set of VisTasks and associated VisQueries necessary to validate their research hypotheses. The known/unknown location/target labels of VisTasks, along with the types of VisQueries, will inform the selection of suitable VisChatbot's input/output modalities during the Interaction Analysis stage.

Visualisation Types Analysis Stage

This stage determines the kind of visualisations required to analyse the data regarding the **Visual Mapping** and the **View Transformation** operations. As presented in the DataVis Pipeline in Chapter 2, Section 2.1, **Visual Mapping** refers to visual aspects such as visualisation layouts, glyph, size, and colour meanwhile **View Transformation** is concerned with operations like zoom, panning, multiple views, and focus+context [34] [167] [96].

Different charts and diagrams can be used in the **Visual Mapping** stage, depending on the VisTask and the kind of data. For instance, with bar charts or line charts, users analyse trends in data, while with hierarchical layouts or network diagrams, the focus shifts to exploring paths and the relationships between different data points. Moreover, when dealing with complex data, there exists a wide variety of layouts that can help the users to perform their VisTasks, though in some cases one layout can be more informative than others as explained in Chapters 2 and 3. In this context, the VisChatbot can automatically select the most appropriate layout for each visualisation [176, 81, 97, 122, 121]. For example, Mackinlay et al. [121] used a rule-based method to automatically decide the best way to display the data in a chart. For example, in the case of one quantitative attribute and one categorical attribute, the system recommended a bar chart.

Additionally, colour and glyphs are essential visual elements for an effective visual analysis. Nevertheless, choosing the right ones can be challenging for novice users. Novice users may struggle to select the most effective colours and glyphs to best present the data and to arrange elements in a way that clearly communicates the insights, potentially leading to confusing or misleading interpretations of the visualisations. Therefore, designers of the VisChatbot should consider implementing features for the automatic generation of colours and glyphs [168], tailored to achieve the desired visual outcomes.

Moreover, **View Transformations** like zooming or panning, influence how the visualisation is displayed. Thus, designers need to decide whether the VisChatbot should be aware of these changes. Alternatively, they can also decide to empower the VisChatbot with the ability to automatically locate the view on the most relevant areas in response to a user query. Finally, note that based on these **Visual Mapping** and **View Transformation** analyses, designers specify the VisChatbot's intents, terminology, and types of additional output. Additional outputs will be elaborated upon in the next section.

Interaction Requirements Stage

This stage is responsible for analysing the required VisChatbot's Input/Output characteristics. Notice that a VisChatbot is embedded in a three-faceted interactive ecosystem that is characterised by diverse user interactions and dynamic data visualisations: WHERE do users interact: in areas such as menus, canvas 2D/3D, and text boxes; HOW do users interact: using multimodal interactions (i.e., text, sound, gestures); and WHAT is their interaction style: WIMP, AR, VR. Therefore, a good VisChatbot should be able to handle information from and to these different sources (see Input and Output in Figure 6.3) to make users feel like it all works together seamlessly.

Across this ecosystem, the VisChatbot's inputs are complex, since users may interact through **low** or **high** level VisQueries using: NL, pointing to objects in the 2D/3D canvas, or selecting GUI widgets, among others. As outlined in Chapter 5, Section 5.3, **Low-level** queries are usually very specific, and, in most cases, can be solved using NL and also directly through the GUI. In contrast, **high-level** queries are so broad and varied that they are not manageable all at once by the GUI

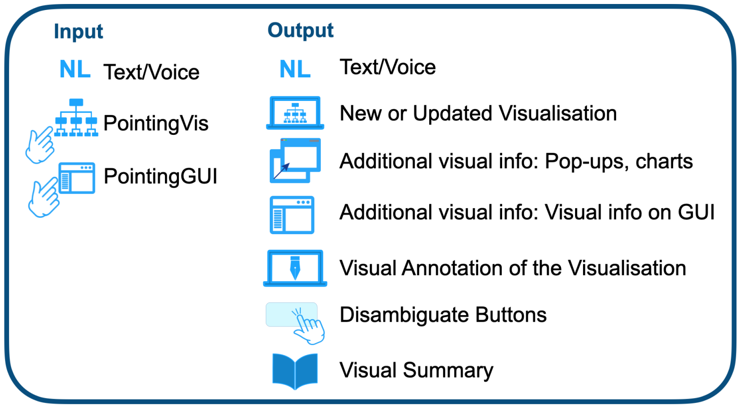


Figure 6.3: Input and Output modalities of a VisChatbot.

nor predictable. Therefore, they require the use of various input modalities (e.g., NL and canvas interaction) that should be preserved in the context of the conversation.

Given the complexity of user-chatbot interactions and the need for multiple input modalities, the designer must analyse the VisQueries defined in the VisTasks Analysis stage to specify the most adequate inputs to be maintained in the VisChatbot context and also to be considered in the **Design Phase**. By maintaining this information in the context, the follow-up conversations can refer to pronouns or co-references to previous text-based and visual inputs. For instance, if the user signals a 3D object in the visualisation, afterwards she can refer to it in the conversation saying *"Tell me more about it"*.

In addition, **Guidance**—such as auto-complete, recommendations, suggested queries, and hints—, and **Proactivity**, to suggest to users the necessary steps to complete an analysis, are relevant VisChatbot characteristics to be considered and are closely related to the type of VisTask. Indeed, **Guidance** and **Proactivity** are relevant characteristics in LocateT, BrowseT, and ExploreT tasks as either the location, the target, or both are unknown. For example, imagine a user performing a LocateT task through the IdentifyQ query, analysing online conversations, *"Show me the most positive thread"*. A proactive chatbot decides an automatic zoom-in to help the user to find and see important details, of the unknown location.

The analysis of VisChatbot's output characteristics is also driven by the VisChatbot's interactive ecosystem since there are multiple ways to enrich the **Feedback** such as through additional charts, annotated visualisations, statistics, and even haptics in VR, among others (see the right-hand side in Figure 6.3). That means the chatbot can give responses such as **a New or Updated Visualisation**, and visual feedback (i.e., **View Transformations** using zoom and **Visual Mapping** like colour changes). It can also confirm or inform about the answers using **Natural Language** and **Additional visual info**, such as displaying supplementary charts and highlighting widgets in the GUI. Additional outputs are **Visual Annotations** in the updated Visualisation, the **Disambiguate Buttons** in the chat, and **Visual Summaries**.

Finally, note that supplementary charts generate **Multiviews**. Thus, the chatbot should be able to differentiate which view the user is referring to. Actually, **Multiviews** are closely related to **CompareQ** queries since they assist users in contrasting different parts of a visualisation or two different datasets.

In summary, the designer must define the VisChatbot's input and output characteristics, including multiviews, multimodality, and also user interactions the VisChatbot is intended to support. We emphasise the connection between VisTasks and their interaction requirements, particularly how guidance and follow-up skills relate to different VisTasks. For example, tasks like LookupT, where both the location and target are known, generally require less guidance. In contrast, more complex tasks, such as LocateT, BrowseT, or ExploreT, may necessitate additional support and sophisticated interactions. Thus, after this stage, the designers should have the complete set of required interaction methods.

6.1.2 Design Phase

Once the requirements of the VisChatbot have been defined in the previous phase, in this phase, the VisChatbot design is conceptualised, adhering to design principles for conversational interactions: **Recipient design**, **Minimisation**, and **Repair** [131].

Recipient design refers to the way that speakers shape or structure their conversation in various ways depending on the specific individual with whom they are talking. Therefore, designers of VisChatbots should adjust the vocabulary based on user characteristics, such as expertise level. They should also offer multiple conversation paths to ensure users with varying levels of expertise can navigate efficiently. For instance, experts may require shorter interactions to accomplish VisTasks than novice users.

The **Minimisation** principle reflects real-world conversational norms in which individuals aim to use concise language. This principle should inform VisChatbot design, in the sense that text can be minimised both in terms of how it is expressed and by utilising complementary visuals and WIMP interactions. For instance, users can make shorter queries while pointing to elements in the visualisation, such as asking for more details about a specific element.

The last principle, the **Repair** principle, similar to conversational strategies like repeating or paraphrasing, aims to rectify misunderstandings or ambiguities. For instance, highlighting specific parts of the visualisation can clarify the user's intent.

Based on these principles, our methodology proposes three design artefacts: (1) VisAgent Persona, (2) Conversational-Visualisation Patterns, and (3) VisChatBot Transcripts (see Figure 6.4). The VisAgent Persona aids in designing the characteristics of the chatbot according to the target users. The Conversation-Visualisation Patterns help to design the interaction between the user and the VisChatbot. From these, designers can define the VisChatBot Transcripts, which are instantiations of the pattern and will inform the deployment phase of the VisChatbot interactions. These transcripts will be the basis for defining the agent's training data.

In this phase, collaboration involves designers, developers, and users. Initially, designers will complete the VisAgent Persona using information collected from users. If any details are missing or require confirmation, users will participate to provide the necessary input. In the second design artefact, designers and developers will work together to create and refine conversation patterns. Finally, the VisChatbot Transcripts will be developed collaboratively by designers and developers to ensure they align with the established conversation patterns and effectively support user interactions. Next, we will present and discuss each of the proposed design artefacts in detail.

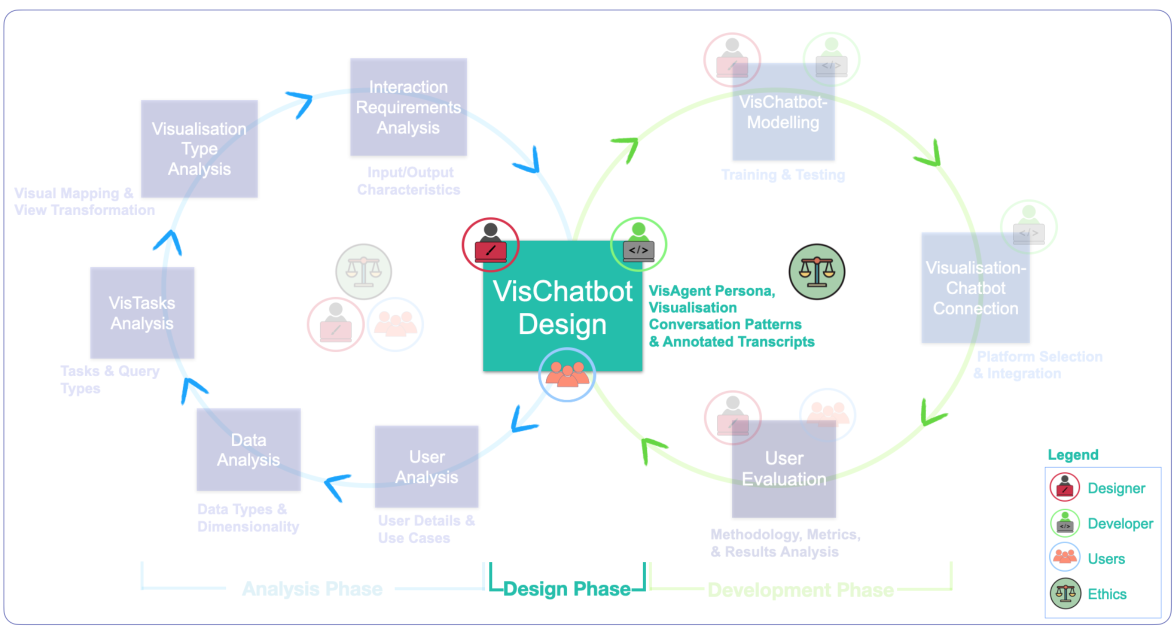


Figure 6.4: Design Phase of the VisChat Methodology

Artefact 1: VisAgent Persona

User Persona artefacts are widely used in classical HCI research [93], and Moore et al. [131] suggested the need to define a Conversational Agent Persona for text-based chatbots. Following these ideas, we propose a Conversational Agent Persona template called VisAgent Persona that describes an archetype of the chatbot incorporating both conversational aspects and the visual dimension of VisChatbots, providing a consistent view of the intended users (see Figure 6.5). The template has six key sections outlining general descriptions: **Goals**, **Traits**, **Target Visualisation**, **Target Interactions**, **Target Users** characteristics, and **Skills** of the VisAgent, which is filled according to the outcome of the **Analysis Phase**.

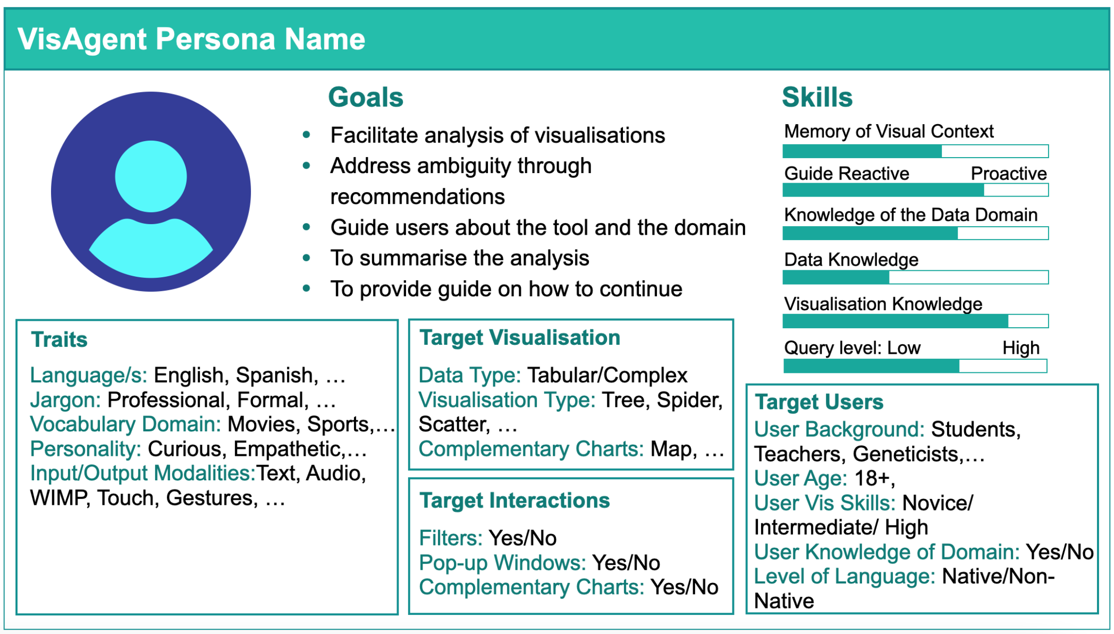


Figure 6.5: VisAgent Persona Template. Adapted from [93]

In Figure 6.5, the **Goals** section outlines general objectives that designers must customise to the specific VisChatbot instance, using the information collected from User Analysis, Data Analysis, and Task Analysis stages in the Analysis Phase. For example, the goal "*Guide users about the domain*" in data visualisation in the music domain would be "*Give the users info about their preferred artists and songs*". **Traits** specifies the characteristics that the VisChatbot must possess, such as the language, jargon, and vocabulary of the chosen domain, the personality of the VisChatbot, and more importantly, input/output modalities. These are the information gathered from the User and Data Analysis stages as well as, the Interaction Requirements Analysis stage.

The interaction with these modalities is specified in the **Target Interactions** section, which refers to the possible interactions with the GUI's buttons, together with complementary charts which are collected from the Interaction Requirements Analysis stage.

Regarding new sections in the proposed template that are closely related to VisChatbots, **Target Visualisation** includes data types, the types of the main visualisation, and the types of the complementary charts, among other aspects related to **Visual Mapping** and **View Transformation**, which are collected from Visualisation Type Analysis stage. In addition, **Target Users** focuses on users' background (e.g., students, biologists, etc.), age range, visualisation skills, data domain knowledge and level of expertise in VisChatbot's language, which are gathered from the User Analysis stage.

Finally, the **Skills** section demonstrates the expected capabilities of the VisChatbot, including its ability to retain and recall visual context changes or interactions during conversations, ensuring continuity and relevance which is called Memory of Visual Context. Moreover, it includes Guidance (reactive or proactive), Knowledge of the Data Domain, Data and Visualisation knowledge, and Query level (**low** or **high**). This information is collected from various Analysis stages such as Interaction Requirements Analysis and VisTasks Analysis.

Artefact 2: Visualisation Conversation Patterns

The second artefact, Visualisation Conversation Patterns are predefined templates that specify user-chatbot interactions. These patterns provide a starting point for designers and developers so they do not have to reinvent the basic mechanics of conversational structures. They emerge from stakeholder brainstorming and the study of visual analytical conversations and are influenced by the defined VisAgent Persona.

Notably, the NCF framework proposed by [131] described conventional text-based conversational patterns which address the primary goals of user interactions such as inquiring about details, requesting repetition, and aborting conversations. Drawing inspiration from these, we introduce generic patterns tailored for VisChatbots. Our attention now turns to VisTasks and their possible input/output modalities that enrich the conversation.

We showcase all the **Visualisation-Conversation Patterns** through pattern cards displayed in Figures 6.6 to 6.13. Each card introduces a pattern name (e.g., Guidance Tool) alongside its specific objectives and an outline of its intended purposes. We present dialogue patterns in sequential steps, demonstrating interactions between users and VisChatbot, and giving an example for each pattern. As there are different input and output styles in a VisChatbot, in all the patterns, these types are represented by icons (see Figure 6.3).

We organise our patterns into three families depending on their purpose:

- The **Visualisation Patterns** family refers to those patterns that aim to define the interactions of the VisChatBot in VisTasks and VisQueries. In this family, we propose two patterns: the **Request Visualisation Task and Query** and the **Request Visualisation Details** (see Figures 6.6 and 6.7, respectively).

- The **Assistance Patterns**, which aim to guide users in the analysis. In these patterns, we focus on the VisChatbot's capability to observe user conversations and suggest effective interaction methods to aid in task completion, as described in Remark 5.2. This remark highlights the importance of VisChatbots' guidance strategies in improving the usability of V-NLIs, ultimately enhancing the overall user experience. Additionally, these patterns address ambiguities in user communication, where potential vagueness or unclear expressions can lead to misunderstandings. Thus, we propose two proactive patterns in this family: **Sniffer**, and **Ambiguity Helper** (see Figures 6.8 and 6.9, respectively). Moreover, another key aspect of guidance focuses on helping users navigate visualisations and data domains, thereby enhancing their ability to derive meaningful insights. Therefore, we present two reactive patterns: **Guidance Tool**, and **Guidance Domain** (see Figures 6.10 and 6.11, respectively).

- Finally, the **Flow Patterns** family focuses on the thread of the analytical conversations. In this family, we specifically focus on how VisChatbots can assist users in summarising data analysis findings while ensuring the insights are clear and unbiased. Thus, we propose: **Visual Summary** (see Figure 6.12). Additionally, we present **Closer Invite to Continue** (see Figure 6.13) to offer users a smooth transition to further engagement or to gracefully end the conversation.

Pattern 1: Request VisTask and Query

Pattern 1 allows users to interact with visualisations through VisTasks (LookupT, LocateT, BrowseT, ExploreT) and queries (IdentifyQ, CompareQ, SummariseQ). As illustrated in Figure 6.6, this pattern involves the following steps: the user requests a visualisation task and query, the VisChatbot responds with the relevant visualisation and query results, and optionally, the VisChatbot may recommend further actions [1]. This pattern supports input via text/voice and

[1] Step 3 is optional and indicated by brackets [ ].

PointingVis, and output through text/voice, updated visualisations, additional visual details, and visual annotations. As shown in the example provided in the figure, the user asks to see the country with the highest number of players for the game Witcher. The VisChatbot responds by updating the visualisation to highlight Australia on the map and annotating it with both a visual marker (the bar chart showing the top four countries) and a text label displaying the exact number of players. Additionally, the VisChatbot states textually that the top-ranked country is Australia and explains that the bar chart shows the top four countries. Following this, the VisChatbot offers a recommendation by suggesting an additional analysis, asking if the user would like to explore data trends for Australia over time, and illustrating the optional recommendation step.

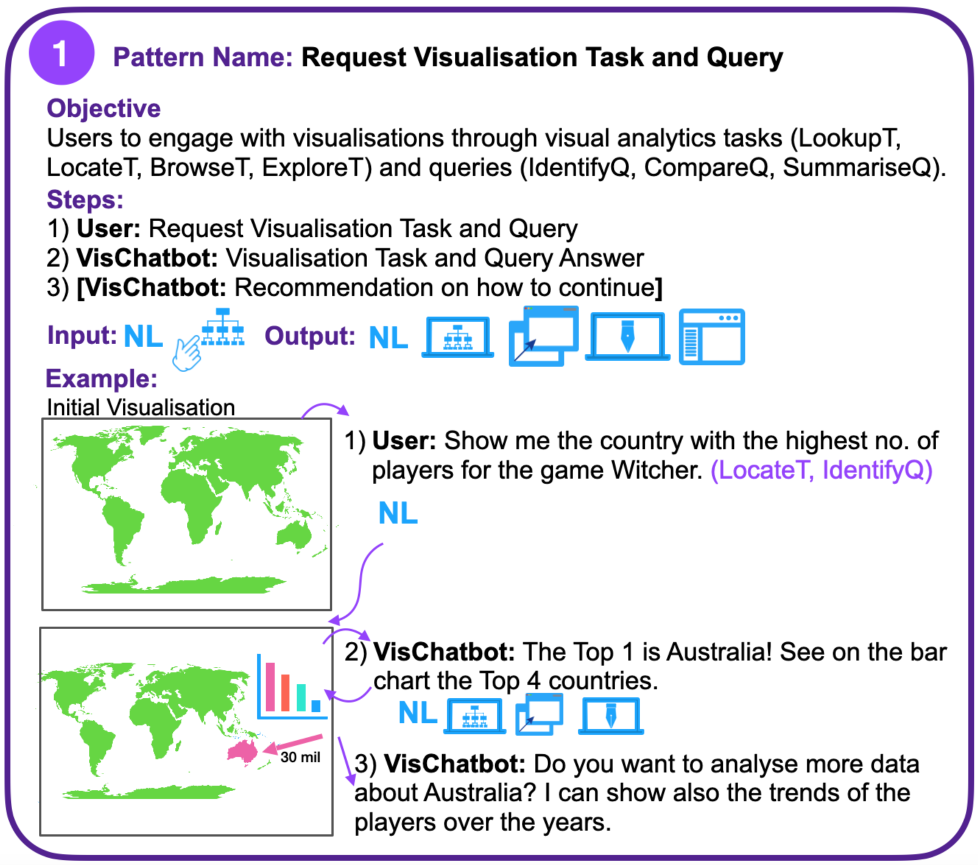


Figure 6.6: Request VisTask and Query Pattern from Visualisation Patterns Family.

Pattern 2: Request Visualisation Details

Pattern 2 focuses on enhancing visualisations by allowing users to request finer details such as zooming, panning, adjusting colours, resizing, and more (see Figure 6.7). The steps in this pattern include: the user requesting specific visualisation details, the VisChatbot responding with the requested adjustments, and optionally, the VisChatbot suggesting further actions [2]. This pattern supports input via text/voice and PointingVis, and outputs through text/voice, updated visualisations, additional visual details, and visual annotations. This pattern continues with the Witcher example from Pattern 1. The user requests to see the country with the fewest Witcher players, and the VisChatbot responds by zooming in on Sri Lanka, presenting a more detailed view with visual annotations such as labels identifying the player count which is 1000. Additionally, it provides further insights with a bar chart highlighting other popular games in the country, such as Sims and FIFA.

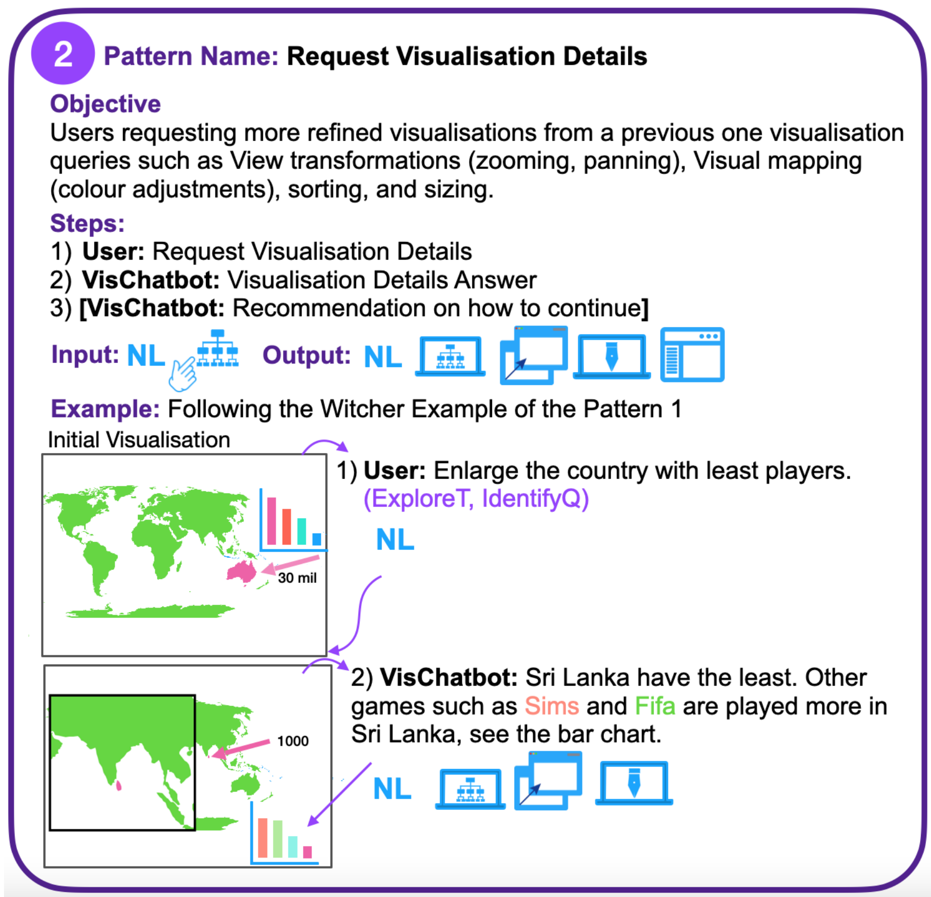


Figure 6.7: Request Visualisation Details Pattern from Visualisation Patterns Family.

[2]Step 3 is optional and indicated by brackets [ ].

Pattern 3: Sniffer
Pattern 3 is intended to observe users' interactions within both the visualisation and the VisChatbot. For example, as users explore or navigate over the canvas, and users can type and delete text in the chat, in these cases the VisChatbot remains attentive, ready to offer additional support or guidance as needed. This pattern includes input of text/voice and PointingVis, and text/voice output. In the example, the user seems lost moving the mouse on the canvas, without a concrete direction but mostly centred around China or writing but deleting text continuously. The VisChatbot asks whether the user is looking for China or information about it and states that it can provide additional details, such as demographics in China

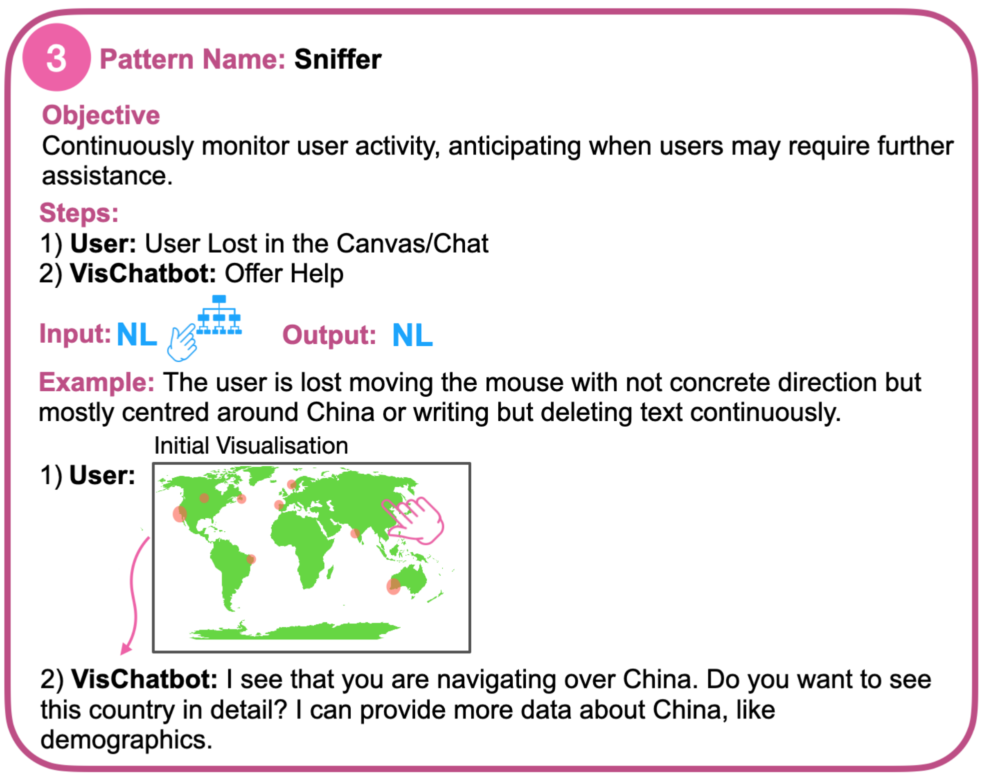


Figure 6.8: Sniffer Pattern from Assistance Patterns Family Proactive, i.e., observe users' interactions within both the visualisation and the VisChatbot.

Pattern 4: Ambiguity Helper

Pattern 4 aims to guide users through disambiguate queries to assist with the analysis when the VisChabot has some doubts about the user's intents (see Figure 6.9). This pattern involves four steps: 1) the user requests a visualisation task and query, 2) the VisChatbot offers disambiguation interactions, 3) the user selects a response, and 4) the VisChatbot provides a final answer or further disambiguation. The pattern supports text/voice and PointingVis input, with outputs including text/voice, updated or new visualisations, additional visual information, and disambiguation buttons. Continuing from the Sniffer pattern example, the VisChatbot suggests exploring China. The user hovers over China in the visualisation and asks, "*Tell me about that*" referring to the country. The VisChatbot detects ambiguity and responds with a clarifying prompt: "*Do you want to explore the players by Age or Gender?*" (displaying clickable disambiguation buttons). After the user selects an option (e.g., "Age"), the VisChatbot generates further information, such as a new visualisation (e.g., a spider chart), along with textual data explaining that the top players in China are aged 19.

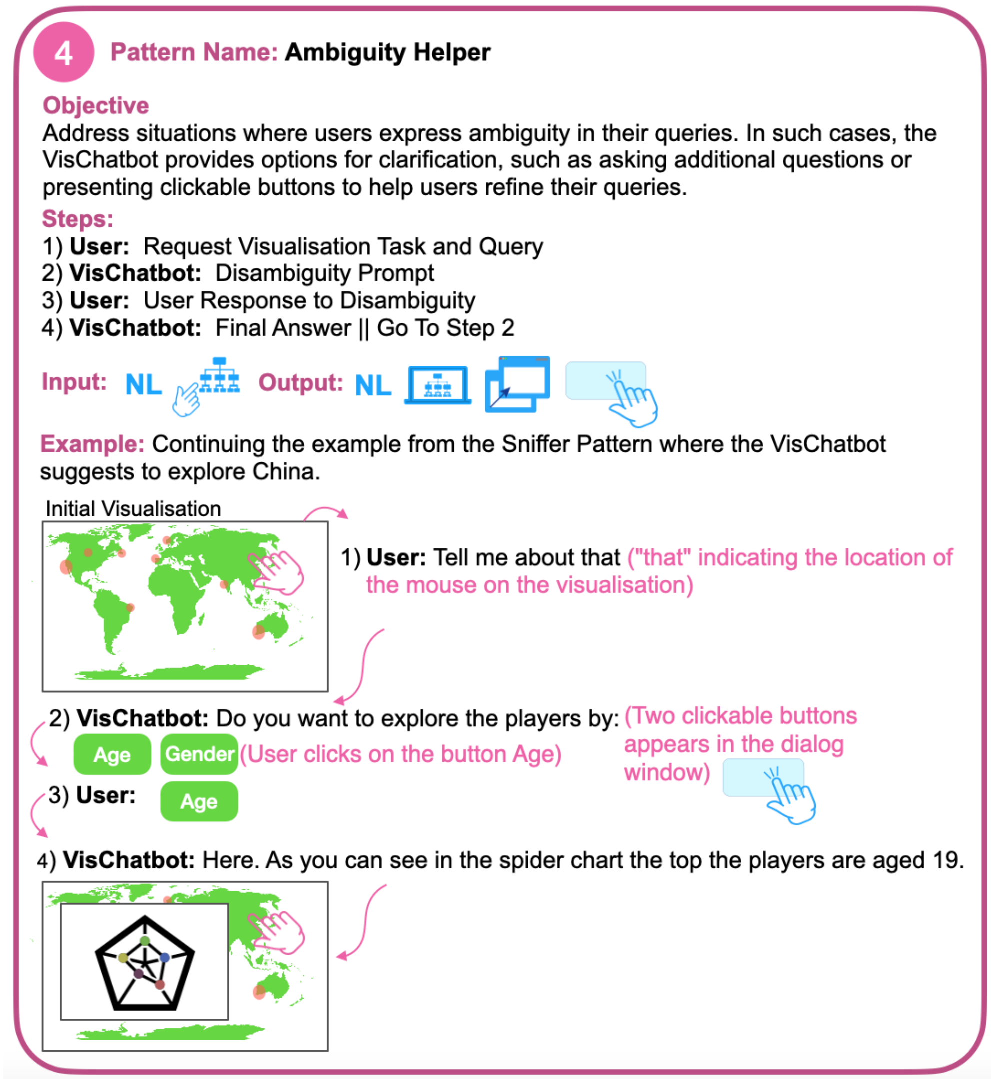


Figure 6.9: Ambiguity Helper Pattern from Assistance Patterns Family Proactive.

Pattern 5: Guidance Tool

Pattern 5 helps users with queries regarding the visualisation tool providing guidance through the GUI. This pattern includes two steps; the user's intent to request guidance on the tool and a guidance tool response. It accepts text/voice and PointingGui as input and provides text/voice, additional visual information, and visual annotations as output. In the example in Figure 6.10, , the user points to the Filters on the GUI and asks, "*What else can I analyse?*" The VisChatbot responds by transitioning the interface into tutorial mode. During this mode, the GUI highlights the available filters (such as RPG, Action, and MMO), displaying additional visual elements like charts and offering an explanatory narration. For example, the VisChatbot may highlight a bar chart displaying the statistics of different game genres played over the past year, guiding the user through the filter options interactively. The speech playback can be paused or resumed at any time.

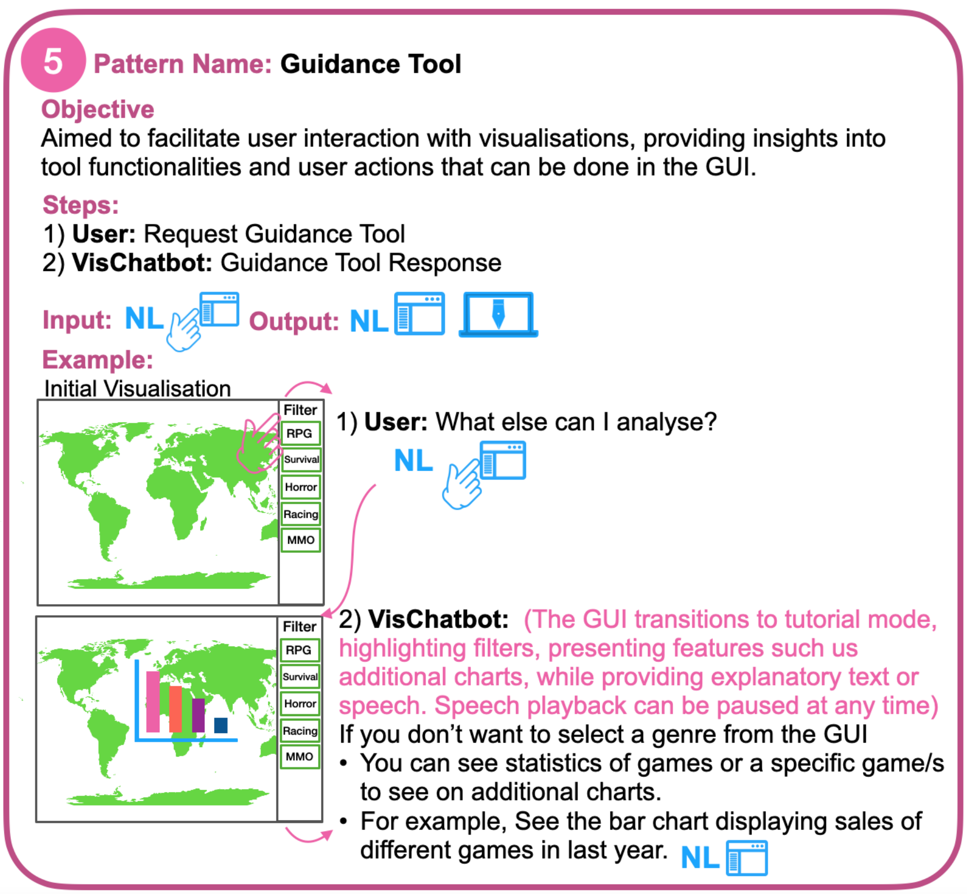


Figure 6.10: Guidance Tool Pattern from Assistance Patterns Family Reactive.

Pattern 6: Guidance Domain

Pattern 6 helps users with queries about the specific vocabulary and concepts implied in the application being analysed. This pattern involves the user requesting domain guidance and the VisChatbot providing a response. It accepts text or voice input and outputs through text, voice, and additional visual information on the GUI. Returning to the example of analysing data from the game Witcher, it can be observed in Figure 6.11, the user asks about the game's plot. In response, the VisChatbot provides a video trailer of the game, which can be played or paused by the user at any time. Alternatively, the VisChatbot could present the plot as text instead of a video. Additionally, the GUI highlights relevant information (e.g., genre) and prompts the user with a follow-up question: "*Do you want to explore other games with the same type of genre?*"

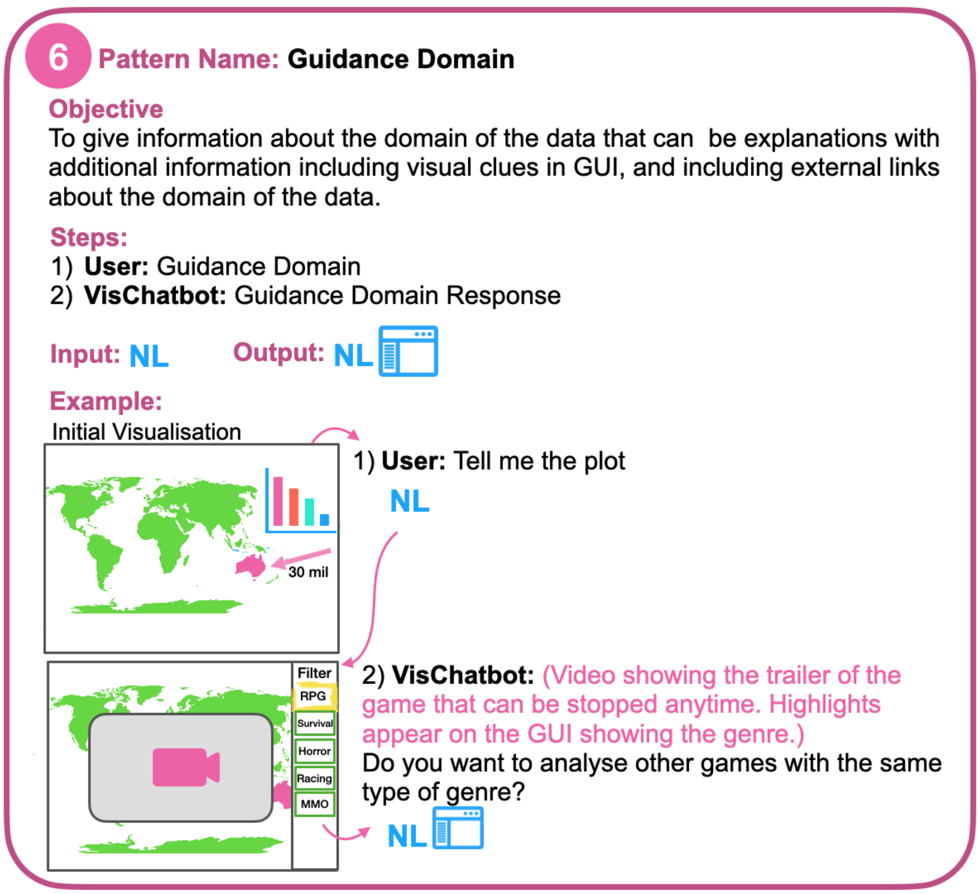


Figure 6.11: Guidance Domain Pattern from Assistance Patterns Family.

Pattern 7: Request Visual Summary

Pattern 7 gives users a visual summary of key points from their analysis with annotations under the frames (see Figure 6.12). This pattern also includes two steps; the user requesting a visual summary and getting a visual summary response from the VisChatbot. It includes text/voice as input and output, as well as, visual summary output. Here we can see in the example that the user requests a summary of their session, and VisChatbot provides a visual summary with annotations. The annotations in the summary include highlighted key actions from the user's interaction, such as asking about the most-played country, zooming into specific countries, and exploring the map to analyse other regions.

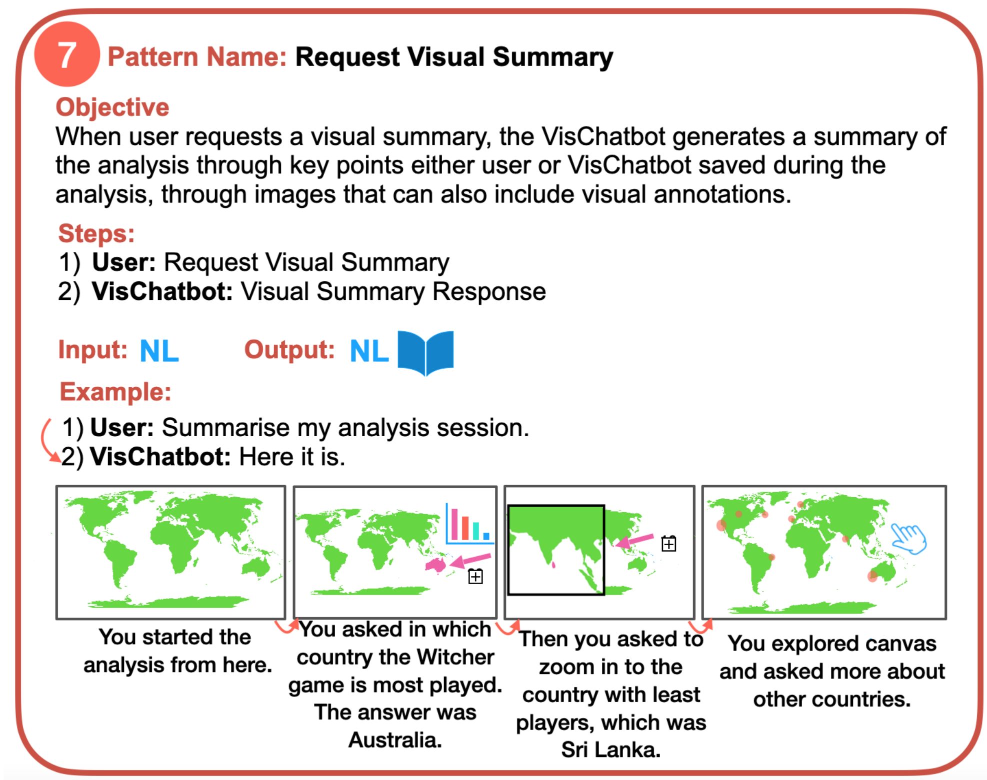


Figure 6.12: Request Visual Summary Pattern from Flow Patterns Family.

Pattern 8: Closer Invitation to Continue Pattern

Pattern 8 monitors user activities to recommend steps to close the current analysis and start a new one (see Figure 6.13). This pattern involves four steps: 1) The user completes the current VisTask, 2) The VisChatbot invites the user to continue with another VisTask, 3A) If the user accepts, 4A) the VisChatbot suggests a query to proceed, 3B) If the user declines, 4B) the VisChatbot concludes the session. This pattern accepts text/voice and PointingGUI as input and provides text/voice and additional visual information on the GUI as output. In the example, after the user closes some pop-up windows to finish their analysis, the VisChatbot asks if the user would like to see anything else before quitting. If the user responds affirmatively, the VisChatbot offers a query for further exploration. If the user declines, the VisChatbot responds with a polite farewell, ensuring a smooth and respectful conclusion to the session.

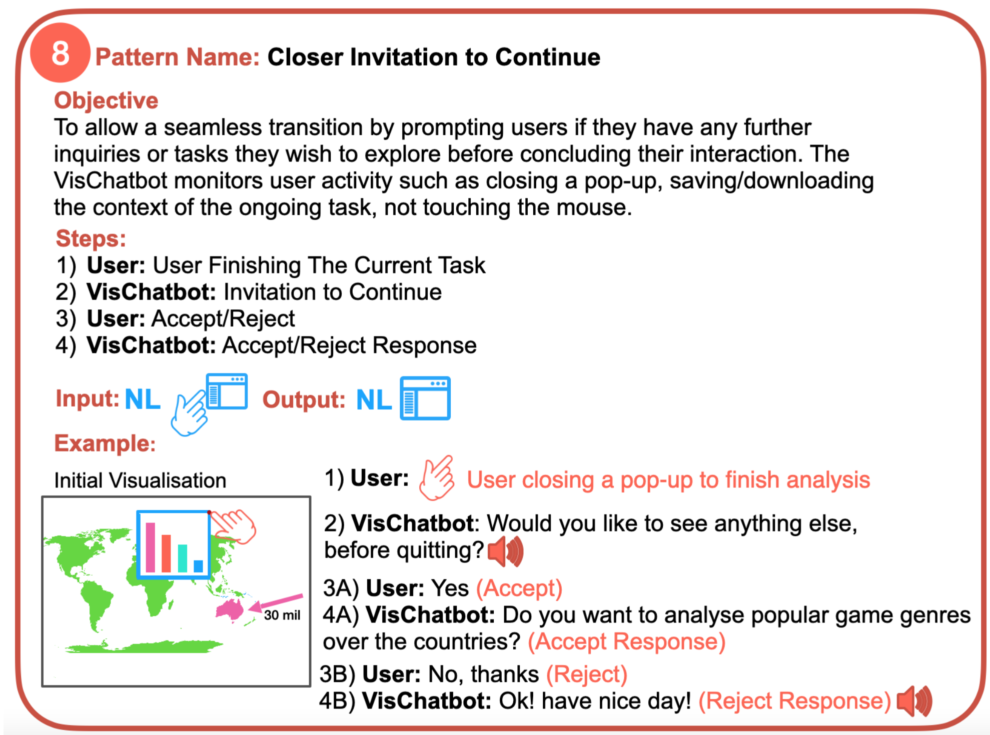


Figure 6.13: Closer Invitation to Continue Pattern from Flow Patterns Family. 3A-B depicting the conversation when the user is affirmative, and 4A-B when user is rejecting. Speaker indicates that VisChatbot prompted user with an alert.

It should be emphasised that these patterns can be combined to form larger, more complex patterns, allowing for a mix-and-match approach in any order. Additionally, the conversational text-based patterns proposed by the NFC framework [131] can be included among the VisChat patterns, such as: requesting repetition, expressing gratitude, or aborting conversations, and patterns that help participants navigate the beginning and end of interactions, facilitating control over the overall engagement state. These patterns are agnostic to any type of task or query, offering flexibility and adaptability to suit various interaction scenarios. For example, consider the requesting repetition pattern used when the VisChatbot encounters unclear or unfamiliar user input. If a user types, "**Show me the stats for players in the game**", but the VisChatbot's programmed terminology is more specific, like "**visualise player statistics**", it may not immediately grasp the request. The chatbot, recognising the confusion, would then ask for clarification of the user intent to ensure it provides the correct visualisation.

Artefact 3: Annotated Transcripts

Usually, transcripts are text-based lines specifying user-chatbot dialogues, for instance, "User:" and "VisChatbot:" steps in Figure 6.13. In the case of VisChatbots, we extend the lines of the transcript with **side notes** related to visual and interactive aspects of the conversation. Regarding VisChatbot's lines, for instance, a note may indicate that an additional chart appears along with the main visualisation and simultaneously with the text response. Concerning the user's lines, a note may point out that an ellipsis in the text refers to an object in the visualisation (for example, "it" refers to the location at which the mouse is pointing). This third artefact is based on the selected Visual Conversation patterns, considering the VisAgent persona. That is, each pattern is used to guide the generation of transcript lines of conversation examples, aligned with the defined VisAgent Persona (e.g., examples with child vocabulary if the users are 6 years old).

Transcripts will make up the VisChatbot training data in the Development phase. Notice that the **side notes** with visual aspects are an add-on in the VisChatbot's context. That is, the VisChatbot is able to remember specific points or areas previously interacted with or selected, as well as interactions with GUI elements such as filters and buttons. Also, the side notes in transcripts are the key points to detail all the VisQueries related to the data types, visualisation types, visualisation transformations, and interaction requirements identified in the **Analysis phase**. See the side notes in Figure 6.14, highlighting the important key points that should be implemented by the developer in the development phase. In the example, we categorised each user query into input (what the chatbot will detect) and output (what actions the chatbot should take to fulfil this user query). The input involves processing natural language understanding (NLU) and visual comprehension. It includes notes about recognising the user's request, such as identifying key terms (e.g., "*country with least players*"), extracting contextual data (e.g., the current dataset and visualisations), saving information (e.g., Sri Lanka as the country with the fewest players), and interpreting user interactions, such as mouse movements, identifying locations (e.g., China). The output is natural language generation (NLG) and visual generation. In the output, notes are divided into updated visualisation (e.g., show the country with the least players), additional visual info notes (e.g., a complementary bar chart displaying statistics about other games, like Sims and FIFA), visual annotations (e.g., displaying the number of least players on the map), and textual explanation about the output. These notes will guide developers in systematically programming user intents, VisChatbot responses, visual transformations, and interactions. Note that the three artefacts presented in this section (VisAgent Persona, Visualisation Conversation Patterns and Annotated Transcripts) are agnostic of the conversational technology, which is selected in the next phase of the methodology.

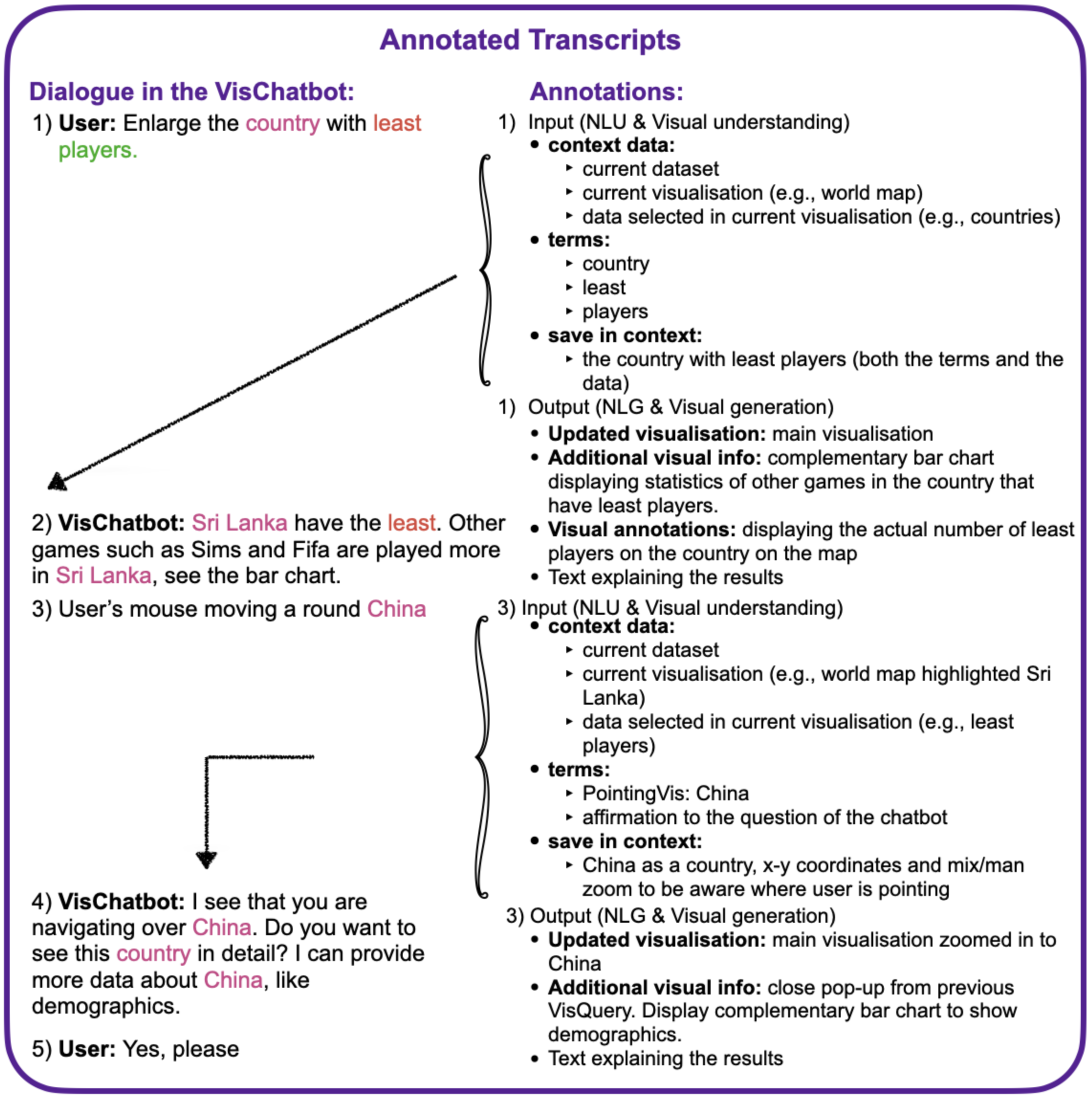


Figure 6.14: Including examples from Pattern 2 and Pattern 3, side notes illustrating Annotated Transcripts.

6.1.3 Development Phase

In this phase, the VisChatbot is developed according to the three design artefacts. Next, we detail the three stages: **VisChatbot-Modelling**, **Visualisation-Chatbot Connection**, and **User Evaluation**. The first two stages are developer-oriented while the last stage involves designers and users (as indicated by icons on the right-hand side of Figure 6.15). In the V**isChatbot Modelling** stage, the developer consolidates all the information and artefacts from the Design Phase crafted by the designer. Here, the designer can assist the developer by clarifying design intentions and ensuring that all elements are implemented correctly. During the **Visualisation-Chatbot Connection** phase, the developer focuses on integrating the chatbot with the visualisation platform, ensuring seamless interaction between the two systems. Some VisChatbots also display some visualisation output directly in the chatbot window, so this integration should account for both embedding visualisations within the chat interface and handling external visualisation displays smoothly. In the **User Evaluation** phase, both the designer and users collaborate once again. Users participate in testing the VisChatbot and provide feedback, while the designer collects and analyses this feedback. This data is crucial for refining the system and preparing for subsequent iterations. In the following, we explore these three stages.

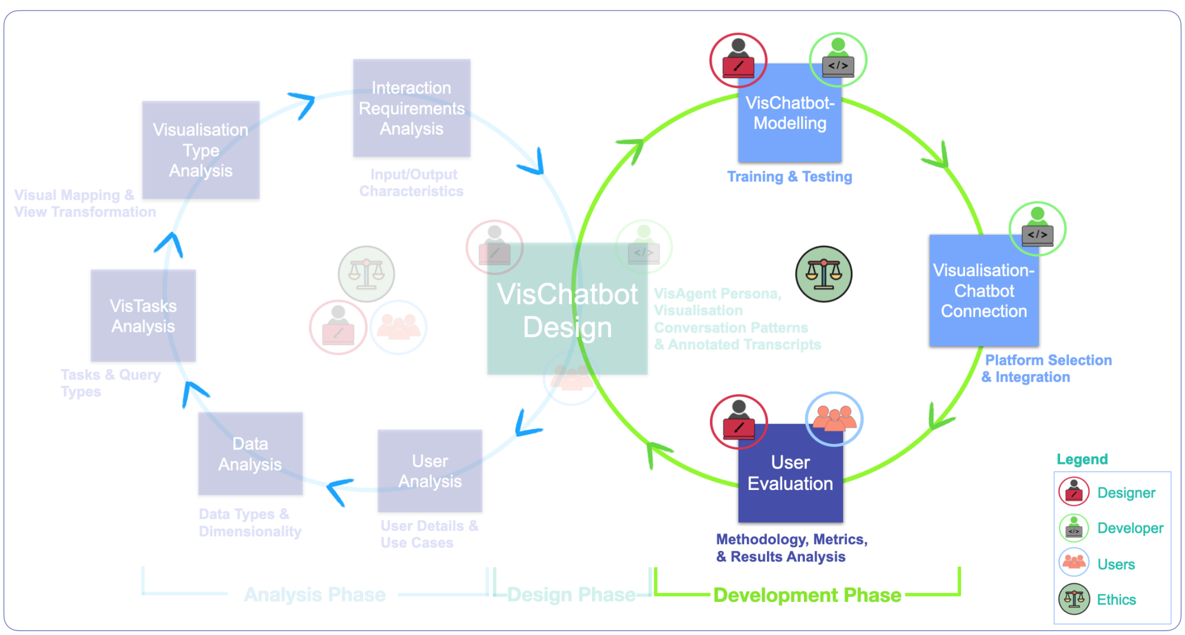


Figure 6.15: Development Phase of the VisChat Methodology

VisChatbot Modelling

This stage creates the conversational model for the VisChatbot. At this stage, it is therefore important to decide on the technology to build it. On the one hand, information-retrieval technologies [52] search for relevant information from a predefined set of transcripts in response to user queries. They can be based on rules, such as AIML, ChatScript, or based on machine-learning strategies such as Rasa, Microsoft's Bot Framework, Google's DialogFlow, IBM Watsonx, and AmazonLex. On the other hand, generative technologies [125] produce new content or responses based on learned patterns and models, rather than retrieving pre-existing information, as is the case with ChatGPT and Gemini.

In information-retrieval technologies based on Machine Learning, the VisChatbot model undergoes a process of training and refinement using the generated Transcripts, equipping it with the knowledge to effectively respond to user inputs in data visualisation. In the generative approaches, a fine-tuning approach may incorporate additional training data through the Annotated Transcripts. All these technologies involve a testing step to validate the accuracy of the generated model.

Visualisation-Chatbot Connection

The goal of this phase is two-fold: (i) the identification of the details of the platform on where the VisChatbot will be integrated, and (ii) the integration of VisChatbot itself.

The platform details, including the **Modalities** defined in the Interaction Requirements stage and the types of visualisations specified in the **Analysis phase** (see Section 6.1.1), must be carefully documented and addressed in this stage. For instance, the VisChatbot should be connected to the defined visualisations to facilitate dynamic interaction and real-time data updates. Moreover, if the platform is web-based, factors such as menus, filters, and navigation bars must be taken into account to ensure smooth communication between the VisCatbot and the user interface. If the platform is a mobile app, considerations should include touch interactions and screen orientation. In Extended Reality (XR), it is essential to consider how users interact with 3D environments, through spatial interactions.

Second, the most costly part is the integration of the VisChatBot model in the platform. In this aspect, **side notes** in the annotated transcripts are crucial to define the bidirectional VisChatbot-platform communication. This communication facilitates the synchronisation between the conversation and the visualisation, as well as preserving not only textual but also visual information in the VisChatbot conversation context.

User Evaluation

This is the last stage of VisChatbot methodology. It aims to test the quality of user—Vischatbot communication –such as NL understanding, the appropriateness of responses, and visualisations' aesthetics—as well as general usability dimensions such as effectiveness, efficiency, error management and satisfaction.

Recall that our methodology incorporates a use case in each iteration, starting from the User Analysis and progressing until the User Evaluation. However, when we deal with the evaluation of one use case, we perform an iterative testing limited to the **Development Phase** (see the green circle in the right part of Figure 6.15). The first test helps designers to identify user actions and agent responses that were not anticipated, and therefore developers should feed the VisChatbot Modelling and Visualisation-Chatbot Connection stages. Then, subsequent tests focus on refining the chatbot model based on how users express themselves, adding variations of existing intents, entities and synonyms.

Concretely, we propose an evaluation that follows a well-defined methodology that establishes: (1) evaluation goals and metrics; (2) criteria to recruit participants; (3) VisTasks that users will perform; (4) location and the necessary equipment; and (5) data security and privacy policies. Finally, (6) data is analysed based on established metrics to propose improvements, and if necessary, go back to previous stages of our VisChat methodology. Since goals, and especially metrics, are different to the evaluation of other interactive systems, we think they deserve the thorough analysis that we present next.

Similarly to other contexts, when evaluating the UX with a VisChatbot, the goals are to ensure the experience is effective, engaging, easy to use, and aesthetically pleasing. Regarding metrics, VisChatbots have unique features because the users may naturally interleave natural language (NL) queries with interactive manipulations on the canvas/visualisation, and they are designed to deal with particular aspects of the visualisation pipeline. Thus, we first propose special metrics for VisChatbots, and afterwards, we also highlight the standard metrics used in the evaluation of text-based chatbots [148] [149] all of which are shown in Table 6.1 grouped by the evaluation goals. Note that some metrics appear in multiple rows of the table because they measure different aspects of the UX. The proposed VisChatbot metrics are:

- Success Rate of Multimodal Queries: This metric measures the ability of the VisChatbot to deal with different input/output modalities simultaneously. For instance, to understand when users employ both NL and interactions, such as gestures, in canvas in their queries.

- Multimodality Use at Pattern Level: Measures the types of input/output modalities (text, voice, mouse, gestures, haptics) used in the steps of conversation patterns.

- Pattern Completion Rate (PCR): It is the percentage of completed steps of a Visualisation Conversation Pattern. This metric is defined at a finer granularity than the TCR (Task Completion Rate) used in general chatbots.
- Success Rate of data/visualisation queries: Measures how correctly the VisChatbot interprets the user's data and visualisation needs and generates a relevant and accurate visualisation.
- Appropriateness of created/updated visualisations: Assess whether the user gained the intended insights or learned something new from the visualisation generated by the VisChatbot.
- Appropriateness of additional visual output: Measures how well the complementary visual output chosen by the VisChatbot (e.g., bar chart popups, hierarchical chart) communicates the query results.
- Visualisation score: Measures the visual appeal and overall design quality of the visualisations.

On the other hand, we propose to maintain also the chatbot metrics used in the literature [35] [189]:

- Average Conversation Length (AvCL): Measures user engagement with the conversational system.
- Retention Rate: Measures the percentage of users who return to interact with the chatbot after an initial session.
- Task Completion Rate (TCR): The percentage of tasks that the users completed. This metric is closely related to the level of mutual understanding since misunderstanding may lead to non-completed conversations (and, consequently uncompleted tasks).
- Understanding Ratio: The number of queries successfully understood by the chatbot over the total number.
- Interactional Efficiency: The number of turns needed to complete a task. This measure informs designers about the efficiency of the conversation by comparing the expected minimum number of turns to the actual number of turns taken by the user to complete the tasks, expressed as a ratio of the minimum to the actual turns.
- Response Error Rate (RER): Measures the percentage of times the chatbot provides inaccurate, irrelevant, or nonsensical answers to users' queries.

Table 6.1: Suggested metrics in relation to each evaluation goal.

Evaluation goals	**Metrics**
Quality and effectiveness of the visual analytic conversation	Success Rate of Multimodal Queries Multimodality Use at Pattern Level Pattern Completion Rate (PCR) Task Completion Rate (TCR) Response Error Rate (RER) Understanding Ratio Appropriateness of visualisation Appropriateness of additional visual output
Aesthetics of the visualisation	Visualisation score Additional visual output score
Engagement	Average Session Lenght (AvCL) Retention rate
Usability	Task Completion Rate (TCR) Response Error Rate (RER) Number of Help Requests Interactional Efficiency Customer Satisfaction Score (CSAT)

- Number of Help Requests: Measures the frequency of users seeking help or guidance from the chatbot.
- Customer Satisfaction Score (CSAT): Measures user satisfaction with the chatbot on a rating scale.

All these metrics are gathered by logs, likes/dislikes during the experience, usability questionnaires, focus groups, and expert evaluations. Moreover, automating the collection of these metrics is crucial to minimise any disruption to the user experience while still ensuring that the data gathered is accurate and reliable.

6.2 Conclusions

In this chapter, we introduced the **VisChatbot Methodology**, an iterative approach guiding VisChatbot development through the Analysis, Design, and Development phases. The Analysis phase encompasses User, Data, Tasks, Visualisation Type, and Interaction Requirements analyses. This phase aims to collect essential information beforehand, which will then be used to shape the design phase.

In the Design phase, we proposed the VisAgent Persona and Visualisation Conversation Patterns. Additionally, we introduced Annotated Transcripts to enrich the visual conversational context. The **VisAgent Persona** offers several benefits, including providing a comprehensive framework that integrates both conversational and visual aspects of the VisChatbot. It ensures a consistent and user-centred design by detailing goals, traits, and interactions, while also specifying the VisChatbot's skills and capabilities. This structured approach aids in aligning the VisChatbot with user needs and interaction requirements, streamlining the development process, and ensuring a coherent and effective user experience.

Regarding the eight **Visualisation Conversation Patterns** provided, not only establish a **solid baseline for VisChatbot design but also demonstrate flexibility and adaptability to multiple scenarios**. Moreover, they can serve as guides to identify key points for collecting logs associated with the evaluation metrics. Also, it should be noted that this set of patterns can grow to address new challenges.

In relation to **Annotated Transcripts**, although they are more complex than other text-based documents [131], they have **served as valuable documentation in both the VisChatbot-Modelling and the Visualisation-Chatbot connection** phases. Indeed, we proposed a preliminary version of side notes that address concrete visual aspects of the conversation such as a complementary chart, or a visual cue, which can be enriched with more sophisticated interactions, such as gestural poses, eye movements, and collaborative experiences [142].

Additionally, we also proposed a comprehensive set of both user-centred and chatbot-centred metrics to evaluate the usability, effectiveness, and engagement of the VisChatbot design. However, the existing literature lacks **dedicated usability questionnaires for VisChatbots**, presenting an ongoing challenge for the development and validation of such tools. Finally, we would like to highlight the social implications of the **VisChatbots** designed using our methodology. On the positive side, they **can democratise access to complex data analysis tools**, making them more accessible to visual analysts without specialised technical skills. However, there are also potential **negative implications, such as concerns about ethics, data privacy and security**. Chatbots interacting with sensitive data may raise questions about who has access to this information and how it is being used, potentially leading to privacy breaches or the misuse of conversational data. In the next chapter, we demonstrate a use case of the VisChatbot Methodology by utilising it to design a VisChatbot for our platform, DViL, introduced in Chapter 4.

Chapter 7

DViL Chatbot: Analysis, Design, and Development

In this chapter, we present a case study related to hate speech analysis where we use our VisChat Methodology introduced in the previous Chapter, to design a VisChatbot named DViL (Data Visualisation in Linguistics) Chatbot. We describe a first iteration through all the phases of our methodology. This iteration included one use case that allowed linguists to explore their research hypotheses.

7.1 DViL Chatbot: Analysis Phase

In this phase, we conduct analyses of Users, Data, VisTasks, Visualisation Types, and Interaction Requirements to gather essential information. We will then use this information in the subsequent Design phase.

User Analysis Stage
Our primary users are linguistics who annotated hate speech in comments on online news articles. The main goal of the VisChatbot is to facilitate the visual analytics of their annotated comments. The linguists are adults (students and professionals) and most do not have prior experience in the data visualisation field, nor English is their mother tongue.

Given their limited experience with data visualisation, the VisChatbot must be intuitive and user-friendly, requiring minimal technical expertise. The VisChatbot must provide clear guidance and support to help users navigate and interpret complex visualisations effectively. Additionally, efficiency is paramount, as users need to quickly analyse large datasets of annotated comments.

Moreover, linguists aim to analyse two specific hypotheses based on their annotated corpus. The first hypothesis explores: *"Not all argumentative messages have to be constructive, but constructive ones are typically non-toxic."* The second hypothesis states: *"Intolerance is a prominent feature in toxic subtrees, with stereotypes being most prevalent at higher levels of toxicity"*. Thus, we focus on having a VisChatbot that is designed to solve their hypotheses.

Data Analysis Stage

The data is hierarchical and multivariate, consisting of a set of news articles and their related comments that refer directly to the news or replies on other comments, forming threads as presented in Chapter 4, Section 4.1. Each comment has 13 annotated features such as argumentation, constructiveness, sarcasm, insult, and intolerance, among others, and a level of toxicity (i.e., having four labels ranging from "Non-Toxic to "Very-Toxic"), described in detail in Section 4.1.2. Ten of these features are abstract and nominal, and three of them, such as the target person, are concrete and nominal. Moreover, levels of toxicities are considered ordinal. Note that all these concepts define the specific jargon of the hate speech VisChatbot.

VisTask Analysis Stage

This case study focuses on two main VisTasks (VisTasks-1 and VisTasks-2), each with subtasks (A and B) derived from the two hypotheses that linguists aim to validate during the visual analytics process, which were stated in the User Analysis stage. We specifically considered two different types of VisTasks (defined in Chapter 2, Section 2.1), to explore these hypotheses, LocateT and ExploreT, both *IdentifyQ* since both aim to spot data points/values (see Table 2.1). Next, we detail the VisTasks and provide the rationale for using LocateT and ExploreT.

- VisTask-1A (i.e., LocateT): *"In how many comments do we see Argumentation and Constructiveness together?"*
- VisTask-1B (i.e., LocateT): *"What is the most common level of toxicity for comments tagged with the feature Constructiveness? Are there other features that appear for this level of toxicity?"*
- VisTask-2A (i.e., LocateT): *"What is the most common level of toxicity for the feature Stereotype?"*
- VisTask-2B (i.e., ExploreT): *"What is the most common level of toxicity for the feature Intolerance in the most toxic subtree?"*

Although the target in VisTasks-1AB is known (Constructive, Argumentation, and Levels of Toxicity) users cannot know the location of the comments that include these features until they interact with the GUI or with the VisChatbot since the features are not directly shown on the canvas. Therefore, it is a LocateT task. Similarly, VisTask-2A follow the same LocateT approach. Otherwise, VisTask-2B, which is an ExploreT task, identifies trends in the data, i.e., the most common level of toxicity of a feature (unknown target) in the most toxic tree, which is not directly visible in the current visualisation (unknown location).

Visualisation Type Analysis Stage

In the following, we analyse the visualisation types that the VisChatbot will deal with. We introduced our data visualisation platform DViL (more details were presented in Chapter 4, Section 4.4) where the VisChatbot aimed to be integrated along with all the visualisations and functionalities. For visual mapping, the VisChatbot will handle the visualisation layouts: Tree, Force, Radial, and Circle. The VisChatbot should be able to recognise the names of these layouts, as it will need to switch between them. Since determining the layout that best suits the topology of the hierarchy can be a complex decision, the VisChatbot automatically should also decide the initial layout based on the characteristics of the data, using the categorisation presented in Chapter 3. The VisChatbot will be able to visualise additional subtrees that cannot be visualised in the DViL platform using the WIMP. Thus, the automatic categorisation will be applied to the subtrees as well, as the hierarchical structure of the subtree can be different from the main visualisation. The VisChatbot should be able to understand to display different parts of the hierarchy, such as the largest thread, and the most toxic thread.

Moreover, as our data is tagged with various features (e.g., sarcasm, constructiveness, and mockery, among others), each comment (or node) can visualise these features using different glyphs. Thus, the VisChatbot is required to recognise vocabulary related to these features to ensure they are accurately visualised using glyphs. In the visual mapping, the chatbot will also handle visualising complementary charts, such as bar charts and pie charts to show the statistics of the tagged features. Regarding the View Transformation, these complementary charts and subtrees will be displayed in pop-up windows. The VisChatbot should be able to understand the context of this and respond to relevant user queries related to these pop-up windows as well.

Interaction Requirements Stage

The DViL chatbot provides the users with a multimodal experience through text-based, speech, and mouse-based interactions, all synchronised with the visualisations. That is, the DViL chatbot is endowed with text/voice and PointingGUI as inputs and both text-based and additional visual outputs, as shown in Figure 7.1.

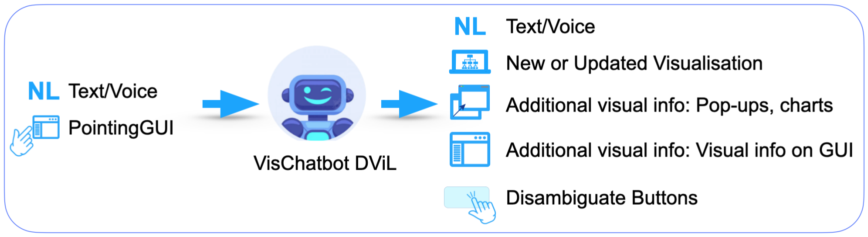


Figure 7.1: DViL VisChatbot's input and output, as defined in Figure 6.3.

Moreover, our VisChatbot is capable of maintaining the context of a follow-up conversation addressing not only the user's previous queries (text co-references) but also input interactions within the platform (GUI co-references). It incorporates both *low-level* queries (simple, doable in one step) and *high-level* queries (more abstract and complex ones, which cannot be performed using only the WIMP). The latter usually require additional visualisations, or multi-view to visualise statistics or subtrees on separate pop-up windows.

7.2 DViL Chatbot: Design Phase

In this phase, we present the three artefacts; VisAgent Persona, Visual Conversation Patterns, and Annotated Transcripts that were filled with the information from the Analysis Phase to inform the design of the VisChabot.

Artefact 1: VisAgent Persona

Figure 7.2 outlines the main characteristics of the DViL chatbot persona, designed specifically to support adult linguists working with hate-speech data. Since users are generally non-native English speakers with limited experience in data visualisation, the chatbot adopts a professional yet accessible tone, understanding even imperfect grammar to facilitate smoother communication. The VisChatbot's primary goals include assisting users in the complex analysis of annotated hate-speech data and providing guidance on tool functionality, and the data domain. A notable goal is its ability to "save key moments of the visualisation," enabling users to save specific insights or views that they can revisit later.

Moreover, the VisChatbot's skill set should include the capacity for storing visual context, allowing it to recall previously viewed visualisations and provide consistent, context-sensitive assistance. Also, the VisChatbot should have a through understanding of the data, data domain, and target visualisations and interactions. It should be designed with a query level that can handle both low and high level visualisation queries, allowing it to answer straightforward questions and guide users through complex visual analyses. To enhance this capability, it can suggest complementary charts or provide pop-up visual aids to make intricate visualisations easier to understand. Furthermore, it has a supportive and friendly personality to emphasise the use of help in the chatbot.

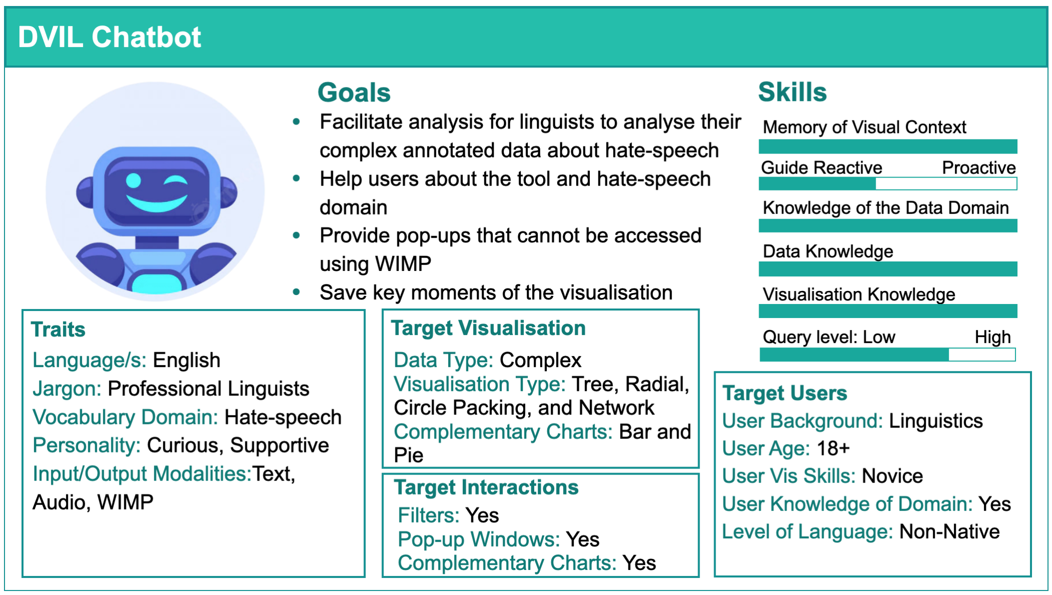


Figure 7.2: VisAgent Persona: DViL Chatbot characteristics.

Artefact 2: Visual Conversation Patterns

In our VisChatbot, from the eight proposed patterns (recall from Figure 6.6 to Figure 6.12 of Chapter 6), we applied the following:

- From the Visualisation family patterns, we used (1)-Request Visualisation Task and Query, and (2)-Request Visualisation Details. The steps defined in these patterns include the low and high-level queries that allow the users to perform the VisTasks-1AB and VisTasks-2AB. Indeed, the VisTasks-1AB and VisTask-2A are considered *low-level* queries that can be also performed using the GUI. However, the VisTask-2B is a *high-level* one, which is not available in the GUI and needs to be solved through multiple low-level queries like changing layout and glyph, glyph activation, highlighting features and access to supplementary statistical charts. For instance, VisTask-2B *"What is the most common level of toxicity for the feature*

Intolerance in the most toxic subtree" involves asking first about the most toxic subtree, and afterwards, once the user has visualised it, also involves asking about the most common level of toxicity for intolerant comments contained in this subtree. An example of this VisTask is displayed using Pattern 1 in the following Figure 7.3.

- From the Assistance Family, we selected the (4)-Ambiguity Helper pattern to address misunderstandings. Concretely, as there are two ways to select a hate speech feature - using the glyphs and highlighting on the main graph - this pattern defines the user-VisChatbot interaction to prompt the users and use buttons to allow them to choose the desired method (see Figure 7.4). Moreover, we selected (5)-Guidance Tool and (6)-Guidance Domain Patterns, which facilitate the visual analysis for users who are not familiar with the visualisation platform (helping with operations on layouts, statistics summaries, subtrees, etc.) and the data domain (features' definition, links to the news where the data come from).

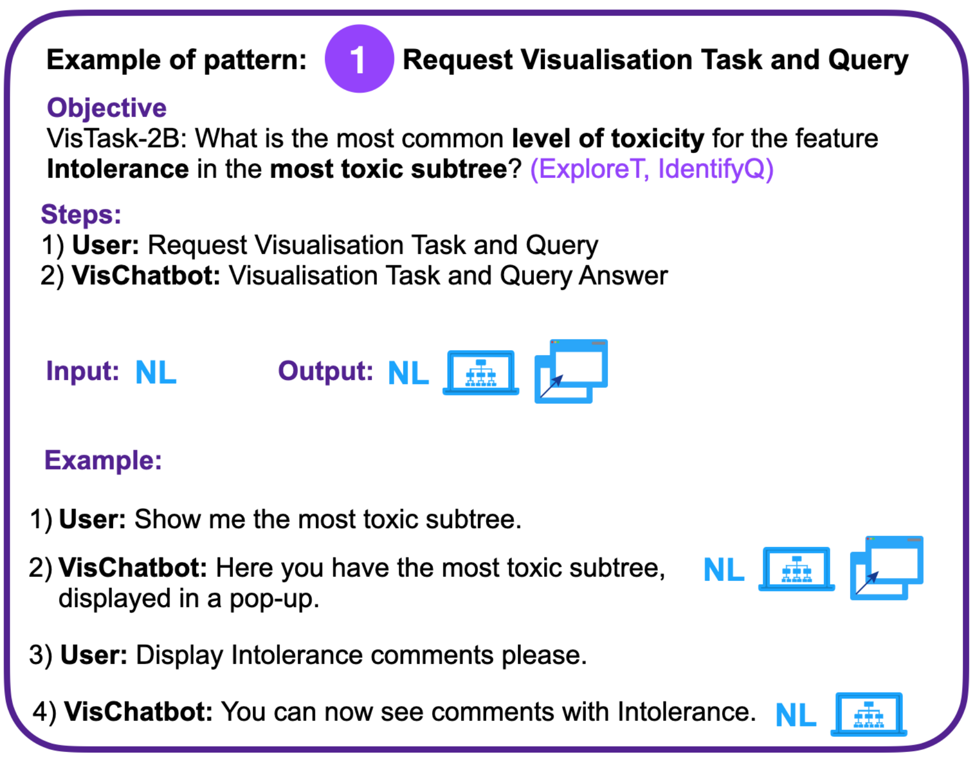


Figure 7.3: Example of Pattern 1: Request Visualisation Task and Query for VisTask-2B of the hate-speech case study.

Artefact 3: Annotated Transcripts

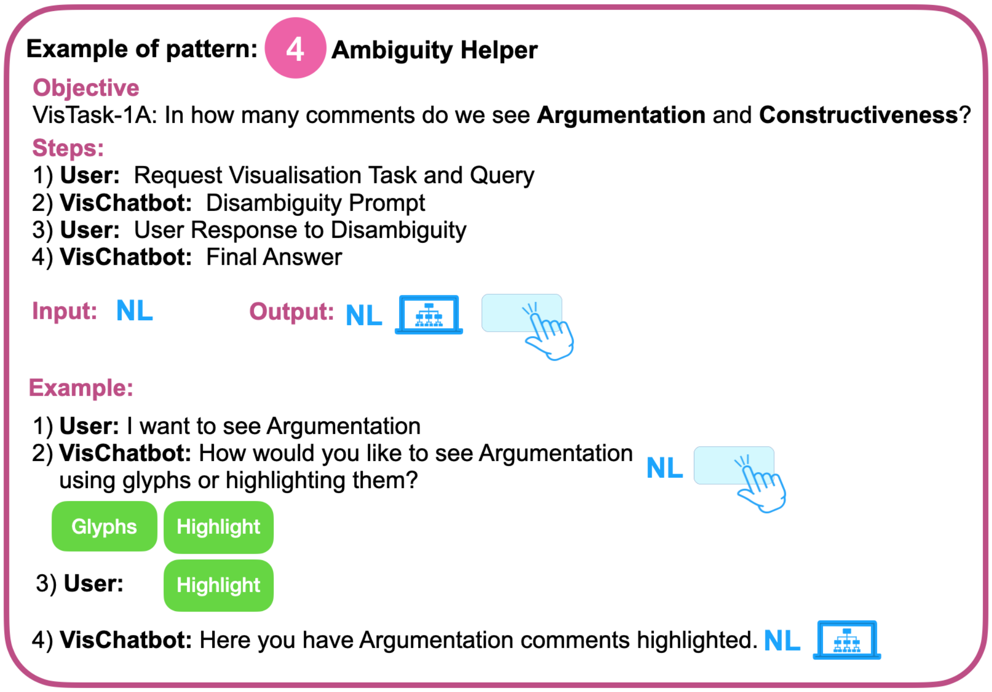


Figure 7.4: Example of Pattern 4: Ambiguity Helper for VisTask-1A of hate-speech case study.

The application of the patterns described earlier presents the initial instances of user-DViL chatbot conversations, setting the stage for the Transcripts definition. As an example, Figure 7.3 shows a possible pattern example displaying visual analytic conversation to perform VisTask-2B. In Figure 7.5, Annotated Transcripts for VisTask-2B can be observed. This figure presents the potential dialogue to solve VisTask-2B between a user and the VisChatbot. In the first interaction, the user asks to view the "most toxic subtree" using (1)-Request Visualisation Task Query Pattern, noting on the annotations that the VisChatbot should save "most toxic subtree" in its memory to understand future related commands. The VisChatbot should then display the requested subtree in a pop-up window, and the annotations specify that the visualisation should be updated and a pop-up generated. In the next interaction, the user requests to view comments with "Intolerance", using (1)-Request Visualisation Task Query Pattern. The chatbot should recognise "Intolerance" as a tagged feature and save it to memory. It should understand that this refers to specific nodes (comments) in the visualisation and updates the display to show these comments both in the pop-up and the main view, with filters turned on to focus on intolerance-related comments. Each user query includes notes about both the input (how the VisChatbot should interpret the request) and the output (actions taken by VisChatbot to update the visualisation and give textual feedback to the user), demonstrating how the VisChatbot should manage and visually represent complex data in response to user commands.

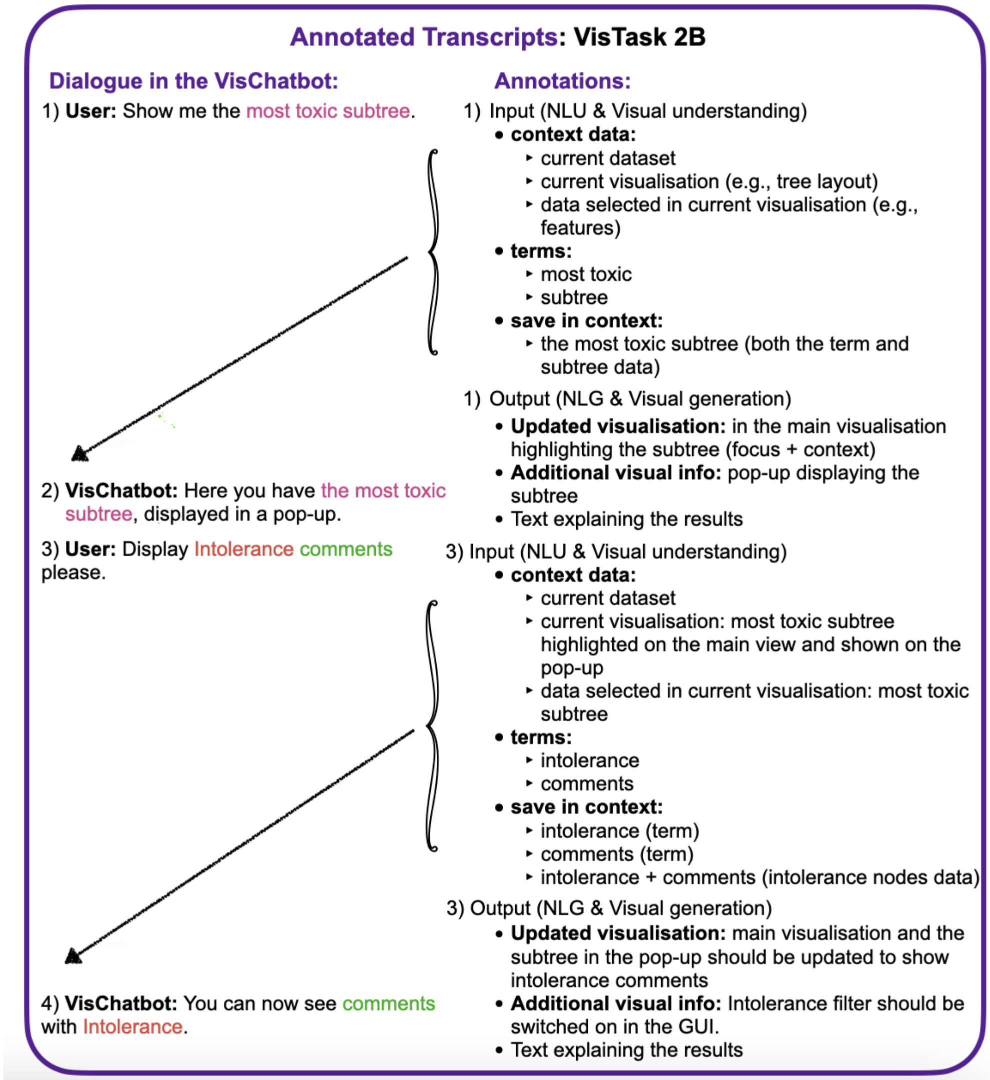


Figure 7.5: Annotated Transcripts example for VisTask2B.

7.3 DViL Chatbot: Development Phase

In this phase, we outline the three stages involved: VisChabot Modelling, where we discuss the technology selected to create our VisChatbot; Visualisation-Chatbot Connection, where we explain how we integrated this technology with our data visualisation platform, DViL; and finally, a User

Evaluation section, where we conduct a comprehensive evaluation of the DViL chatbot to ensure each iteration enhances the system and supports the ongoing development process.

VisChatbot Modelling

We used an information-retrieval ML-based technology that generates a trained model for the DViL chatbot. Information retrieval ML-based chatbots use machine learning techniques to retrieve and present relevant information from large datasets, documents, or knowledge bases in response to user queries. These chatbots incorporate models for intent recognition, entity extraction, and query understanding to process user input and generate appropriate responses [131].

Specifically, we selected Rasa [146], an open-source conversational platform. Rasa's key components include **intents**, which refers to the possible user queries; **responses**, which are the chatbot's replies; **rules**, which define a one-to-one mapping between a specific intent and its corresponding response; and **stories**, which are sequences of intents and responses that guide the chatbot's behaviour in more complex conversations. Moreover, **slots** are variables used to store and manage information extracted from user inputs during a conversation, aiming to help the chatbot remember the context and make decisions based on previous interactions. These components work together to help the chatbot to understand users' queries. Specifically, Rasa NLU (Natural Language Understanding) processes the user's input to identify intents and extract relevant data, while Rasa Core determines the appropriate response based on the context of the conversation. The Rasa NLU uses DIET (Dual Intent and Entity transformer) which is a multi-task transformer architecture that handles both intent classification and entity recognition.

Moreover, the Rasa Core uses TED (Transformer Embedding Dialogue) policies to predict the next action to be performed. When the next action is not a direct response, Rasa Action Server uses custom actions to perform operations such as recovering information from the conversation tracker (which involves accessing the bot's memory to retrieve past events and the current state of the conversation), composing responses and also to connect to external systems. In these actions, we implemented functionalities such as login, logout, sign-up, and opening a dataset, that can be managed through the VisChatbot.

Additionally, custom actions were created to enhance user interaction with visual data. As introduced in Chapter 4, Section 4.4, our platform DViL includes filters called "Select Node and Edge," allowing users to highlight specific graph features, where selected elements are emphasised and the rest are greyed out. We integrated this with the DViL chatbot, enabling the control of these filters directly through conversational commands. Moreover, actions were implemented to toggle the visibility of features using pronouns like "it" or "them," allowing users to refer to previously selected features without repeating their names. This is achieved through Rasa slots, which store contextual information throughout the conversation. They act as the memory of the chatbot, retaining important details. These actions streamline user engagement, facilitating complex graph manipulation and ensuring a seamless, and interactive data visualisation experience.

Indeed, all these components are defined based on the VisAgent persona, Visualisation Conversation Patterns, and Annotated Transcripts that are defined in the Design phase. For instance, we defined the intents in English, creating a comprehensive corpus that includes both visualisation-related jargon and hate speech, which are specifically based on our tagged features. In particular, we developed intents focused on visualisation layouts such as Tree, Radial, Force, and Circle Packing layouts. To complete the VisTasks, users need to interact dynamically with the features in the visualisations. We incorporated terminology focused on selecting, deselecting, and exploring parts of hierarchical data to ensure clarity and ease of navigation in task execution. On the one hand, the rules describe one-turn user-DViL chatbot interactions which correspond to Transcripts' lines. On the other hand, stories define multiple-turn interactions, coming either from a specific pattern or from a combination of patterns. An example can be observed in Figure 7.6, combining patterns Guidance Domain, Guidance Tool and Request Visualisation Task and Query, and the corresponding story. The descriptions of intents can be found in the Appendix B.

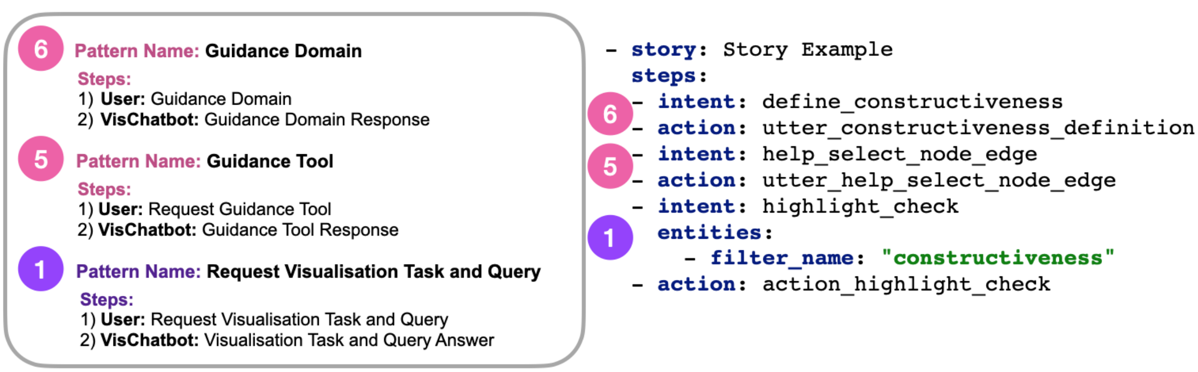


Figure 7.6: The combination of the patterns Guidance Domain, Guidance Tool, and Request Visualisation Task and Query and the corresponding Story defined in Rasa.

Visualisation-Chatbot Connection

Figure 7.7 shows the connection between the DViL chatbot and the web-based visualisation platform, which was developed using HTML/CSS, JavaScript, D3.js, and jQuery on the frontend, and Python and a PostgreSQL database on the backend. The DViL chatbot includes the main components of

Rasa: the Rasa Server, including Rasa NLU and the Rasa Core for language understanding and the generation of responses respectively; and the Rasa Action Server to perform more sophisticated actions and handle complex requests.

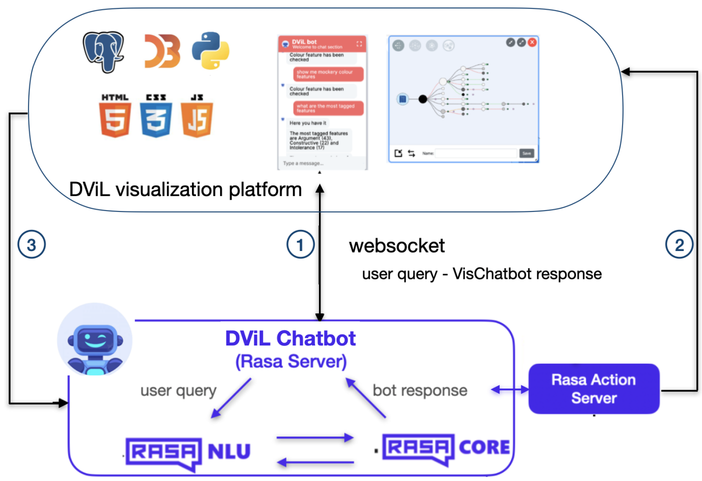


Figure 7.7: Visualisation Platform and DViL chatbot connection.

There are three channels of communication:

1. WebSocket (Chatbot Frontend to Agent Rasa Server): This WebSocket connection is responsible for enabling real-time, two-way communication between the chatbot's frontend and the Rasa server. Through this channel, user inputs are sent to Rasa, and responses are immediately received, ensuring smooth and interactive dialogue. For example, when a user types a message in the chat, such as "*Show me the constructive nodes*", the WebSocket connection sends this text input to the Rasa server. Rasa processes this input and determines what is the user's intent, and responds with the relevant information.

2. REST API (Rasa Action Server to Visualisation Platform Backend): This API enables direct communication between the Rasa Action Server and the backend of the visualisation platform (DViL). It is particularly useful for executing custom actions that have been predefined. For example, if a user requests to highlight specific features on the graph using commands with pronouns, such as "highlight it" referring to a previously selected node, the Rasa Action Server handles this through custom actions. The server uses the REST API to interact with the DViL backend to perform the necessary updates. It queries the backend to fetch and manipulate the relevant data, such as highlighting nodes based on the user's request.

3. Rasa HTTP API (Visualisation Platform Frontend to Rasa Server): This API enables the visualisation platform's frontend to interact with Rasa. For example, when a user inputs "*change layout to radial*" into the VisChatbot, the request is sent via the Rasa HTTP API. Rasa HTTP API interprets the command based on predefined intents and identifies the corresponding action, which triggers the graph layout change to radial. After executing the action, Rasa sends a textual response back to the chatbot frontend through the WebSocket connection established in step 1.

User Evaluation

We performed one test iteration, incorporating the aforementioned VisTasks-1AB and VisTasks-2AB (hereafter referred to as VT1A-B and VT2A-B). We aimed to assess the quality and effectiveness of the visual analytic conversation and the VisChatbot Usability. Following the metrics suggested in Table 6.1, we gathered: i) Task Completion Rate-TCR, ii) Interactional Efficiency, iii) Understanding Ratio, iv) Appropriateness of created/updated visualisations and v) Usability from BUS questionnaire [25]. We performed this evaluation with the approval of the Bioethics Committee of our university and obtained the participants' consent to store the acquired data on secure servers.

Participants and Setup

We recruited 16 participants which were mostly professionals and students from the Faculty of Philology and Communication at the University of Barcelona; 56% of the participants were male, 62% were aged between 18 and 30, 81% of the participants had experience in message annotation, and 63% of them in data visualisation. The study was an exploratory and moderated observational test, conducted in a classroom with participants doing the test one by one under the guidance of a moderator, and an observer. Participants interacted with our platform using Google Chrome on a computer. The moderator positioned alongside the user, using a separate computer to display the Google Forms that included VisTasks and instructions for the user to complete, ensuring that the user's interaction with the primary platform remained undisturbed. The observer was seated behind the user to observe and take notes while the user doing the study. In all the sessions, we recorded the screen, and audio as we asked users to think aloud and saved logs of the users' interactions and chat history. Furthermore, to maintain consistency across participants, we kept the task order uniform, ensuring that all participants experienced the same conditions. The goal was for users to start with the easier task and then move on to the more complex ones, progressing step by step. This approach allowed us to focus on testing the VisChatbot's conversational abilities, rather than evaluating users' performance or how quickly they completed the VisTasks. By structuring the tasks in this way, we aimed to identify errors and gather insights that could help improve future interactions.

Procedure

At the beginning of the sessions, we explained to the users our study with a presentation that included: a small introduction to the annotated corpus (tagged features), the structure of conversation threads, an explanation of how they were mapped onto our visualisation layouts, and an introduction to our platform and DViL Chatbot (10-15 min.). We did not provide any examples of how to ask queries to the DViL Chatbot to prevent any potential bias towards influencing participants to interact in a specific manner.

After the presentation, we provided participants with information about the structure of the test. Then, we gave users five minutes of training time to get them familiarised with the platform, however, we asked them not to interact with the DViL Chatbot at this stage and only interact with the platform using a mouse. During the training, first, we let the users explore the DViL platform freely and then, we guided users through the platform and asked them to complete some simple VisTasks. In the training time users were allowed to ask us questions about the platform (5 min.).

Afterwards, we asked participants to complete two VisTasks with subtasks, using the DViL Chatbot that we introduced in section 7.1 (total 20-30 min.). To complete the VisTasks, users were instructed to rely solely on the VisChatbot for interaction and not to use the mouse. They were informed that using the mouse would be considered a failure of the VisTask. In VisTask-1A users need to find the number of comments tagged with two specific features, VisTask-1B requires identifying the most common toxicity level for comments tagged with a particular feature and checking for any co-occurring features, VisTask-2A focuses on determining the most common toxicity level for another feature, and VisTask-2B involves exploring the most toxic subtree to find the common toxicity level associated with a given feature. While performing the VisTasks, we asked users to direct any questions they had to the DViL Chatbot first. The moderator intervened only in cases where a user encountered significant difficulties that impeded their ability to continue, such as repeatedly asking the same question without receiving an understandable response. Since each VisTask required users to apply filters and use complementary pop-up charts, after each VisTask we instructed users to clear all filters and close pop-up windows if there were any. Also, after each VisTask, we reminded users of DViL Chatbot's abilities by stating "*to help you carry out the analysis, you are free to ask the DViL Chatbot for a tutorial, definitions, explanations and fast help and to change layouts and glyph types at any point*". After finishing, the users provided feedback through open questions, the BUS usability questionnaire, and a small interview (5-10 min.).

Results

i) Task Completion Rate-TCR

Results were positive, with a high Task Completion Rate (TCR) of VisTasks (94% for VT1B and VT2B, and 100% for VT1A and VT2A). One participant, the first user in the evaluation, was unable to complete VT2B. Despite the concept of a subtree being explained during the presentation, the participant struggled to understand it and did not seek assistance from the moderator. For all VisTasks, we assisted users only if they first asked the DViL Chatbot and it was unable to answer their question. This assistance included reminding users of the information provided during training or in the presentation sessions, suggesting users to ask their questions in an alternative way, after a couple of failed attempts with the same question, stating that the DViL Chatbot is unable to perform that. Users received help during all tasks except for VT2A, where no assistance was needed, possibly due to participants getting more familiar with the VisTasks or the DViL Chatbot's functionality. There were cases where the DViL Chatbot failed to comprehend user queries, giving the users Guidance Domain or Guidance Tool responses, meaning, the VisChatbot provided users with responses including help about the domain of the data or the visualisation tool. From here on, these responses are referred as Assistance responses. For example, in VT1B, a user asked, "*Show me statistics for no toxic comments*" and in response, the VisChatbot provided an assistance response explaining how the statistics charts work, as the user's request was not clearly understood by the VisChatbot due to its inability to display the statistics of a single level of toxicity in the complementary charts. Interestingly, these kind of responses (Assistance response) aided several users in VisTask completion.

ii) Interactional Efficiency

To assess the interactional efficiency which is the number of turns needed to complete a VisTask, we set a minimum number of steps required to solve each VisTask. We present the overall aggregated results in Figure 7.8. The VT1A exhibited the lowest interactional efficiency at 38%, with users having used the highest number of total intents to complete the VisTask. This is understandable because VT1A was the first task and the initial interaction with the chatbot, so users naturally needed more time to familiarise themselves with the system, leading to lower efficiency. The efficiency increased to 58% with VT1B, indicating that users began to feel more comfortable and familiar with the system. While all other tasks showed higher efficiency than the first VisTask, the trend fluctuated slightly, starting with an increase as described, followed by a small dip to 43% for the VT2A, and then another rise to 46% for the VT2B. This is interesting because, in the final task, users needed to access a subtree using the VisChatbot for the first time. Despite this, VT2B have better interactional efficiency than VT2A.

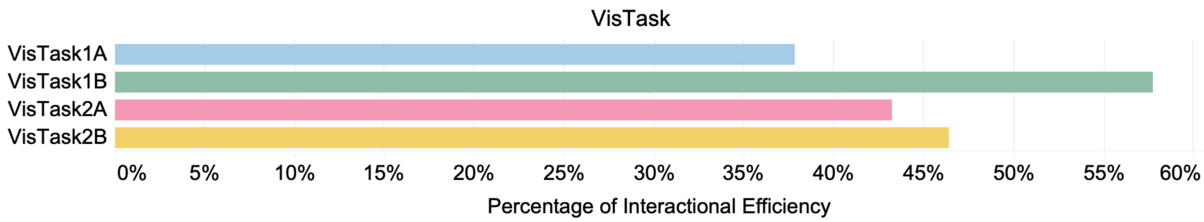


Figure 7.8: Summary of the aggregated results for Interactional Efficiency

Next, we summarise the results of each VisTask in the Figure 7.9. For VT1A, where the expected number of user intents to solve the task was three, most users exceeded our expectations (see Figure 7.9.a). User U12 notably used 15 steps, well above the required count, while others, like U11 and U15, remained closer to the expected value. The average number of user intents (indicated by the red line) was driven higher by outliers such as U6 and U12, showing a general trend of exceeding the baseline. In VT1B, with an expected count of four intents, users displayed a similar pattern, with most users surpassing the expectation (see Figure 7.9.b). Specifically, user U13 was the standout performer, using 21 intents to solve the task, far above the rest. However, a number of users remained closer to the expected value. The average number of intents to complete the task was seven. For VT2A, with a lower expectation of one intent to solve the task, most users performed close to or slightly above the requirement (see Figure 7.9.c). Users U6 and U10 stood out significantly, using six and seven queries respectively, creating a marked gap between them and the majority who completed the task in only one or two queries. The average was two which is represented by the green line. Finally, in VT2B, where the expected number of intents was two, performance was more varied (see Figure 7.9.d). While users such as U6 and U10 again stood out with nine queries each, others, like U3, U4, U15, and U16, remained closer to or below the expected value. The average was four, represented by the green line. In the following, we provide the details of the actual user performance compared to the minimum expected intents for each VisTask. In the comparison, we excluded the user intents regarding the Assistance patterns. The results of VT-1AB are summarised in table 7.1 and VT-2AB are in table 7.2.

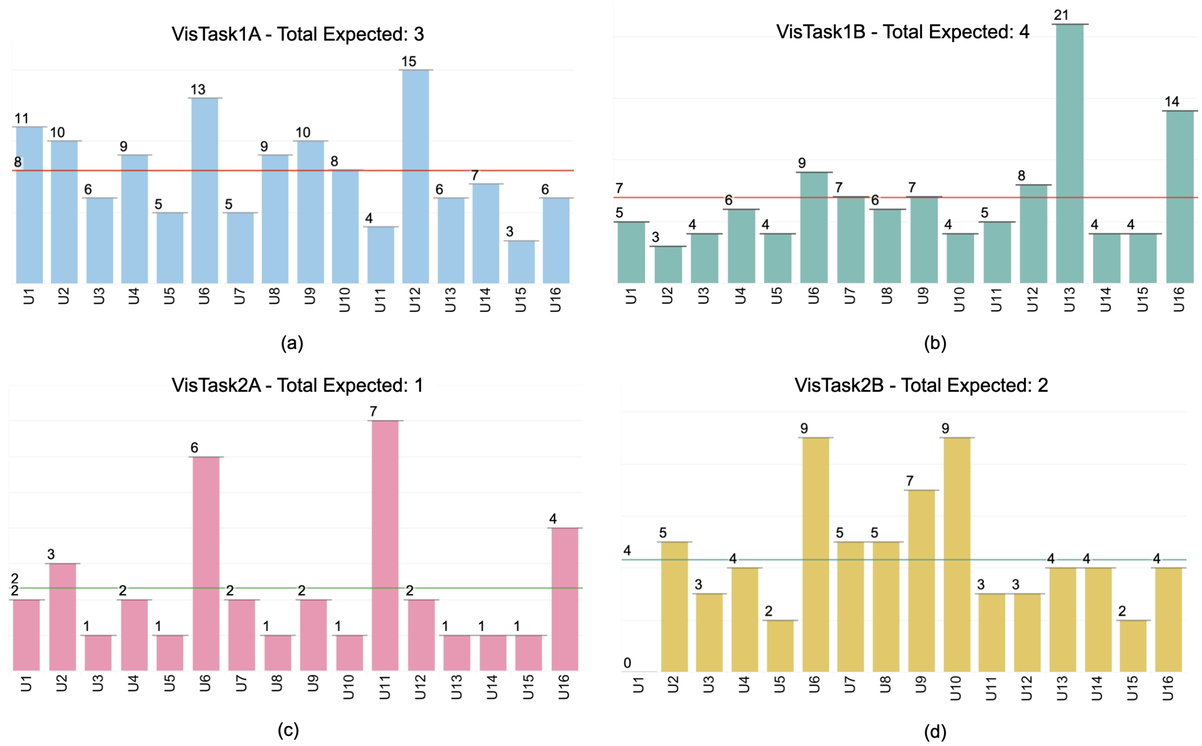


Figure 7.9: Summary of the results for Interactional Efficiency for a) VisTask1A, b) VisTask1B, c) VisTask2A, and d) VisTask2B.

In the initial VisTask labelled as **VT1A** (that aims to find the number of constructive and argumentation comments) the baseline number of intents was three (see Appendix B, Table B.1 for the steps to complete this VisTask). Given that this was the users' first interaction with the DViL Chatbot, most participants naturally inquired about more intents compared to the baseline. It should be noted that in this VisTask, the average number of both Visualisation intents and intents asking for assistance (from now on referred to as Assistance intents) was higher than in other tasks. This increased level of interaction was likely due to users' curiosity and eagerness to explore the DViL Chatbot's capabilities and understand the evaluation study. The average number of intents for this VisTask was eight. See Appendix B, Figure B.4 for an example of the expected conversation, and Figures B.5 and B.6 for two real user conversations, the longest and minimal conversations, respectively.

In this VisTask, most participants attempted to get answers directly from the DViL Chatbot without using any filters. For example, participants asked, "*How many comments show both argumentation and constructiveness together?*" However, our chatbot is not designed to answer

Table 7.1: Results of Interactional Efficiency for VisTasks 1A and 1B.

	VisTask1A: Total Expected: 3			**VisTask1B: Total Expect: 4**		
User id	**#Vis**	**#Assist.**	**#Total Exp./#Vis**	**#Vis**	**#Assist.**	**#Total Exp./#Vis**
U1	11	0	3/11	5	0	4/5
U2	10	2	3/10	3	0	4/3
U3	6	0	3/6	4	0	4/4
U4	9	1	3/9	6	0	4/6
U5	5	0	3/5	4	0	4/4
U6	13	1	3/13	9	0	4/9
U7	5	1	3/5	7	0	4/7
U8	9	0	3/9	6	0	4/6
U9	10	0	3/10	7	0	4/7
U10	8	0	3/8	4	0	4/4
U11	4	0	3/4	5	0	4/5
U12	15	1	3/15	8	0	4/8
U13	6	0	3/6	21	0	4/21
U14	7	0	3/7	4	0	4/4
U15	3	0	3/3	4	0	4/4
U16	6	2	3/6	14	0	4/14
Avg.	**8**	**-**	3/8	**7**	**0**	4/7

such questions specifically. Instead, its primary aim is to facilitate visual analysis and the use of intricate visual analytic tools. Consequently, users spent time understanding how to approach the DViL Chatbot and the VisTasks. Another common challenge was attempting to select two features simultaneously, which is not currently supported by our VisChatbot. In these cases, we reminded users about the VisChatbot's capabilities.

Moreover, when examining participants U1, U2, U6, U9, U12, and U16, who had 10 or more intents, we found that only U12 and U16 had more failed intents than successful ones. It should be noted that some users explored aspects unrelated to the VisTask. Additionally, U2, U4, U7, U12, and U16 asked Assistance intents more frequently than in other tasks, supporting the idea of users being curious. U2 sought help with the data, while U4, U6, and U12 sought help with the DViL platform. Interestingly, U11 solved the VisTask by filtering for argumentation alone. By observing the complementary statistics chart, which showed the number of each feature with argumentation, U11 identified the number of constructive comments to determine how many nodes had both argumentation and constructiveness. This can be observed in Figure 7.10 that the user asked to see argumentative comments and then to see the statistics information. It can be seen that the user is pointing to the constructive column in the statistics bar chart to find the answer.

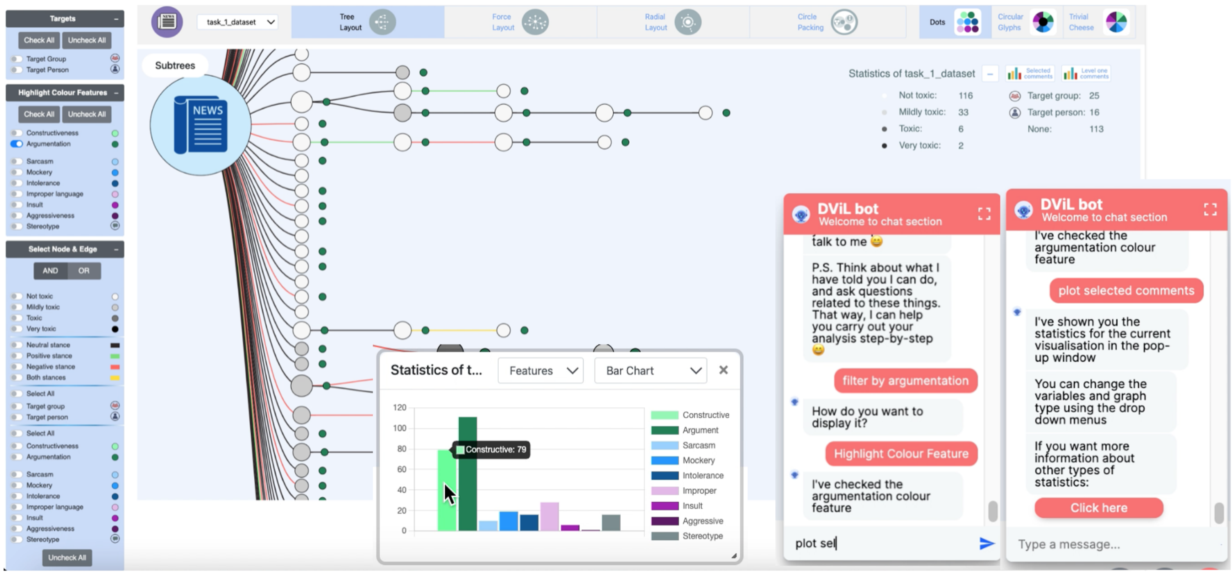


Figure 7.10: Conversation of the U11 to solve VisTask1A

The minimum number of required intents to solve **VT1B**, the one that aims to find the most common level of toxicity with constructiveness, was four (see Appendix B, Table B.1 for the steps to complete this VisTask). In this VisTask, five users (U3, U5, U10, U14, U15) completed it with four intents and the average number of intents was seven. It should be noted that compared to the previous VisTask, the total number of intents of the most participants decreased significantly. This could be because users began to better understand the system's functionality. Moreover, in this VisTask, users did not use any Assistance intents.

U2 is the only participant who completed the VisTask with fewer intents than the baseline using three intents. However, when we examined the user logs and recorded screen, we found out that the user used the mouse to open the statistics window therefore, in this case, we do not consider this VisTask as completed. Another noteworthy observation is that U13 used 21 intents to solve the VisTask. To address the second part of this VisTask, which involves identifying the features that co-occurred with constructiveness, users needed to make use of a statistical chart. This user used a different approach and applied and removed all the features to see which ones were present thus, having a lot more intents at the end compared to the baseline. U16 used a total of 14 intents, getting stuck on visualising complementary charts because these questions were not included in the training data, such as "*please show me the histogram*" or "*visualise the data*".

From the feedback we gathered from the open questions, participants stated that they initially faced some confusion and uncertainty when using the DViL Chatbot and interface as they were new to the system. However, with time, they found it easier to navigate and interact with it. Users stated that they adapted their way of communicating according to the DViL Chatbot to make sure it understood them. This can also be confirmed by the decrease in the number of intents between VT1A and the subsequent tasks.

Table 7.2: Results of Interactional Efficiency for VisTasks 2A and 2B.

User id	**VisTask 2A: Total Expected: 1**			**VisTask 2B: Total Expected: 2**		
	#Vis	**#Assist.**	**#Vis/#Total Exp.**	**#Vis**	**#Assist.**	**#Vis/#Total Exp.**
U1	2	0	1/2	/	/	/
U2	3	2	1/3	5	1	2/5
U3	1	0	1/1	3	0	2/3
U4	2	0	1/2	4	0	2/4
U5	1	0	1/1	2	0	2/2
U6	6	0	1/6	9	0	2/9
U7	2	0	1/2	5	1	2/5
U8	1	0	1/1	5	0	2/5
U9	2	0	1/2	7	0	2/7
U10	1	0	1/1	9	0	2/9
U11	7	0	1/7	3	0	2/3
U12	2	0	1/2	3	0	2/3
U13	1	0	1/1	4	0	2/4
U14	1	0	1/1	4	0	2/4
U15	1	0	1/1	2	0	2/2
U16	4	0	1/4	4	0	2/4
Avg.	2	-	1/2	5	-	2/5

Users could solve **VT2A** (aims to find the most common level of toxicity with stereotype comments) with one intent (see Appendix B, Table B.1 for the steps to complete this VisTask). Seven participants completed this VisTask with one intent as expected and the average is two intents. Only U2 used two Assistance intents to help with the context of the data. Moreover, users needed a minimum of two intents to solve the **VT2B** (see Appendix B, Table B.1 for the steps to complete this VisTask), which aims to find the most common level of toxicity for the feature intolerance in the most toxic subtree (see Figure 7.11 for an example of to solve this task). Users first had to ask to see the most toxic subtree and then filter the comments with intolerance and find the answer visually as there were few nodes. Additionally, they can open the statistics window, as it is synchronised with the pop-up window that displays the subtrees in the DViL Chatbot. At this stage of the evaluation, users were more confident using the DViL Chatbot. Two users completed the VisTask using only two intents, while the average number of intents for this VisTask was five. Some users spent time trying to understand what the most toxic subtree was, mistakenly thinking the VisTask referred to the most toxic nodes. We had to re-confirm to most users that a subtree is indeed different from a node.

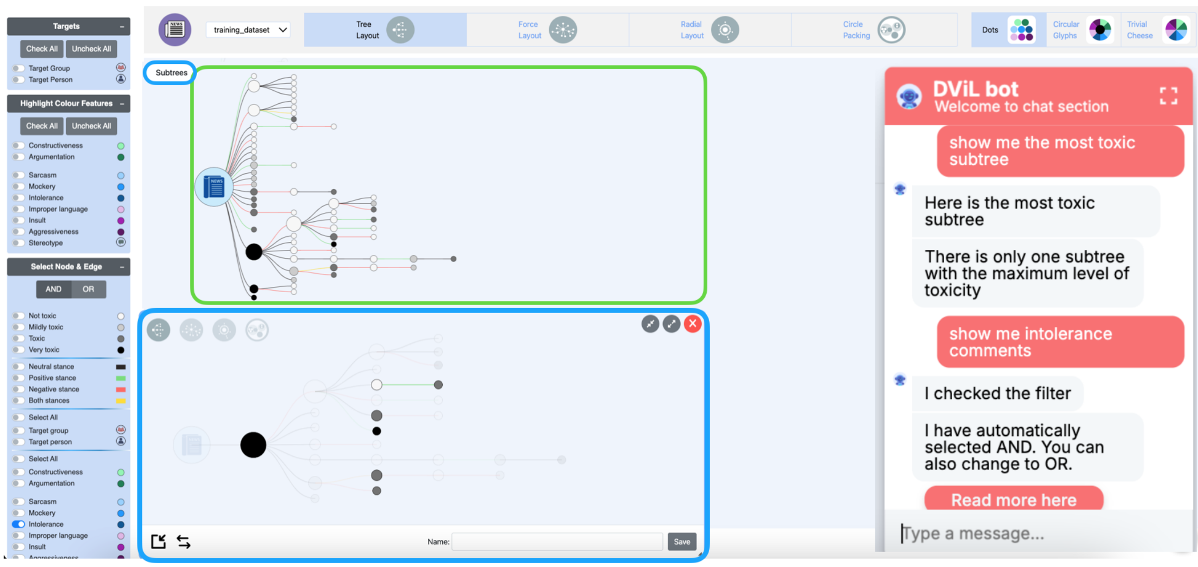


Figure 7.11: VisTask2B: User-DViL chatbot dialogue and VisTask solution example.

iii) Understanding Ratio

In this section, we present the results of the DViL Chatbot's understanding ratio, which is closely related to the classification task performed by the NLU module of the Vischatbot. We conducted separate analyses to take a closer look at the Visualisation intents used for solving VisTasks and using Assistance (user intents to ask directly for assistance from the VisChatbot). This allowed us to better understand the role of different types of intents and offer a more comprehensive analysis. We categorised failed intents into two categories for both Visualisation intents and Assistance intents. In Visualisation intents, we have Assistance failed and failed intents. Assistance failed refers to scenarios where intent has initially failed, but the DViL Chatbot responded with an Assistance response. These particular responses are designed to assist the user, which in some cases helped the users complete their VisTask successfully. In contrast, failed responses had different reasons to fail, such as the query was impossible for the DViL Chatbot to understand, or it failed because of not enough training data. We will elaborate on the failed cases in the following.

We summarise the results of visualisation intents in Figure 7.12 and in Table 7.3, which presents the understanding ratio. We observed significant differences in the understanding ratio between the first two VisTasks, while no notable differences were found in the remaining tasks. This indicates that initial interactions with the VisChatbot tend to be less effective; however, as users engage in more interactions, the communication between the user and the VisChatbot improves.

VT1A has a relatively lower understanding ratio, with 70 successful intents out of 126 (56%). This can be due to the VisTask being the first interaction for users with the DViL Chatbot and they required some time to adapt to the chatbot's communication. This VisTask had 11 failed visualisation intents answered through responses of Assistance intents, i.e., the visualisation intent was incorrectly classified as an assistance intent, but they finally guided the user to perform the task. Among these, six of them helped users with their queries and of them, 5 did not. For example, U1, U2, U5, U10, U12, and U15 all asked the chatbot "*how many comments are selected?*", it answered this question by explaining, that the Selected Comments button displays the overall number of features for comments that the user has selected in complementary charts, directing the user to the right point. However, in some cases, such as when U8 asked "*how can I see the number of comments with argumentation?*", the chatbot answered with a Guidance Domain answer from the Assistance pattern by stating a definition of Comment ID, which at the end did not guide the user to the right direction to solve the VisTask. The VisChatbot could be improved by identifying more potential user misunderstandings and enhancing its ability to ask follow-up questions. For instance, when a user mentions "argumentation" the chatbot could clarify whether they want to view comments related to argumentation or additional data on complementary charts about it. Expanding the training data to cover a broader range of user intents could also make interactions smoother and more intuitive.

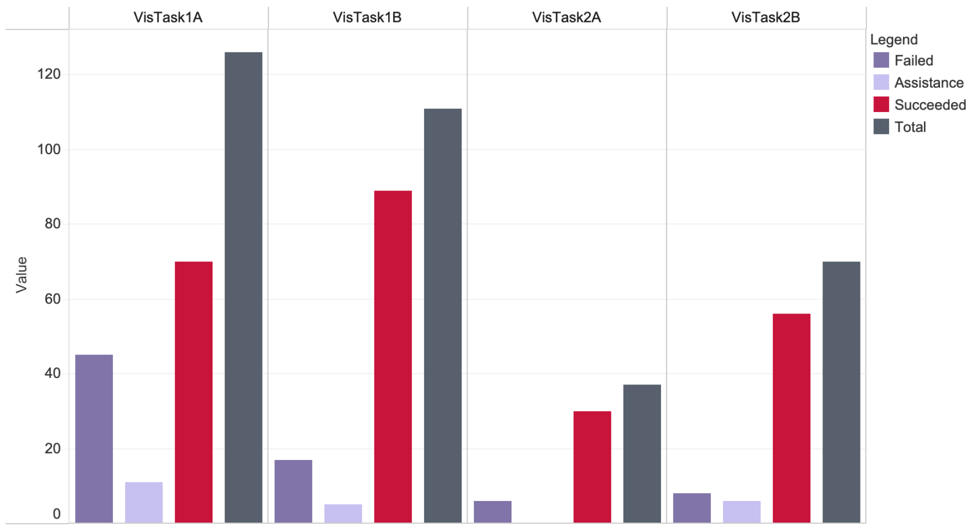


Figure 7.12: Results of the Understanding Ratio - Visualisation Intents for each VisTask

Table 7.3: Results of the Understanding Ratio - Visualisation Intents

VisTask id	**Visualisation Intents**			
	Succeeded	**Failed**		**Unders. Ratio**
	#succeed	#Assistance	#failed	#succeed/totalI
VisTask-1A	70	11	45	70/126=56%
VisTask-1B	89	5	17	89/111=80%
VisTask-2A	30	0	6	30/37=81%
VisTask-2B	56	6	8	56/70=80%
Total	**245**	**22**	**77**	**245/344=71%**

Moreover, this VisTask had the highest number of failed intents. This could again be due to users still familiarising themselves with the chatbot's capabilities and vocabulary. Out of the 45 failed intents, 19 were due to attempts to filter two features at once—-an unsupported action by the VisChatbot– despite being explained in training. Fourteen additional failures occurred from unsupported queries, such as when U1 and U9 requested statistics charts for specific features (e.g., constructiveness). However, it is possible that the users misunderstood the functionality of the DViL Chatbot. While our VisChatbot offers complementary charts, such as bar or pie charts, which visualise the number of each feature in selected comments, it is important to note that these charts display the count of comments associated with features like constructiveness, rather than presenting detailed statistics chart for individual features.

Moreover, U9 asked to see non-constructive comments, U10 asked to see the subtree of selected nodes, and U11 asked the chatbot to count argumentation comments, either questions made no sense or are not possible to do in our VisChatbot. Additionally, nine of the intents failed because of the lack of training data. Among those, seven of them failed because U2 and U6 were asking to change the filter option from AND to OR. We concluded that we need to review these specific intents and add more training data. Furthermore, in this category, U13 asked the DViL Chatbot to filter argumentation nodes and then asked: "*and select those with constructiveness as well*". Although these kinds of intents are included in the training data, this intent failed. Only two of them failed because of grammar or spelling errors.

In **VT1B**, the success ratio significantly increased to 89 successful intents out of 111 (80%), indicating that users became more comfortable with the DViL Chatbot. In this VisTask, five intents failed in the Assistance category. For example, U5 asked "*How toxic are these comments?*" and was guided to the Selected Comments statistics charts to visualise toxicity. However, U7 and U9 asked "*What are the relevant features when the level of toxicity is not toxic*" and "What is the most typical for constructive comments?" failed to receive proper guidance on toxicity, and U12's query about "*What features are included?*" resulted in guidance on layouts instead.

In this VisTask, 17 intents failed. Half failed due to being unsupported by the VisChatbot. For example, two users asked, "*What other features appear with a non-toxic level of toxicity?*". This high-level query is not supported by our chatbot yet. Some users also attempted to apply two filters simultaneously. Some failures occurred due to a lack of training data. For instance, U7 asked to "*mark the comments with the feature constructiveness*", but the word 'mark' was not included as a synonym for select, causing the failure. U16 asked for complementary charts in ways not covered in the training data, like "*show me histogram*" or "*visualise the data*". Additionally, some queries failed due to the context of the conversation managed by the chatbot was not complete enough. For example, U11 simply said "*unfilter*," and U15 asked "*show me the graph*" without specifying which filters to unselect or which graph to display. This issue stemmed not only from the users' input but also from the chatbot's inability to clarify or disambiguate. While the DViL chatbot can handle pronouns, it struggles with vague or ambiguous references. This limitation should be noted in the annotated transcripts and could be addressed by prompting users for clarification or improving context storage for better handling of follow-up questions.

VT2A showed a strong understanding ratio, with 30 out of 37 (81%) intents succeeding. This is likely because VT2 required only one intent to complete, and by this point, users were more familiar with the VisChatbot's functions. For instance, U6 requested to "*uncheck filters*" without specifying which ones, and U11 repeatedly asked to "*filter stereotype*" despite prompts to rephrase. These issues may point to gaps in the training data.

The final VisTask **VT2B** had 56 successful intents out of 70 (81%), with only six Assistance failures and eight other unsuccessful intents. All Assistance failures were unhelpful, such as when U14 asked, "*What is the most common level of toxicity for feature intolerance?*" and received information about the level of toxicity. Among the eight failures, two were due to user errors either conceptual or grammatical, like U6 attempting to "*uncheck toxicity*" and spelling "*selected comments*" incorrectly. U7 and U10 also asked to see charts with undefined queries, containing grammar mistakes. The remaining failures were due to missing scenarios in the annotated transcripts of the design phase, such as visualising a subtree with only very toxic comments.

Similarly, we used the same approach to analyse Assistance queries, and we have Visualisation failed intents and failed intents. That is, if a user makes an Assistance query, they either receive an inappropriate visualisation response or the Assistance intent simply fails. We analysed 11 Assistance intents (intents asking for an assistance such as Guidance Domain or Guidance Tool), of which seven were successful. VT1A had the highest number, with six. For instance, U2 asked for definitions of constructiveness and argumentation, U7 asked for the definition of constructiveness, U4 asked about the filter in the UI, and U6 asked about the difference between AND and OR. There were no Assistance intents in VT1B. VT2A had two Assistance intents. In the first, U2 asked for the definition of intolerance with poor grammar and received an incorrect response, but after rephrasing the question, they received the correct answer. VT2B had one Assistance intent, where U7 asked about a subtree and received the correct answer.

Correlation analysis between Interactional Efficiency and Understanding Ratio

We explored the relationship between Interactional Efficiency (IE) and Understanding Ratio (UR) only for VisTasks 1B and 2B since VisTasks 1A and 2A served primarily as onboarding tasks. Specifically, we calculated the Pearson correlation coefficient to identify how strongly these variables are related to each other. For VisTask1B, the correlation coefficient is $r = 0.29$ (not statistically significant with $p - value = 0.2675 > 0.05$), which indicates a weak positive correlation where higher efficiency in User-VisChatbot interactions is associated with a higher understanding ratio. Likewise, the correlation between IE and IE in VisTaks2B shows a weak association $r = 0, 36$, with a marginal significant trend ($p = 0.0505 > 0.05$). This analysis confirms the importance of ensuring that the VisChatbot has a strong understanding capability to facilitate user interactions with the visualisation. Nevertheless, it would be needed to go further with this preliminary study to better capture the relationship between IE and UR, because of the factors such as the limited understanding of the VisChatbot, the small sample size and the presence of outliers that could distort the correlation.

iv) Appropriateness of created/updated visualisations

To assess the Appropriateness of Created/Updated Visualisations, users were asked to select their preferred layout before starting each VisTask. As detailed in Chapter 3, we introduced a formalisation for automatic layout selection, which is also applied to the subtree visualisations in pop-up windows by the DViL Chatbot. Upon starting, the VisChatbot automatically selects the layout based on this predefined selection. In this evaluation, we aim to compare the VisChatbot's automatic layout selection with users' layout preferences to understand how well the automatic system aligns with user expectations. Thus, in the following, we present the results of the Appropriateness of Created/Updated Visualisations. For VT1A the appropriateness was 44%. In this VisTask, our chatbot suggested the best layout to use is Radial. Most of the users stated that it was easier to differentiate the comments, see depth, and visualise the data as a whole with the Radial layout (7/16). Although some users found the Force layout (3/16) cool and organised, eventually Radial layout was preferred as it was clearer with this dataset. Additionally, most users agreed that due to the compact nature of the dataset, the Tree layout (6/16) resulted in a wide visual representation and required extensive zooming, which ultimately caused a loss of the comprehensive view of the entire visualisation.

For VT2A the appropriateness was also 44%. Most users selected the Tree layout (7/16) although the DViL Chatbot selected the Force layout for them. We asked users to justify their choices and collected valuable insights into their decision-making. Those who opted for the Tree layout both stated that they selected this layout as it was easier to see information. Although some users found Force and Radial layouts similar stated that it was more comfortable and clear to see the information with the Radial layout. Users who preferred the Force layout commented that they appreciated its ability to present information without overlaps, resulting in a more organised representation of the dataset. Specifically, users noted that the Radial layout made it easier to differentiate between two levels of comments and understand the relationships between them.

Finally, for VT2B the appropriateness was 93%. In this VisTask the DViL Chatbot selected Tree layout to visualise the subtree in the pop-up window. All the users except one preferred Tree layout. We excluded one answer from the results as it was decided based on the wrong visual view. Similarly, we asked users to justify their selections. Users stated that it was easier to understand the hierarchical relationship between comments and the distribution of the data was very clear.

Overall, the automatic layout selection by the DViL Chatbot was well received by users. For two VisTasks, the VisChatbot's layout selection matched users' preferences, demonstrating the system's effectiveness. Although the DViL Chatbot selected the Force layout for VT2A, users generally preferred the Tree layout, finding it easier to interpret the information. Users who favoured the Force layout appreciated its well-organised structure and the lack of overlaps, enabling a more comprehensible presentation of the data.

v) Usability from BUS questionnaire

We asked users Bot Usability Questionnaire [25] to test users' perceptions and experiences with the DViL Chatbot and we asked them to rate the questions on a scale from 1 (strongly disagree) to 5 (strongly agree). Table 7.4 displays the questions included in the questionnaire and Figure 7.13 shows the average of the results on the bar chart, accompanied by a standard deviation illustrated on each question. Overall, the data suggests that users had generally positive perceptions of the DViL Chatbot. The average ratings for three statements are above 4, indicating favourable opinions and the rest is 3 and above. Moreover, we believe that results are in line with the analysis we made previously as the overall understanding ratio for all tasks is 71% (245/344).

Table 7.4: Bot Usability Questionnaire

1. I was immediately made aware of what information the chatbot can give me.
2. The chatbot gives me the appropriate amount of information.
3. The chatbot's responses were easy to understand.
4. I feel like the chatbot's responses were accurate.
5. I find that the chatbot understands what I want and helps me achieve my goal.
6. The interaction with the chatbot felt like an ongoing conversation.
7. The chatbot was able to keep track of context.
8. Did the chatbot help you to understand visualisations?

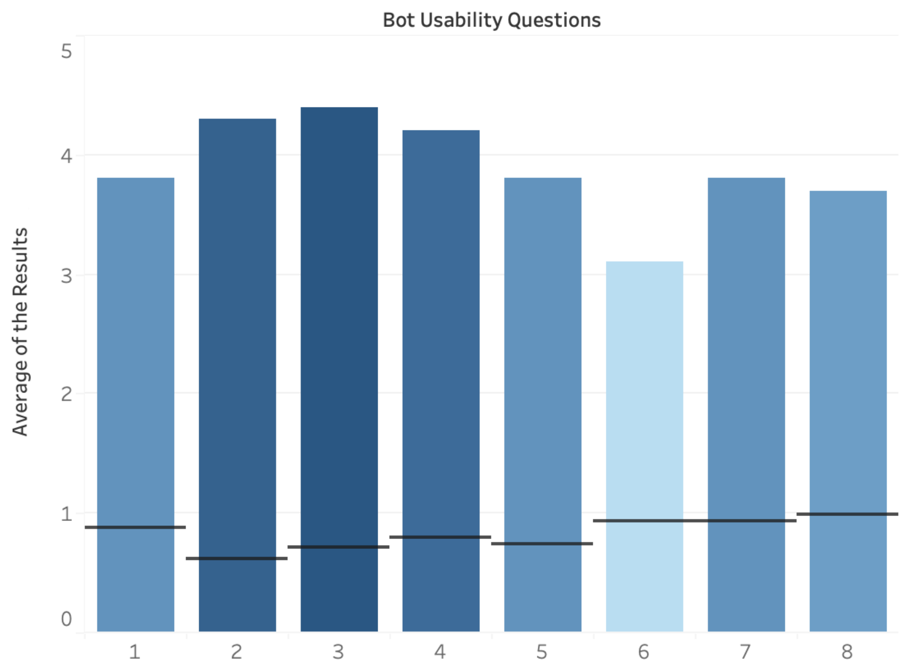


Figure 7.13: Bot Usability Questionnaire Results (responses range from 1 to 5, as shown on the y-axis). The bar chart displays the average results for each question, with the standard deviation represented as a horizontal line across the bars.

Additionally, when we asked users to elaborate on question eight, they expressed that the chatbot played a significant role in helping them understand the visualisations. They appreciated the clarity and usefulness of the definitions they requested from the DViL Chatbot. Users acknowledged the chatbot's usefulness in explaining the definitions of different filters. Some users expressed a desire for the DViL Chatbot to provide specific numerical answers corresponding to the graphics, such as indicating the number of comments with specific features. Moreover, some users stated their preference for using a mouse over the chatbot. This may be due to the familiarity and ease of interacting with traditional input methods, highlighting a potential area for improvement in the chatbot's usability or the learning curve for users.

Also, we asked participants to provide suggestions or recommendations for future improvements. Their feedback highlighted several desired features. Users expressed the need to perform multiple actions simultaneously, enabling them to accomplish several tasks concurrently. They also desired the ability to view the opposite ends of the hierarchy, such as identifying both the least toxic threads. Participants requested more direct questioning capabilities, allowing them to specify complex queries like finding comments with specific features and a designated toxicity level. Furthermore, users expressed an interest in accessing the original article and obtaining information regarding the comment level where the most toxic comments were located. Also, some users suggested to use of matrix visualisation to analyse all combinations of features.

7.4 Discussion

In the previous chapter, we proposed the VisChat Methodology, a comprehensive approach designed to guide the creation of effective VisChatbots. To illustrate the practical utility of VisChat Methodology, in this chapter, we presented a case study focused on hate speech analysis in social media. This case study demonstrates how the methodology can be applied in real-world scenarios to address complex challenges and deliver meaningful insights. Our experience as designers, and in line with other chatbot methodologies in education [13] and healthcare [30], suggests the utility of our proposal in directing the design of VisChatbots beyond the specific context of hate speech.

As previously discussed, the Data and Task Analysis stages are closely interrelated, and their sequence can vary depending on the domain and application (data-driven or task-driven). In our case, since the data had already been annotated by linguists with specific features within the hierarchical comment structure, it was most natural to start by exploring the data before defining the tasks to validate our hypotheses. Conversely, in more established domains, task analysis is often prioritised initially [64].

Furthermore, our case study is limited to only one iteration of the design and a preliminary evaluation performed by 16 participants. Although we only conducted one iteration, we collected a considerable amount of data, including recorded screen and audio data, as well as logs which will be used in the following iterations to refine and enhance the VisChatbot's performance and user experience, either passing through Analysis and Design stages or by improving the model training using reinforcement learning and parameter tuning [48].

Finally, when we explored the results, we found that the evaluation demonstrated a notably high task completion rate, highlighting the effectiveness of the DViL Chatbot. The high completion rate can be partly due to the moderator's assistance, which included restating instructions and clarifying questions, particularly in the initial and final tasks but never involved instructing participants on how to interact with the VisChatbot. Users received more help with understanding concepts such as subtrees, indicating that initial unfamiliarity with the visualisations contributed to the need for intervention. Over time, participants required less assistance. This suggests that users adapted to the system and became more self-reliant.

The higher number of intents in VT1A compared to other tasks likely stemmed from users' initial interactions with the DViL Chatbot and their curiosity about the new interface and data context. Users, unfamiliar with how to effectively use the chatbot, asked more questions and explored functionalities beyond the VisTask's scope. Despite providing explanations, we did not offer examples of chatbot interactions, which led to a greater number of exploratory queries. This exploratory behaviour contributed to a lower understanding ratio in VT1A, compared to the 80% or above understanding ratios in subsequent VisTasks. As users became more familiar with the chatbot's functions and adapted their communication strategies, the number of intents decreased and understanding improved, reflecting a typical learning curve associated with new technology.

Our analysis of failed intents revealed that some issues stemmed from insufficient training data. Users frequently posed queries or used terminology that the chatbot was not prepared to handle, such as changing filter options or using specific terms as synonyms. Addressing these challenges involves expanding and refining the chatbot's training data to better recognise and respond to these user nuances and preferences, a process informed by the insights gathered from this user evaluation. On the other hand, some errors arose due to limitations in the chatbot's capabilities, such as handling simultaneous feature filtering, counting comments with specific features, excluding certain features, or retrieving subtrees of selected nodes. This suggests us to revisit the Analysis phase to add additional requirements and do another iteration for the VisChatbot development. While we are addressing the challenge of simultaneous feature filtering, other requests, like viewing non-constructive comments, are not aligned with the data logic. Feedback from our linguist partners indicates that users are more interested in combining features rather than isolating them, suggesting a need for further refinement in handling such intents.

Another key finding of user evaluation is that the DViL Chatbot occasionally provided Assistance responses to queries that did not explicitly request assistance. While some of these responses were beneficial, guiding users to relevant resources and giving explanations, others did not provide clear direction. Despite their limited use, Assistance patterns demonstrated a high understanding ratio, indicating their effectiveness when employed. To improve the conversation flow of the DViL chatbot, we need to analyse user feedback and their conversations with the VisChatbot more thoroughly. This includes adding more potential user intents, whose absence led the VisChatbot to respond with Assistance responses. Additionally, enhancing the VisChatbot's ability to retain more conversational context could help reduce the need for assistance by improving its understanding of user queries. All the findings and implications in VisChatbot design are really useful to refine our initial Design of the DViL Chatbot for further evaluations following the green circle in Figure 6.1.

Finally, throughout all phases, we were concerned about ethical and privacy issues, particularly during the stages of user analysis, and data analysis, as well as during the user evaluation stage. Users were informed that their data would be securely stored and kept private. For the design of the chatbot, we ensured that harmful contexts were excluded. This involved verifying the authenticity of the data, securely storing them, and minimising bias in the visualisation of sensitive content (by using balanced datasets and reviewing visual representations to avoid reinforcing stereotypes or misrepresenting vulnerable groups).

Part III

Dissertation Conclusions

Chapter 8

Conclusions

This final chapter provides a summary of the contributions in relation to the established objectives, discusses unresolved research challenges, and outlines potential directions for future work.

8.1 Contributions Summary

Tracking and analysing the vast amounts of data generated from social networks, digital platforms, and societal activities presents a significant challenge, not only because of the overwhelming volume but also due to the complex relationships embedded within the data. Data visualisation emerges as an effective approach that enhances the analysis by helping to uncover profound insights and intricate patterns. However, creating meaningful visualisations also presents challenges, as is the case with **hierarchical** and **multivariate** data. Hierarchical data contains interrelationships that require careful analysis, while multivariate data adds additional dimensions to analyse, as each point is linked to multiple attributes. Therefore, to develop effective visualisations, it's essential to avoid **visual clutter** and **maximise the information conveyed** within **limited canvas space**. Additionally, with the abundance of hierarchical visualisations available today, the challenge lies in selecting **the most appropriate** one for the data, as well as determining **the best method** for visualising multivariate information without causing clutter on the layouts.

Furthermore, this type of data requires interactive elements to facilitate analysis, as visualisations alone may not suffice due to the complexity of the data. Also, interactions are essential for addressing **visualisation tasks**, which often require interactive elements to be effectively solved. Moreover, interactions can also pose issues, as they may impose a **high cognitive load** on users due to the numerous elements on the user interface to control, such as filters. To address these issues,

Visual Natural Language Interfaces (V-NLIs, also referred to in this dissertation as VisChatbots) have been developed, aiding users in analysis through natural language. Although V-NLIs are rapidly evolving, particularly due to advancements in natural language processing and generative language models, they still encounter difficulties, such as resolving ambiguities, understanding the context of the visual analytic conversation, misinterpretations, and the relationship between visualisations and the underlying data.

In this thesis, we investigated the interconnected fields of **Visualisation of Hierarchical Multivariate Data** and **Visualisation-oriented Natural Language Interfaces** to contribute to a more effective analysis and interaction with hierarchical multivariate datasets. To achieve this, we established two objectives (see Section 1.4). The first objective, **O1-Vis**, pertains to the improvement of hierarchical multivariate visualisations, emphasising the identification of the most informative layouts and techniques to effectively depict complex data structures and multivariate attributes. In contrast, the second objective, **O2-VisChatbot**, involves the creation and demonstration of a design methodology for a VisChatbot, focusing on the exploration and integration of chatbots with data visualisation techniques.

Regarding the first objective, **O1-Vis**, we proposed a categorisation algorithm to classify hierarchies in order to select the most informative layout in the visual mapping. To validate our proposal we developed a visualisation platform called DViL (Data Visualisation in Linguistics) that integrates this algorithm to automatically select the most informative layout. The DViL platform also incorporates various interactive elements, such as filters and interactive complementary charts, and visual representations, such as multivariate data glyphs, to enhance user engagement and facilitate data exploration.

Specifically, in Chapter 2, we laid the foundation for understanding hierarchical multivariate data by introducing the Data Visualisation Pipeline (DataVis Pipeline), encompassing three essential spaces: Data Space, Visual Space, and Interaction Space. These spaces serve as the framework for defining the key vocabulary and principles of data visualisation. Through a comprehensive literature review, we examined three core areas: the visualisation of network and hierarchical data, multivariate data, and conversational data, identifying both the contributions of prior research and the gaps that remain in each domain. Notably, in hierarchical visualisation, we highlighted that previous approaches have predominantly focused on a single layout strategy, overlooking the necessity of flexibility. Our analysis underscored that no singular layout is universally optimal—successful visualisations must harmonise aesthetic considerations, clarity, and scalability to maximise information displayed without compromising context or creating clutter. For multivariate data visualisation, we explored a variety of techniques, including interaction-driven methods, hybrid approaches, and glyph-based, icon-based, and animation-based visualisations, each contributing distinct advantages for representing and interpreting complex datasets. These insights emphasise the importance of adapting visualisation strategies to the specific demands of the data, a theme that underpins the conclusions drawn throughout the chapter.

In Chapter 3, we proposed a novel categorisation algorithm to classify hierarchical structures as Elongated or Compact, depending on defined key attributes such as the growth factor and the number of direct children of a node. This algorithm is versatile and can be applied to any hierarchical dataset, providing a systematic approach to structure classification. Our analysis confirmed the algorithm's efficiency, and we further explored the suitability of various visualisation layouts for each category. Specifically, we proposed that Tree and Circle layouts are the most informative for Elongated structures, while Radial and Force layouts offer clearer representations for Compact structures. Despite the potential of this categorisation, we identified certain limitations that suggest enhancement for further development. Additional categories, such as n-compact structures (consisting of multiple compact structures) or hybrid configurations (where Elongated and Compact structures coexist), could be incorporated, thereby increasing the algorithm's flexibility and enhancing its applicability to more complex datasets. Moreover, we formalised the features of multivariate data visualised within hierarchical structures, developing guidelines for visualising different multivariate data types. For ordinal data, we argued that direct representation using hue colours on nodes or edges is most effective, as it conveys clear, distinguishable values (e.g., small, medium, large). In contrast, nominal data is best represented using icons for concrete features, while abstract features can be conveyed through glyphs mapped to unique colours. These formalised approaches ensure more intuitive and accurate visual representations of complex multivariate data embedded in hierarchical structures.

In Chapter 4, we presented our visualisation platform, Data Visualisation in Linguistics (DViL), which is designed to visualise hierarchical multivariate data. Our goal was to create a visualisation of hierarchical structures that facilitates a clear, effective, and efficient analysis of both parent-child relationships and feature distributions, ensuring a user-friendly experience that minimises cognitive overload while maintaining clarity. DViL not only achieves this but also offers adaptability, making it suitable for visualising any hierarchical multivariate dataset. To validate our proposed algorithm in DViL, we conducted an in-depth case study on hate speech data collected from comment sections of online newspapers, demonstrating the potential of DViL in real-world applications. This case study highlighted the flexibility and effectiveness of our platform in representing complex datasets while enabling users to easily explore and understand both the structural and feature-based relationships within the data.

Moreover, we conducted a user evaluation using this case study to validate the categorisation introduced in Chapter 3. It should be noted that user feedback from our evaluation indicated that DViL was well-received, with participants noting that it was easy to learn and navigate. This ease of use contributed to participants' ability to explore and interpret the data effectively, thereby reinforcing the platform's accessibility and reducing cognitive strain. In relation to the proposed layout selection, while most users found the Tree layout to be the most intuitive for both Elongated and Compact data, there is strong evidence suggesting that Force and Radial layouts offer distinct advantages for broader datasets, particularly in visualising Compact structures. Interestingly, user feedback indicated that these alternative layouts required more time for participants to fully understand, which may have contributed to the overall preference for the Tree layout. This insight points to a potential learning curve associated with the more complex layouts, highlighting an area for future improvement in user training, interface design, or the design of those layouts to maximise their effectiveness.

Considering the second objective, **O2-VisChatbot**, we developed a VisChatbot methodology to establish a standardised approach for creating V-NLI. This methodology was used to create our VisChatbot, which we integrated into our visualisation platform which was evaluated with users.

In Chapter 5, we introduced the V-NLI pipeline, an extension of the DataVis Pipeline. Similarly, we explored the key vocabulary related to V-NLI and presented a literature review examining the synergies between the fields of data visualisation and natural language interfaces, highlighting gaps and contributions in the field. Our analysis revealed that there are limited studies concerning complex data and advanced visualisations in V-NLIs, as well as gaps in using high-level queries and guidance strategies. Additionally, we revealed the gap in the methodologies to design VisChatbots, while there are several design methodologies for creating chatbots, none specifically address the unique and complex requirements of VisChatbots.

To fill this critical gap in the design methodology for VisChatbots, in Chapter 6, we proposed a new methodology called Visualisation Chatbot (VisChat Methodology), which aims to guide the incremental creation of VisChatbots for visual analytics processes. This iterative approach encompasses three main classical phases of the development process: Analysis, Design, and Development. We outlined several key stages for each phase. The Analysis phase includes stages; User, Data, VisTask (visualisation task), Visualisation Type, and Interaction Requirements analyses, ensuring a deep understanding of both user needs and data characteristics. In the Design phase we contributed with three design artefacts; VisAgent Persona, Visualisation Conversation Patterns and Annotated Transcripts, which are derived from the Analysis phase and aimed at guiding the conversational flow and interaction design of the VisChatbot. Finally, the Development phase includes VisChatbot Modelling, Visualisation-Chatbot Connection and User Evaluation stages, facilitating the smooth implementation and the continuous development of the VisChatbot.

Our findings demonstrated that this structured, phase-based approach enhances the development process, making it more systematic and adaptable to varying contexts. However, we also identified some key limitations. For example, **the order of data analysis and task analysis** is closely linked and interchangeable, depending on whether the methodology is data-driven or task-driven. This relationship warrants additional consideration in future iterations, as the order could significantly influence the design outcomes. Moreover, we suggest that additional **Visualisation Conversation Patterns** could be incorporated to address more specific user needs, VisTasks, data types, and interaction modalities. For instance, tasks such as **Request Visualisation Task and Query** and **Request Visualisation Details** could be further refined to support distinct VisTasks like **LocateT**, **LookupT**, **BrowseT**, and **ExploreT**, as well as, VisQueries such as **IdentifyQ**, **CompareQ**, and **SummariseQ**.

Overall, our proposed VisChat Methodology contributes a robust, flexible framework for the iterative development of VisChatbots, filling a critical shortcoming in current methodologies. The findings from our work highlight not only the utility of this structured approach but also the need for continued refinement, particularly in tailoring the methodology to different analytic contexts and in developing better tools for evaluating VisChatbot usability.

In Chapter 7, we applied the VisChat Methodology presented in Chapter 6 to design our VisChatbot and integrate it within the DViL platform. We demonstrated how we utilised each phase in developing our VisChatbot, systematically guiding its creation through the methodology. This hands-on application not only validated the methodology but also showcased its flexibility and adaptability in real-world use cases. Specifically, we presented a user evaluation to assess the performance of our VisChatbot with users. Our results are promising, as users were able to complete each VisTask, demonstrating the VisChatbot's capability to facilitate efficient visual analytics interactions. As users became more familiar with the VisChatbot, their performance improved, underscoring the effectiveness of our iterative design approach in enhancing usability over time. These findings highlight the potential of VisChatbots to improve user engagement and streamline complex data VisTasks, particularly in environments where hierarchical multivariate data needs to be explored. Moreover, our results indicated that the VisChatbot effectively reduced the cognitive load associated with traditional data exploration methods. Users appreciated the VisChatbot's conversational interface, which provided a more intuitive and interactive way to navigate VisTasks. This reinforces the importance of integrating chatbots into visual analytics platforms like DViL, offering a more user-friendly alternative to manual interaction methods. Despite these encouraging findings, our study has some limitations. Our case study is limited to a single iteration of the design; therefore, we recommend conducting experiments with different iterations. This can be achieved by revisiting the Development Stage or returning to the Analysis Stage, completing either a half-circle or a full circle of the entire methodology. Additionally, our preliminary evaluation was conducted with a relatively small sample of 16 participants. Future studies should involve a larger and more diverse group of users to validate the generalisability of our findings and also to gain more comprehensive insights into the VisChatbot's usability.

Finally, it should be noted that this thesis contributes to the 2030 Agenda for Sustainable Development [87], particularly Goal 4 (Quality Education), by democratising access to data through the VisChatbot, which enables a wider audience, including those without technical expertise, to gain insights via natural conversations. Additionally, both DViL and VisChatbot, designed for non-expert analysis, align with Goal 17 (Partnerships for the Goals) by fostering collaboration across diverse groups. Moreover, our case study on hate speech in online news furthers Goals 16 (Peace, Justice, and Strong Institutions), 10 (Reduced Inequalities), and 5 (Gender Equality) by helping analysts explore complex data to identify patterns related to discrimination and bias, enabling critical insights into inequality.

8.2 Future Work

This research lays the foundation for a wide range of future investigations in different fields. Concerning our first objective **O1-Vis**, we focused on developing a categorisation algorithm aimed at selecting the most informative layouts for visualising hierarchical data. In this dissertation, we introduced two primary categories, "Elongated" and "Compact", which provide a foundation for classifying hierarchical structures. However, we acknowledge that these categories represent only an initial step, and there is significant potential to expand this framework further. Future categorisations could introduce new classes that better reflect the diversity of hierarchical structures and their subtrees, considering not only the arrangement of the hierarchy itself but also how features within the hierarchy are visually represented. For example, we presented two approaches for visual mappings of multivariate data: individual features can be mapped one-by-one, and displayed all-in-one using glyphs, which offers a more compact representation. By incorporating these visual elements into the categorisation process, our algorithm could evolve to account for both structural and feature-based complexities, offering a more comprehensive approach to hierarchical and multivariate data visualisation.

Moreover, we focused primarily on hierarchical visualisations such as Tree, Radial, Force, and Circle Packing layouts, which are well-suited for representing parent-child relationships in multivariate data. However, there is potential to explore new visual metaphors that could further enrich the analytical experience. For instance, temporal visualisations could be particularly useful in scenarios where time-based data is a significant factor, enabling users to track changes over time within the hierarchical structure.

Additionally, there is room to create more versatile, general-purpose glyphs that could be adapted to various contexts as we developed application-specific glyphs in our current implementation. These new glyphs could provide more flexibility in representing a broader range of data types and relationships within the hierarchy, allowing for more intuitive visual mapping and interaction with complex datasets.

We implemented a rule-based decision-making process to guide the selection of the most informative layout for hierarchical data visualisation. However, this rule-based system could be enhanced by adopting a Machine Learning approach, specifically leveraging reinforcement learning [198] by using user interactions and preferences during the visual analysis. In this scenario, the visualisation selection process could become more dynamic, as the system learns from user interactions and feedback to better tune the visual representation of each dataset over time. The initial state of this Machine Learning model would still be guided by our established taxonomy, ensuring that the system starts with well-founded principles for hierarchical data classification. As the model evolves, it could refine its decision-making process, allowing for more personalised and context-sensitive visualisations that adapt to the specific needs and characteristics of each dataset. This shift from rule-based to Machine Learning would not only increase flexibility but also enable more intelligent and adaptive visualisation strategies that respond to user preferences and data complexities.

Regarding our second objective, **O2-VisChatbot**, the field of VisChatbots presents a promising yet evolving area of research, with significant potential for future developments. In this thesis, we introduced a VisChatbot methodology to address the existing gap in integrating chatbots with visualisations. In the Analysis Phase of our methodology, we focus on refining and defining the appropriate vocabulary for the visualisations. With advancements in chatbot technologies such as

LLMs, the need for manually curating extensive vocabulary lists is significantly reduced. Recent advances in chatbot technologies, such as ChatGPT-4 [138] and Gemini [188], showcase their ability to efficiently respond to complex visual queries. These developments minimise the necessity for explicitly teaching chatbots specialised visualisation vocabulary, making them more adaptable to diverse user inputs. By relying less on extensive training datasets, these advancements enhance the overall development process, making VisChatbots more intuitive, efficient, and effective in supporting data analysis.

In the Design Phase, we presented several patterns that provide a framework for understanding VisQueries within a VisTask, while highlighting their potential for expansion. For instance, the Request Visualisation Task and Query pattern can be enhanced by creating distinct patterns for each type of VisTask and VisQuery. Similarly, the Request Visualisation Details pattern, which currently integrates multimodal input and output, can be refined into more specific patterns that first address these modalities separately, and then together. This separation would allow for a clearer representation of user-VisChatbot interactions and ensures that the VisChatbot can adapt to emerging needs in data visualisation.

Furthermore, to enhance storytelling capabilities within the VisChatbot interface, such as a study [169] that explored high-level queries with narrative visualisations, a new Visual Narrative pattern can be defined inspired by the Visual summary pattern. The new Visual Narrative pattern can be built on the Visual Summary pattern by not only summarising key data insights but also presenting them as a coherent story. Unlike the Visual Summary, which offers a concise overview, the Visual Narrative weaves data into a structured sequence, providing context and guidance throughout. This approach would help users understand the broader story behind the data, offering a more comprehensive and engaging exploration of insights.

In the Development Phase, we utilised Rasa as a first approach for our VisChatbot. However, the integration of cutting-edge technologies like large language models (LLMs) presents significant opportunities for enhancing visualisation capabilities. While many existing tools excel at handling either LLMs or visualisations independently, few offer a truly seamless and integrated solution that combines both functionalities. Libraries such as Lida [46], Plotly Chatbot Builder [55], and Chat2Vis [122] have made commendable strides in this direction, paving the way for more effective interaction between users and data visualisations through LLMs.

Recent research has explored the readiness of LLMs for visual analytics, exploring the potential benefits of this integration while also acknowledging the challenges that remain. The use of LLMs can significantly enhance the interpretability and usability of visualisations by enabling more natural language interactions, thereby allowing users to query and manipulate data intuitively. Despite the unresolved issues that still need to be tackled, there are promising advancements in this area to maximise the capabilities of both LLMs and visualisations [192].

In this phase, we also proposed a comprehensive set of both user-centred and chatbot-centred metrics to evaluate the usability, effectiveness, and engagement of the VisChatbot design. However, the existing literature lacks dedicated usability questionnaires for VisChatbots, presenting an ongoing challenge for the development and validation of such tools. Addressing this gap could significantly improve the evaluation process and ensure that VisChatbots meet user expectations in terms of functionality and interaction quality. Additionally, the potential fatigue induced by lengthy data analyses poses another obstacle, highlighting the need for VisChatbots to assume an evaluator role [152], collecting user feedback seamlessly during the visual analytics experience.

Furthermore, we sought to explore gaps, particularly in using VisChatbots with complex data and advanced visualisations. There remains much to investigate. One potential future direction is addressing how complex data is often projected into a two-dimensional space, limiting queries on intricate structures such as multivariate hierarchical and network data, which may be better explored in three-dimensional spaces [28]. The immersive environments such as VR and AR, can enhance the exploration of multivariate hierarchical data by allowing users to manipulate and interact with the data spatially, making complex relationships more apparent. For example, users could navigate through a three-dimensional hierarchical structure, rotating and zooming in on different levels to uncover insights that would be challenging to discern in a static, two-dimensional representation. Thus, designing V-NLIs to accommodate high-level queries and extending their study beyond traditional WIMP interfaces, such as incorporating immersive analytics in VR and AR environments [110], represents a promising direction for further exploration.

Moreover, to let users better express their intents with less ambiguity, V-NLI systems use either guidance strategies or multimodality. While our systems incorporate multimodality through natural language and WIMP interfaces, it is essential that these modalities communicate effectively with one another. While most systems facilitate user-chatbot interactions through WIMP interfaces, it is noteworthy that few support touch interactions [108, 174, 175, 178], and there is limited exploration into the use of gestures [95]. This indicates significant opportunities for improvement, particularly in relation to multimodality [155], which can enhance the transformation of data by enabling users to interact with the system in a more comprehensive manner—using not only text and voice but also gestures and gaze. Furthermore, multimodality can support collaborative analysis of visualisations and serve as an additional input for the NLP system, enriching the context during analytical conversations.

Additionally, as presented in Chapter 6, we introduced the *'Sniffer'* pattern, which incorporates a passive listening mode that allows the VisChatbot to observe conversations between users and automatically propose relevant interaction methods accordingly [19]. In line with this, recent research has examined an always-listening agent that acts as a third collaborator in multi-person visual analysis, generating visualisations based on observations of users' conversations [185]. The concept of passive listening could be enhanced by incorporating other input signals, such as eye tracking [117] and emotional indicators like tone of voice [5]. These enhancements could be easily integrated into our proposed pattern and significantly improve the guidance provided by the VisChatbot.

Part IV

Bibliography

Bibliography

[1] Ibm watson analytics.

[2] Microsoft power bi.

[3] Eleni Adamopoulou and Lefteris Moussiades. Chatbots: History, technology, and applications. *Machine Learning with Applications*, 2:100006, 2020.

[4] Robert Amar, James Eagan, and John Stasko. Low-level components of analytic activity in information visualization. In *IEEE Symp. on Information Visualization, 2005. INFOVIS 2005.*, pages 111–117. IEEE, 2005.

[5] Amazon Web Services Inc. Amazon lex documentation. `https://docs.aws.amazon.com/lex/index.html`, 2023.

[6] Lakshay Anand and Carlos M Rodriguez Lopez. Chromomap: an r package for interactive visualization of multi-omics data and annotation of chromosomes. *BMC Bioinformatics*, 23(1):33, 2022.

[7] Daniel Angus, Bernadette Watson, Andrew Smith, Cindy Gallois, and Janet Wiles. Visualising conversation structure across time: Insights into effective doctor-patient consultations. *PloS one*, 7(6):e38014, 2012.

[8] Ronald P Anjard. Management and planning tools. *Training for quality*, 1995.

[9] AppleInc. Speech — apple developer. `https://developer.apple.com/documentation/speech`, 2023.

[10] Daniel Archambault, Tamara Munzner, and David Auber. Grouseflocks: Steerable exploration of graph hierarchy space. *IEEE Transactions on Visualization and Computer Graphics*, 14(4):900–913, 2008.

[11] Jillian Aurisano, Abhinav Kumar, Alberto Gonzales, Khairi Reda, Jason Leigh, Barbara Di Eugenio, and Andrew Johnson. Show me data": Observational study of a conversational interface in visual data exploration. In *IEEE VIS*, volume 15, page 1, 2015.

[12] Jillian Aurisano, Abhinav Kumar, Alberto Gonzalez, Jason Leigh, Barbara DiEugenio, and Andrew Johnson. Articulate2: Toward a conversational interface for visual data exploration. In *IEEE Visualization*, 2016.

[13] Mohammed Bahja, Rawad Hammad, and Gibran Butt. A user-centric framework for educational chatbots design and development. In *HCI Int. 2020-Late Breaking Papers: Multimodality and Intelligence: 22nd HCI Int. Conf., HCII 2020, Copenhagen, Denmark, July 19–24, 2020, Proc. 22*, pages 32–43. Springer, 2020.

[14] Mathieu Bastian, Sebastien Heymann, and Mathieu Jacomy. Gephi: an open source software for exploring and manipulating networks. In *Proc. of the Int. AAAI Conf. on Web and Social Media*, volume 3, pages 361–362, 2009.

[15] Mary Bates. Health care chatbots are here to help. *IEEE pulse*, 10(3):12–14, 2019.

[16] Samuel Beck, Sebastian Frank, Alireza Hakamian, Leonel Merino, and André van Hoorn. Transvis: Using visualizations and chatbots for supporting transient behavior in microservice systems. In *2021 Working Conf. on Software Visualization (VISSOFT)*, pages 65–75. IEEE, 2021.

[17] Yonatan Belinkov and James Glass. Analysis methods in neural language processing: A survey. *Transactions of the Association for Computational Linguistics*, 7:49–72, 2019.

[18] Anastasia Bezerianos, Fanny Chevalier, Pierre Dragicevic, Niklas Elmqvist, and Jean-Daniel Fekete. Graphdice: A system for exploring multivariate social networks. In *Computer Graphics Forum*, volume 29, pages 863–872. Wiley Online Library, 2010.

[19] Stefan Bieliauskas and Andreas Schreiber. A conversational user interface for software visualization. In *2017 IEEE Working Conf. on Software Visualization (vissoft)*, pages 139–143. IEEE, 2017.

[20] Steven Bird. Nltk: the natural language toolkit. In *Proc. of the COLING/ACL 2006 Interactive Presentation Sessions*, pages 69–72, 2006.

[21] Sanat Kumar Bista, Surya Nepal, and Cécile Paris. Multifaceted visualisation of annotated social media data. In *2014 IEEE International Congress on Big Data*, pages 699–706. IEEE, 2014.

[22] José Borges. Visac: An interactive tool for visual analysis of consanguinity in the ancestry of individuals. *Information Visualization*, page 14738716221096383, 2022.

[23] Rita Borgo, Johannes Kehrer, David HS Chung, Eamonn Maguire, Robert S Laramee, Helwig Hauser, Matthew Ward, and Min Chen. Glyph-based visualization: Foundations, design guidelines, techniques and applications. In *Eurographics (State of the Art Reports)*, pages 39–63, 2013.

[24] Katy Börner, Andreas Bueckle, and Michael Ginda. Data visualization literacy: Definitions, conceptual frameworks, exercises, and assessments. *Proc. of the National Academy of Sciences*, 116(6):1857–1864, 2019.

[25] Simone Borsci, Alessio Malizia, Martin Schmettow, Frank Van Der Velde, Gunay Tariverdiyeva, Divyaa Balaji, and Alan Chamberlain. The chatbot usability scale: the design and pilot of a usability scale for interaction with ai-based conversational agents. *Personal and Ubiquitous Computing*, 26:95–119, 2022.

[26] Matthew Brehmer and Tamara Munzner. A multi-level typology of abstract visualization tasks. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2376–2385, 2013.

[27] Kjell Brunnström and Marcus Barkowsky. Statistical quality of experience analysis: on planning the sample size and statistical significance testing. *Journal of Electronic Imaging*, 27(5):053013, 2018.

[28] Michael Burch, Adrian Vramulet, Alex Thieme, Alina Vorobiova, Denis Shehu, Mara Miulescu, Mehrdad Farsadyar, and Tar van Krieken. Vizwick: a multiperspective view of hierarchical data. In *Proc. of the 13th Int. Symp. on Visual Information Communication and Interaction*, pages 1–5, 2020.

[29] Noel Bush, R Wallace, T Ringate, A Taylor, and J Baer. Artificial intelligence markup language (aiml) version 1.0. 1. *ALICE AI Foundation Working Draft*, 2001.

[30] G Cameron, D Cameron, G Megaw, RR Bond, M Mulvenna, S O'Neill, and M McTear. Back

to the future: Lessons from knowledge engineering methodologies for chatbot design and development. In *Proc. of the 32nd Int. BCS HCI Conf*, 2018.

[31] Nan Cao, Yu-Ru Lin, Fan Du, and Dashun Wang. Episogram: Visual summarization of egocentric social interactions. *IEEE Computer Graphics and Applications*, 36(5):72–81, 2015.

[32] Nan Cao, Jimeng Sun, Yu-Ru Lin, David Gotz, Shixia Liu, and Huamin Qu. Facetatlas: Multifaceted visualization for rich text corpora. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1172–1181, 2010.

[33] Arthur TE Capozzi, Viviana Patti, Giancarlo Ruffo, and Cristina Bosco. A data viz platform as a support to study, analyze and understand the hate speech phenomenon. In *Proceedings of the 2nd International Conference on Web Studies*, pages 28–35, 2018.

[34] Mackinlay Card. *Readings in information visualization: using vision to think*. Morgan Kaufmann, 1999.

[35] Jacky Casas, Marc-Olivier Tricot, Omar Abou Khaled, Elena Mugellini, and Philippe Cudré-Mauroux. Trends & methods in chatbot evaluation. In *Companion Publication of the 2020 International Conference on Multimodal Interaction*, pages 280–286, 2020.

[36] Justine Cassell. Embodied conversational interface agents. *Communications of the ACM*, 43(4):70–78, 2000.

[37] Christian Chabot, Chris Stolte, and Pat Hanrahan. Tableau software. *Tableau Software*, 6, 2003.

[38] Se-Hang Cheong and Yain-Whar Si. Force-directed algorithms for schematic drawings and placement: A survey. *Information Visualization*, 19(1):65–91, 2020.

[39] Imran Chowdhury, Abdul Moeid, Enamul Hoque, Muhammad Ashad Kabir, Md Sabir Hossain, and Mohammad Mainul Islam. Miva: Multimodal interactions for facilitating visual analysis with multiple coordinated views. In *2020 24th Int. Conf. Information Visualisation (IV)*, pages 714–717. IEEE, 2020.

[40] Leonardo Christino and Fernando V Paulovich. Chatkg: Visualizing temporal patterns as knowledge graph. In *EuroVA@ EuroVis*, pages 13–18, 2023.

[41] NCBI Resource Coordinators. Database resources of the national center for biotechnology information. *Nucleic Acids Research*, 45(Database issue):D12, 2017.

[42] Kenneth Cox, Rebecca E Grinter, Stacie L Hibino, Lalita Jategaonkar Jagadeesan, and David Mantilla. A multi-modal natural language interface to an information visualization environment. *Int. Journal of Speech Technology*, 4(3):297–314, 2001.

[43] Pietro Crovari, Sara Pidò, Pietro Pinoli, Anna Bernasconi, Arif Canakoglu, Franca Garzotto, and Stefano Ceri. Gecoagent: a conversational agent for empowering genomic data extraction and analysis. *ACM Transactions on Computing for Healthcare (HEALTH)*, 3(1):1–29, 2021.

[44] Youssef Darzi, Yuta Yamate, and Takuji Yamada. Functree2: an interactive radial tree for functional hierarchies and omics data visualization. *Bioinformatics*, 35(21):4519–4521, 2019.

[45] Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. Automated hate speech detection and the problem of offensive language. *arXiv preprint arXiv:1703.04009*, 2017.

[46] Victor Dibia. Lida: A tool for automatic generation of grammar-agnostic visualizations and infographics using large language models. *arXiv preprint arXiv:2303.02927*, 2023.

[47] Evanthia Dimara and Charles Perin. What is interaction for data visualization? *IEEE Transactions on Visualization and Computer Graphics*, 26(1):119–129, 2019.

[48] Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, et al. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature Machine Intelligence*, 5(3):220–235, 2023.

[49] Judith Donath. A semantic approach to visualizing online conversations. *Communications of the ACM*, 45(4):45–49, 2002.

[50] Yu Dong, Alex Fauth, Maolin Huang, Yi Chen, and Jie Liang. Pansytree: Merging multiple hierarchies. In *2020 IEEE Pacific Visualization Symposium (PacificVis)*, pages 131–135. IEEE, 2020.

[51] Fan Du, Catherine Plaisant, Neil Spring, and Ben Shneiderman. Visual interfaces for recommendation systems: Finding similar and dissimilar peers. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(1):1–23, 2018.

[52] Hamza El Alaoui, Zakaria El Aouene, and Violetta Cavalli-Sforza. Building intelligent chatbots: Tools, technologies, and approaches. In *2023 3rd Int. Conf. on Innovative Research in Applied Science, Engineering and Technology (IRASET)*, pages 1–12. IEEE, 2023.

[53] Mennatallah El-Assady, Valentin Gold, Carmela Acevedo, Christopher Collins, and Daniel Keim. Contovi: Multi-party conversation exploration using topic-space views. In *Computer Graphics Forum*, volume 35, pages 431–440. Wiley Online Library, 2016.

[54] Mennatallah El-Assady, Rita Sevastjanova, Bela Gipp, Daniel Keim, and Christopher Collins. Nerex: Named-entity relationship exploration in multi-party conversations. In *Computer Graphics Forum*, volume 36, pages 213–225. Wiley Online Library, 2017.

[55] Dash Enterprise. Plotly chatbot builder, 2024. Accessed: 27 October 2024.

[56] Siamak Faridani, Ephrat Bitton, Kimiko Ryokai, and Ken Goldberg. Opinion space: a scalable tool for browsing online comments. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1175–1184, 2010.

[57] Ethan Fast, Binbin Chen, Julia Mendelsohn, Jonathan Bassen, and Michael S Bernstein. Iris: A conversational agent for complex tasks. In *Proc. of the 2018 CHI Conf. on Human Factors in Computing Systems*, pages 1–12, 2018.

[58] Jasper Feine, Stefan Morana, and Alexander Maedche. Designing interactive chatbot development systems. In *ICIS*, 2020.

[59] Yingchaojie Feng, Xingbo Wang, Bo Pan, Kam Kwai Wong, Yi Ren, Shi Liu, Zihan Yan, Yuxin Ma, Huamin Qu, and Wei Chen. Xnli: Explaining and diagnosing nli-based visual data analysis. *IEEE Transactions on Visualization and Computer Graphics*, 2023.

[60] Paula Fortuna and Sérgio Nunes. A survey on automatic detection of hate speech in text. *ACM Computing Surveys (CSUR)*, 51(4):1–30, 2018.

[61] Tom C Freeman, Sebastian Horsewell, Anirudh Patir, Josh Harling-Lee, Tim Regan, Barbara B Shih, James Prendergast, David A Hume, and Tim Angus. Graphia: A platform for the graph-based visualisation and analysis of high dimensional data. *PLoS Computational Biology*, 18(7):e1010310, 2022.

[62] Siwei Fu, Yong Wang, Yi Yang, Qingqing Bi, Fangzhou Guo, and Huamin Qu. Visforum: A visual analysis system for exploring user groups in online forums. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 8(1):1–21, 2018.

[63] Siwei Fu, Jian Zhao, Weiwei Cui, and Huamin Qu. Visual analysis of mooc forums with iforum. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):201–210, 2016.

[64] Florian Ganglberger, Nicolas Swoboda, Lisa Frauenstein, Joanna Kaczanowska, Wulf Haubensak, and Katja Bühler. Braintrawler: A visual analytics framework for iterative exploration of heterogeneous big brain data. *Computers & Graphics*, 82:304–320, 2019.

[65] Tong Gao, Mira Dontcheva, Eytan Adar, Zhicheng Liu, and Karrie G Karahalios. Datatone: Managing ambiguity in natural language interfaces for data visualization. In *Proc. of the 28th Annual ACM Symp. on User Interface Software & Technology*, pages 489–500, 2015.

[66] Albert Gatt and Emiel Krahmer. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *Journal of Artificial Intelligence Research*, 61:65–170, 2018.

[67] Google Cloud. Dialogflow. `https://cloud.google.com/dialogflow/`, 2023.

[68] David Gotz and Michelle X. Zhou. Characterizing users' visual analytic activity for insight provenance. In *2008 IEEE Symposium on Visual Analytics Science and Technology*, pages 123–130, 2008.

[69] Lars Grammel, Melanie Tory, and Margaret-Anne Storey. How information visualization novices construct visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):943–952, 2010.

[70] Christine Grové. Co-developing a mental health and wellbeing chatbot with and for young people. *Frontiers in Psychiatry*, 11:606041, 2021.

[71] Jochen Görtler, Christoph Schulz, Daniel Weiskopf, and Oliver Deussen. Bubble treemaps for uncertainty visualization. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):719–728, 2018.

[72] Pat Hanrahan. Vizql: a language for query, analysis and visualization. In *Proc. of the 2006 ACM SIGMOD Int. Conf. on Management of Data*, pages 721–721, 2006.

[73] Maria CR Harrington, Zack Bledsoe, Chris Jones, James Miller, and Thomas Pring. Designing a virtual arboretum as an immersive, multimodal, interactive, data visualization virtual field trip. *Multimodal Technologies and Interaction*, 5(4):18, 2021.

[74] Marti Hearst and Melanie Tory. Would you like a chart with that? incorporating visualizations into conversational interfaces. In *2019 IEEE Visualization Conf. (VIS)*, pages 1–5. IEEE, 2019.

[75] Jennifer Hehn, Falk Uebernickel, and Matthias Herterich. Design thinking methods for service innovation-a delphi study. 2018.

[76] Andrew R Hippisley, Mariam Tariq, and David Chang. Hierarchical data and the derivational relationship between words. 2001.

[77] Jane Hoffswell, Wilmot Li, and Zhicheng Liu. Techniques for flexible responsive visualization design. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–13, 2020.

[78] Matthew Honnibal and Ines Montani. spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing. *To appear*, 7(1):411–420, 2017.

[79] Enamul Hoque and Giuseppe Carenini. Convis: A visual text analytic system for exploring blog conversations. In *Computer Graphics Forum*, volume 33, pages 221–230. Wiley Online Library, 2014.

[80] Enamul Hoque, Parsa Kavehzadeh, and Ahmed Masry. Chart question answering: State of the art and future directions. *arXiv preprint arXiv:2205.03966*, 2022.

[81] Enamul Hoque, Vidya Setlur, Melanie Tory, and Isaac Dykeman. Applying pragmatics principles for interaction with visual analytics. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):309–318, 2017.

[82] Chen-Chi Hu, Hao-Xiang Wei, and Ming-Te Chi. Shareflow: A visualization tool for information diffusion in social media. In *International Conference on Ubiquitous Information Management and Communication*, pages 563–581. Springer, 2019.

[83] Kevin Hu, Michiel A. Bakker, Stephen Li, Tim Kraska, and César Hidalgo. VizML: A Machine Learning Approach to Visualization Recommendation. In *Proc. of the 2019 Conf. on Human Factors in Computing Systems (CHI)*. ACM, 2019.

[84] Jieying Huang, Yang Xi, Junnan Hu, and Jun Tao. Flownl: Asking the flow data in natural languages. *IEEE Transactions on Visualization and Computer Graphics*, 2022.

[85] Yanwei Huang, Yunfan Zhou, Ran Chen, Changhao Pan, Xinhuan Shu, Di Weng, and Yingcai Wu. Interactive table synthesis with natural language. *IEEE Transactions on Visualization and Computer Graphics*, pages 1–16, 2023.

[86] Yichen Huang, Yuchen Wang, and Yik-Cheung Tam. Uniter-based situated coreference resolution with rich multimodal input. *arXiv preprint arXiv:2112.03521*, 2021.

[87] Winfried Huck. Transforming our world: the 2030 Agenda for Sustainable Development. In *Sustainable Development Goals*. Nomos/Hart, 3 2023.

[88] IBM. Ibm watson assistant. `https://www.ibm.com/products/watson-assistant`, 2023.

[89] Md Rafiqul Islam, Imran Razzak, Xianzhi Wang, Peter Tilocca, and Guandong Xu. Natural language interactions enhanced by data visualization to explore insurance claims and manage risk. *Annals of Operations Research*, pages 1–19, 2022.

[90] Sirkka L Jarvenpaa. The effect of task demands and graphical format on information processing strategies. *Management Science*, 35(3):285–303, 1989.

[91] Rogers Jeffrey Leo John, Navneet Potti, and Jignesh M Patel. Ava: From data to insights through conversations. In *CIDR*, 2017.

[92] Shafiq Joty, Giuseppe Carenini, and Raymond T Ng. Topic segmentation and labeling in asynchronous conversations. *Journal of Artificial Intelligence Research*, 47:521–573, 2013.

[93] Plinio Thomaz Aquino Junior and Lucia Vilela Leite Filgueiras. User modeling with personas. In *Proceedings of the 2005 Latin American Conference on Human-computer interaction*, pages 277–282, 2005.

[94] Dietrich Kammer, Mandy Keck, Thomas Gründer, Alexander Maasch, Thomas Thom, Martin Kleinsteuber, and Rainer Groh. Glyphboard: Visual exploration of high-dimensional data

combining glyphs with dimensionality reduction. *IEEE Transactions on Visualization and Computer Graphics*, 26(4):1661–1671, 2020.

[95] Jan-Frederik Kassel and Michael Rohs. Valletto: A multimodal interface for ubiquitous visual analytics. In *Extended Abstracts of the 2018 CHI Conf. on Human Factors in Computing Systems*, pages 1–6, 2018.

[96] Ecem Kavaz, Anna Puig, and Inmaculada Rodríguez. Chatbot-based natural language interfaces for data visualisation: A scoping review. *Applied Sciences*, 13(12):7025, 2023.

[97] Ecem Kavaz, Anna Puig, Inmaculada Rodríguez, Reyes Chacón, David De-La-Paz, Adrià Torralba, Montserrat Nofre, and Mariona Taule. Visualisation of hierarchical multivariate data: Categorisation and case study on hate speech. *Information Visualization*, page 14738716221120509, 2022.

[98] Daniel A Keim. Visual techniques for exploring databases. In *Knowledge Discovery in Databases (KDD'97)*, 1997.

[99] Daniel A Keim and H-P Kriegel. Visualization techniques for mining large databases: A comparison. *IEEE Transactions on Knowledge and Data Engineering*, 8(6):923–938, 1996.

[100] Daniel A Keim and Jörn Schneidewind. Scalable visual data exploration of large data sets via multiresolution. *Journal of Universal Computer Science*, 11(11):1766–1779, 2005.

[101] Spencer Kelly. Compromise. `http://compromise.cool/`, 2019.

[102] Andreas Kerren, Helen Purchase, and Matthew O Ward. Multivariate network visualization (proc. dagstuhl seminar 13201), 2014.

[103] Andreas Kerren, Helen C Purchase, and Matthew O Ward. Introduction to multivariate network visualization. In *Multivariate Network Visualization*, pages 1–9. Springer, 2014.

[104] Kimiya Keyvan and Jimmy Xiangji Huang. How to approach ambiguous queries in conversational search: A survey of techniques, approaches, tools, and challenges. *ACM Computing Surveys*, 55(6):1–40, 2022.

[105] Muzammil Khan and Sarwar Shah Khan. Data and information visualization methods, and interactive mechanisms: A survey. *Int. Journal of Computer Applications*, 34(1):1–14, 2011.

[106] Diksha Khurana, Aditya Koli, Kiran Khatter, and Sukhdev Singh. Natural language processing: State of the art, current trends and challenges. *Multimedia Tools and Applications*, pages 1–32, 2022.

[107] Diksha Khurana, Aditya Koli, Kiran Khatter, and Sukhdev Singh. Natural language processing: State of the art, current trends and challenges. *Multimedia Tools and Applications*, 82(3):3713–3744, 2023.

[108] Young-Ho Kim, Bongshin Lee, Arjun Srinivasan, and Eun Kyoung Choe. Data@ hand: Fostering visual exploration of personal data on smartphones leveraging speech and touch interaction. In *Proc. of the 2021 CHI Conf. on Human Factors in Computing Systems*, pages 1–17, 2021.

[109] Cole Nussbaumer Knaflic. *Storytelling with data: A data visualization guide for business professionals*. John Wiley & Sons, 2015.

[110] Matthias Kraus, Johannes Fuchs, Björn Sommer, Karsten Klein, Ulrich Engelke, Daniel Keim, and Falk Schreiber. Immersive analytics with abstract 3d visualizations: A survey. In *Computer Graphics Forum*, volume 41, pages 201–229. Wiley Online Library, 2022.

[111] Joseph B Kruskal and James M Landwehr. Icicle plots: Better displays for hierarchical clustering. *The American Statistician*, 37(2):162–168, 1983.

[112] Mohammad Amin Kuhail, Nazik Alturki, Salwa Alramlawi, and Kholood Alhejori. Interacting with educational chatbots: A systematic review. *Education and Information Technologies*, pages 1–46, 2022.

[113] Jonathan Lazar, Jinjuan Heidi Feng, and Harry Hochheiser. *Research methods in human-computer interaction*. Morgan Kaufmann, 2017.

[114] Bongshin Lee, Petra Isenberg, Nathalie Henry Riche, and Sheelagh Carpendale. Beyond mouse and keyboard: Expanding design considerations for information visualization interactions. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2689–2698, 2012.

[115] Bongshin Lee, Nathalie Henry Riche, Petra Isenberg, and Sheelagh Carpendale. More than telling a story: Transforming data into visually shared stories. *IEEE Computer Graphics and Applications*, 35(5):84–90, 2015.

[116] Doris Jung-Lin Lee, Abdul Quamar, Eser Kandogan, and Fatma Özcan. Boomerang: Proactive insight-based recommendations for guiding conversational data analysis. In *Proc. of the 2021 Int. Conf. on Management of Data*, pages 2750–2754, 2021.

[117] Fan Li, Ching-Hung Lee, Shanshan Feng, Amy Trappey, and Fazal Gilani. Prospective on eye-tracking-based studies in immersive virtual reality. In *2021 IEEE 24th Int. Conf. on Computer Supported Cooperative Work in Design (CSCWD)*, pages 861–866. IEEE, 2021.

[118] Can Liu, Yun Han, Ruike Jiang, and Xiaoru Yuan. Advisor: Automatic visualization answer for natural-language question on tabular data. In *2021 IEEE 14th Pacific Visualization Symp. (PacificVis)*, pages 11–20. IEEE, 2021.

[119] Yuyu Luo, Nan Tang, Guoliang Li, Jiawei Tang, Chengliang Chai, and Xuedi Qin. Natural language to visualization by neural machine translation. *IEEE Transactions on Visualization and Computer Graphics*, 28(1):217–226, 2021.

[120] Zhengyi Ma, Zhicheng Dou, Yutao Zhu, Hanxun Zhong, and Ji-Rong Wen. One chatbot per person: Creating personalized chatbots based on implicit user profiles. In *Proc. of the 44th Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, pages 555–564, 2021.

[121] Jock Mackinlay, Pat Hanrahan, and Chris Stolte. Show me: Automatic presentation for visual analysis. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1137–1144, 2007.

[122] Paula Maddigan and Teo Susnjak. Chat2vis: Generating data visualisations via natural language using chatgpt, codex and gpt-3 large language models. *arXiv preprint arXiv:2302.02094*, 2023.

[123] Alan Manning and Nicole Amare. Visual design principles and effective sentence diagrams for the 21st century. In *2014 IEEE International Professional Communication Conference (IPCC)*, pages 1–8. IEEE, 2014.

[124] Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. The stanford corenlp natural language processing toolkit. In *Proc. of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, 2014.

[125] Timothy R McIntosh, Teo Susnjak, Tong Liu, Paul Watters, and Malka N Halgamuge. From google gemini to openai q*(q-star): A survey of reshaping the generative artificial intelligence (ai) research landscape. *arXiv preprint arXiv:2312.10868*, 2023.

[126] Andrew M McNutt and Gordon L Kindlmann. Improving the scalability of interactive visualization systems for exploring threaded conversations. In *EuroVis (Posters)*, pages 53–55, 2019.

[127] Michael McTear. *Conversational ai: Dialogue systems, conversational agents, and chatbots*. Morgan & Claypool Publishers, 2020.

[128] Elijah Meeks. *D3. js in Action: Data visualization with JavaScript*. Simon and Schuster, 2017.

[129] Microsoft. Cognitive speech services — microsoft azure. `https://azure.microsoft.com/en-us/products/cognitive-services/speech-services/`,, 2023.

[130] Bonan Min, Hayley Ross, Elior Sulem, Amir Pouran Ben Veyseh, Thien Huu Nguyen, Oscar Sainz, Eneko Agirre, Ilana Heintz, and Dan Roth. Recent advances in natural language processing via large pre-trained language models: A survey. *ACM Computing Surveys*, 56(2):1–40, 2023.

[131] Robert J Moore, Raphael Arar, Guang-Jie Ren, and Margaret H Szymanski. Conversational ux design. In *Proc. of the 2017 CHI Conf. Extended Abstracts on Human Factors in Computing Systems*, pages 492–497, 2017.

[132] Tomas Murillo-Morales and Klaus Miesenberger. Audial: A natural language interface to make statistical charts accessible to blind persons. In *Computers Helping People with Special Needs: 17th Int. Conf., ICCHP 2020, Lecco, Italy, September 9–11, 2020, Proc., Part I 17*, pages 373–384. Springer, 2020.

[133] Arpit Narechania, Arjun Srinivasan, and John Stasko. Nl4dv: A toolkit for generating analytic specifications for data visualization from natural language queries. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):369–379, 2020.

[134] Nitin Newaliya and Yudhvir Singh. Multivariate hierarchical dbscan model for enhanced maritime data analytics. *Data & Knowledge Engineering*, page 102282, 2024.

[135] Carolina Nobre, Miriah Meyer, Marc Streit, and Alexander Lex. The state of the art in visualizing multivariate networks. In *Computer Graphics Forum*, volume 38, pages 807–832. Wiley Online Library, 2019.

[136] OpenAI. Chatgpt. `https://openai.com/models/`, 2021.

[137] OpenAI. Dall-e: A generative model for diverse and creative images. `https://openai.com/dall-e-2/`, 2021.

[138] OpenAI. Gpt-4 technical report, 2023.

[139] Shon Otmazgin, Arie Cattan, and Yoav Goldberg. F-coref: Fast, accurate and easy to use coreference resolution. *arXiv preprint arXiv:2209.04280*, 2022.

[140] Rieko Otsuka, Kazuo Yano, and Nobuo Sato. An organization topographic map for visualizing business hierarchical relationships. In *2009 IEEE Pacific Visualization Symposium*, pages 25–32. IEEE, 2009.

[141] B O'Handley, Yuheng Wu, Haobin Duan, and Chaoli Wang. Treevisual: Design and evaluation of a web-based visualization tool for teaching and learning tree visualization. In *Proceedings of American Society for Engineering Education Annual Conference*, 2022.

[142] Deokgun Park, Mohamed Suhail, Minsheng Zheng, Cody Dunne, Eric Ragan, and Niklas Elmqvist. Storyfacets: A design study on storytelling with visualizations for collaborative data analysis. *Information Visualization*, 21(1):3–16, 2022.

[143] Terence Parr. The definitive antlr 4 reference. *The Definitive ANTLR 4 Reference*, pages 1–326, 2013.

[144] Demetris Paschalides, Dimosthenis Stephanidis, Andreas Andreou, Kalia Orphanou, George Pallis, Marios D. Dikaiakos, and Evangelos Markatos. Mandola: A big-data processing and visualisation platform for monitoring and detecting online hate speech. *ACM Trans. on Internet Technology*, 37(4):1–21, 2019.

[145] Xuedi Qin, Yuyu Luo, Nan Tang, and Guoliang Li. Making data visualization more efficient and effective: a survey. *The VLDB Journal*, 29:93–117, 2020.

[146] Rasa. Rasa conversational platform. `https://rasa.com/`, 2023.

[147] Leon Reicherts, Yvonne Rogers, Licia Capra, Ethan Wood, Tu Dinh Duong, and Neil Sebire. It's good to talk: A comparison of using voice versus screen-based interactions for agent-assisted tasks. *ACM Transactions on Computer-Human Interaction*, 29(3):1–41, 2022.

[148] Ranci Ren, John W Castro, Silvia T Acuña, and Juan de Lara. Evaluation techniques for chatbot usability: A systematic mapping study. *International Journal of Software Engineering and Knowledge Engineering*, 29(11n12):1673–1702, 2019.

[149] Ranci Ren, Mireya Zapata, John W Castro, Oscar Dieste, and Silvia T Acuña. Experimentation for chatbot usability evaluation: A secondary study. *IEEE Access*, 10:12430–12464, 2022.

[150] Paul Resnick, Samuel Carton, Souneil Park, Yuncheng Shen, and Nicole Zeffer. Rumorlens: A system for analyzing the impact of rumors and corrections in social media, 2014.

[151] Rocket.chat. Rocket.chat. `https://www.rocket.chat/`,, 2023.

[152] Eric Roselló-Marín, Maite Lopez-Sanchez, Inmaculada Rodríguez, Manel Rodríguez-Soto, and Juan A Rodríguez-Aguilar. An ethical conversational agent to respectfully conduct in-game surveys. In *Artificial Intelligence Research and Development*, pages 335–344. IOS Press, 2022.

[153] Björn Ross, Michael Rist, Guillermo Carbonell, Benjamin Cabrera, Nils Kurowsky, and Michael Wojatzki. Measuring the reliability of hate speech annotations: The case of the european refugee crisis. *arXiv preprint arXiv:1701.08118*, 2017.

[154] Marcel Ruoff, Brad A Myers, and Alexander Maedche. Onyx-user interfaces for assisting in interactive task learning for natural language interfaces of data visualization tools. In *CHI Conf. on Human Factors in Computing Systems Extended Abstracts*, pages 1–7, 2022.

[155] Ayshwarya Saktheeswaran, Arjun Srinivasan, and John Stasko. Touch? speech? or touch and speech? investigating multimodal interaction for visual network exploration and analysis. *IEEE Transactions on Visualization and Computer Graphics*, 26(6):2168–2179, 2020.

[156] Arvind Satyanarayan, Ryan Russell, Jane Hoffswell, and Jeffrey Heer. Reactive vega: A streaming dataflow architecture for declarative interactive visualization. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):659–668, 2015.

[157] Tobias Schreck, Daniel Keim, and Florian Mansmann. Regular treemap layouts for visual analysis of hierarchical data. In *Proceedings of the 22nd Spring Conference on Computer*

Graphics, SCCG '06, page 183–190, New York, NY, USA, 2006. Association for Computing Machinery.

[158] H-J Schulz and Heidrun Schumann. Visualizing graphs-a generalized view. In *Tenth Int. Conf. on Information Visualisation (IV'06)*, pages 166–173. IEEE, 2006.

[159] Hans-Jorg Schulz. Treevis. net: A tree visualization reference. *IEEE Computer Graphics and Applications*, 31(6):11–15, 2011.

[160] Hans-Jorg Schulz. Treevis.net: A tree visualization reference. *IEEE Computer Graphics and Applications*, 31(6):11–15, 2011.

[161] Hans-Jorg Schulz, Steffen Hadlak, and Heidrun Schumann. The design space of implicit hierarchy visualization: A survey. *IEEE Transactions on Visualization and Computer Graphics*, 17(4):393–411, 2010.

[162] Edward Segel and Jeffrey Heer. Narrative visualization: Telling stories with data. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1139–1148, 2010.

[163] Vidya Setlur, Sarah E Battersby, Melanie Tory, Rich Gossweiler, and Angel X Chang. Eviza: A natural language interface for visual analysis. In *Proc. of the 29th Annual Symp. on User Interface Software and Technology*, pages 365–377, 2016.

[164] Vidya Setlur, Melanie Tory, and Alex Djalali. Inferencing underspecified natural language utterances in visual analysis. In *Proc. of the 24th Int. Conf. on Intelligent User Interfaces*, pages 40–51, 2019.

[165] Cuili Shao, Yonggang Yang, Sapna Juneja, and Tamizharasi GSeetharam. Iot data visualization for business intelligence in corporate finance. *Information Processing & Management*, 59(1):102736, 2022.

[166] Haifeng Shen, Tomasz Bednarz, Huyen Nguyen, Frank Feng, Theodor Wyeld, Peter J Hoek, and Edward HS Lo. Information visualisation methods and techniques: State-of-the-art and future directions. *Journal of Industrial Information Integration*, 16:100102, 2019.

[167] Leixian Shen, Enya Shen, Yuyu Luo, Xiaocong Yang, Xuming Hu, Xiongshuai Zhang, Zhiwei Tai, and Jianmin Wang. Towards natural language interfaces for data visualization: A survey. *arXiv preprint arXiv:2109.03506*, 2021.

[168] Chuhan Shi, Weiwei Cui, Chengzhong Liu, Chengbo Zheng, Haidong Zhang, Qiong Luo, and Xiaojuan Ma. Nl2color: Refining color palettes for charts with natural language. *IEEE Transactions on Visualization and Computer Graphics*, 2023.

[169] Danqing Shi, Yi Guo, Mingjuan Guo, Yanqiu Wu, Qing Chen, and Nan Cao. Talk2data: High-level question decomposition for data-oriented question and answering. *arXiv preprint arXiv:2107.14420*, 2021.

[170] Ben Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *The Craft of Information Visualization*, pages 364–371. Elsevier, 2003.

[171] Nadia Siddiqui and Enamul Hoque. Convisqa: A natural language interface for visually exploring online conversations. In *2020 24th Int. Conf. Information Visualisation (IV)*, pages 440–447. IEEE, 2020.

[172] Carlos Solis and Xiaofeng Wang. A study of the characteristics of behaviour driven development. In *2011 37th EUROMICRO Conf. on Software Engineering and Advanced Applications*, pages 383–387. IEEE, 2011.

[173] Max Sondag, Wouter Meulemans, Christoph Schulz, Kevin Verbeek, Daniel Weiskopf, and Bettina Speckmann. Uncertainty treemaps. In *2020 IEEE Pacific Visualization Symp. (PacificVis)*, pages 111–120. IEEE, 2020.

[174] Arjun Srinivasan, Bongshin Lee, Nathalie Henry Riche, Steven M Drucker, and Ken Hinckley. Inchorus: Designing consistent multimodal interactions for data visualization on tablet devices. In *Proc. of the 2020 CHI Conf. on Human Factors in Computing Systems*, pages 1–13, 2020.

[175] Arjun Srinivasan, Bongshin Lee, and John Stasko. Interweaving multimodal interaction with flexible unit visualizations for data exploration. *IEEE Transactions on Visualization and Computer Graphics*, 27(8):3519–3533, 2020.

[176] Arjun Srinivasan and Vidya Setlur. Snowy: Recommending utterances for conversational visual analysis. In *The 34th Annual ACM Symp. on User Interface Software and Technology*, pages 864–880, 2021.

[177] Arjun Srinivasan and John Stasko. Natural language interfaces for data analysis with visualization: Considering what has and could be asked. In *Proc. of the Eurographics/IEEE VGTC Conf. on Visualization: Short papers*, pages 55–59, 2017.

[178] Arjun Srinivasan and John Stasko. Orko: Facilitating multimodal interaction for visual exploration and analysis of networks. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):511–521, 2017.

[179] Arjun Srinivasan and John Stasko. How to ask what to say?: Strategies for evaluating natural language interfaces for data visualization. *IEEE Computer Graphics and Applications*, 40(4):96–103, 2020.

[180] John Stasko and Eugene Zhang. Focus+ context display and navigation techniques for enhancing radial, space-filling hierarchy visualizations. In *IEEE Symp. on Information Visualization 2000. INFOVIS 2000. Proc.*, pages 57–65. IEEE, 2000.

[181] Nadine Steinmetz, Bhavya Senthil-Kumar, and Kai-Uwe Sattler. Conversational question answering using a shift of context. In *EDBT/ICDT Workshops*, 2021.

[182] Christina Stoiber, Alexander Rind, Florian Grassinger, Robert Gutounig, Eva Goldgruber, Michael Sedlmair, Štefan Emrich, and Wolfgang Aigner. netflower: Dynamic network visualization for data journalists. In *Computer Graphics Forum*, volume 38, pages 699–711. Wiley Online Library, 2019.

[183] Chris Stolte, Diane Tang, and Pat Hanrahan. Polaris: A system for query, analysis, and visualization of multidimensional relational databases. *IEEE Transactions on Visualization and Computer Graphics*, 8(1):52–65, 2002.

[184] Guodao Sun, Tan Tang, Tai-Quan Peng, Ronghua Liang, and Yingcai Wu. Socialwave: visual analysis of spatio-temporal diffusion of information on social media. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 9(2):1–23, 2017.

[185] Roderick Tabalba, Nurit Kirshenbaum, Jason Leigh, Abari Bhatacharya, Andrew Johnson, Veronica Grosso, Barbara Di Eugenio, and Moira Zellner. Articulate+: An always-listening natural language interface for creating data visualizations. In *Proc. of the 4th Conf. on Conversational User Interfaces*, pages 1–6, 2022.

[186] Mariona Taulé, Alejandro Ariza, Montserrat Nofre, Enrique Amigó, and Paolo Rosso. Overview of detoxis at iberlef 2021: Detection of toxicity in comments in spanish. *Procesamiento del Lenguaje Natural*, 67:209–221, 2021.

[187] Mariona Taulé, Montserrat Nofre, Víctor Bargiela, and Xavier Bonet. Newscom-tox: a corpus of comments on news articles annotated for toxicity in spanish. *Language Resources and Evaluation*, pages 1–41, 2024.

[188] Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

[189] Dolça Tellols, Maite Lopez-Sanchez, Inmaculada Rodríguez, Pablo Almajano, and Anna Puig. Enhancing sentient embodied conversational agents with machine learning. *Pattern Recognition Letters*, 129:317–323, 2020.

[190] NT Thomas. An e-business chatbot using aiml and lsa. In *2016 Int. Conf. on Advances in Computing, Communications and Informatics (ICACCI)*, pages 2740–2742. IEEE, 2016.

[191] Melanie Tory and Vidya Setlur. Do what i mean, not what i say! design considerations for supporting intent and context in analytical conversation. In *2019 IEEE Conf. on Visual Analytics Science and Technology (VAST)*, pages 93–103. IEEE, 2019.

[192] Pere-Pau Vázquez. Are llms ready for visualization? In *2024 IEEE 17th Pacific Visualization Conference (PacificVis)*, pages 343–352. IEEE, 2024.

[193] Tatiana Von Landesberger, Arjan Kuijper, Tobias Schreck, Jörn Kohlhammer, Jarke J van Wijk, J-D Fekete, and Dieter W Fellner. Visual analysis of large graphs: state-of-the-art and future research challenges. In *Computer Graphics Forum*, volume 30, pages 1719–1749. Wiley Online Library, 2011.

[194] Johan Wagemans, James H Elder, Michael Kubovy, Stephen E Palmer, Mary A Peterson, Manish Singh, and Rüdiger von der Heydt. A century of gestalt psychology in visual perception: I. perceptual grouping and figure–ground organization. *Psychological bulletin*, 138(6):1172, 2012.

[195] Chenglong Wang, Yu Feng, Rastislav Bodik, Isil Dillig, Alvin Cheung, and Amy J Ko. Falx: Synthesis-powered visualization authoring. In *Proc. of the 2021 CHI Conf. on Human Factors in Computing Systems*. ACM, may 2021.

[196] Lei Wang, Songheng Zhang, Yun Wang, Ee-Peng Lim, and Yong Wang. Llm4vis: Explainable visualization recommendation using chatgpt. *arXiv preprint arXiv:2310.07652*, 2023.

[197] Qianwen Wang, Zhutian Chen, Yong Wang, and Huamin Qu. Applying machine learning advances to data visualization: A survey on ml4vis. *arXiv preprint arXiv:2012.00467*, 2020.

[198] Qianwen Wang, Zhutian Chen, Yong Wang, and Huamin Qu. A survey on ml4vis: Applying machine learning advances to data visualization. *IEEE Transactions on Visualization and Computer Graphics*, 28(12):5134–5153, 2021.

[199] Ran Wang, Kehan Du, Qianhe Chen, Yifei Zhao, Mojie Tang, Hongxi Tao, Shipan Wang, Yiyao Li, and Yong Wang. Rumorlens: Interactive analysis and validation of suspected rumors on social media. In *CHI Conference on Human Factors in Computing Systems Extended Abstracts*, pages 1–7, 2022.

[200] Yun Wang, Zhitao Hou, Leixian Shen, Tongshuang Wu, Jiaqi Wang, He Huang, Haidong Zhang, and Dongmei Zhang. Towards natural language-based visualization authoring. *IEEE Transactions on Visualization and Computer Graphics*, 29(1):1222–1232, 2022.

[201] Matthew O Ward. Multivariate data glyphs: Principles and practice. pages 179–198, 2008.

[202] Colin Ware. *Information visualization: perception for design*. Morgan Kaufmann, 2019.

[203] Pak Chung Wong and R Daniel Bergeron. 30 years of multidimensional multivariate visualization. *Scientific Visualization*, 2:3–33, 1994.

[204] Linda Woodburn, Yalong Yang, and Kim Marriott. Interactive visualisation of hierarchical quantitative data: an evaluation. In *2019 IEEE Visualization Conference (VIS)*, pages 96–100. IEEE, 2019.

[205] Ziyin Xin, Yujun Cai, Louis T Dang, Hannah MS Burke, Jerico Revote, Natalie Charitakis, Denis Bienroth, Hieu T Nim, Yuan-Fang Li, and Mirana Ramialison. Monago: a novel gene ontology enrichment analysis visualisation system. *BMC Bioinformatics*, 23(1):69, 2022.

[206] Jin Xu, Yubo Tao, Hai Lin, Rongjie Zhu, and Yuyu Yan. Exploring controversy via sentiment divergences of aspects in reviews. In *2017 IEEE Pacific Visualization Symposium (PacificVis)*, pages 240–249. IEEE, 2017.

[207] Jing Yang, Matthew O Ward, and Elke A Rundensteiner. Interactive hierarchical displays: a general framework for visualization and exploration of large multivariate data sets. *Computers & Graphics*, 27(2):265–283, 2003.

[208] Yuyou Yao, Tao Li, Wenming Wu, Gaofeng Zhang, and Liping Zheng. Powerhierarchy: visualization approach of hierarchical data via power diagram. *The Visual Computer*, 40(3):1499–1514, 2024.

[209] Ji Soo Yi, Youn ah Kang, John Stasko, and Julie A Jacko. Toward a deeper understanding of the role of interaction in information visualization. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1224–1231, 2007.

[210] Lu Ying, Tan Tang, Yuzhe Luo, Lvkeshen Shen, Xiao Xie, Lingyun Yu, and Yingcai Wu. Glyphcreator: Towards example-based automatic generation of circular glyphs. *IEEE Transactions on Visualization and Computer Graphics*, 28(1):400–410, 2021.

[211] Lin-Ping Yuan, Ziqi Zhou, Jian Zhao, Yiqiu Guo, Fan Du, and Huamin Qu. Infocolorizer: Interactive recommendation of color palettes for infographics. *IEEE Transactions on Visualization and Computer Graphics*, 28(12):4252–4266, 2021.

[212] Jian Zhao, Nan Cao, Zhen Wen, Yale Song, Yu-Ru Lin, and Christopher Collins. fluxflow: Visual analysis of anomalous information spreading on social media. *IEEE trans. on Visualization and Computer Graphics*, 20(12):1773–1782, 2014.

[213] Boyan Zheng and Filip Sadlo. On the visualization of hierarchical multivariate data. In *2021 IEEE 14th Pacific Visualization Symp. (PacificVis)*, pages 136–145. IEEE, 2021.

[214] Qiyu Zhi and Ronald Metoyer. Gamebot: a visualization-augmented chatbot for sports game. In *Extended Abstracts of the 2020 CHI Conf. on Human Factors in Computing Systems*, pages 1–7, 2020.

[215] Sujia Zhu, Guodao Sun, Qi Jiang, Meng Zha, and Ronghua Liang. A survey on automatic infographics and visualization recommendations. *Visual Informatics*, 4(3):24–40, 2020.

Part V

Appendix

Appendix A

Supplementary Information for Chapter 2

In the following, we summarise the key vocabulary introduced in the DataVis pipeline:

For the Data Space, the characteristics include:

- **Data Transformation**: Operations like filtering, clustering, and aggregation applied to raw data before visualisation.
- **Attributes**: Characteristics of data points, including Nominal (categories), Numerical (quantitative), Temporal (time-based), and Spatial (location-based).
- **Data Type**: **Tabular** (structured, non-connected items) or **Complex** (high-dimensional, temporal, or interconnected data).

In relation to the Visual Space, the characteristics considered include:

- **Visualisation Type (Basic or Advanced)**: Basic layouts (e.g., bar, line charts) vs. advanced layouts (e.g., network graphs, parallel coordinates) for mapping data.
- **Graphical Elements:** Abstract (Lines, Points, Bar) and Symbolic (glyphs, icons) for visualising multivariate data.
- **Visual Mapping Identification (Fixed, User-defined, Rule-based, Intelligent)**: Methods for selecting layouts and graphical elements, ranging from fixed choices to dynamic or AI-driven selections.

- **View Transformation (Single or Multiple)**: Allows users to change the viewpoint (e.g., zooming, panning), measure values, create distortions, and use multiple views or animations to highlight data.
- **Interaction Style (Basic—WIMP or Advanced—NL)**: Basic interaction (e.g., WIMP interfaces) vs. more advanced interaction methods (e.g., Natural Language interfaces).

Lastly, for Interaction Space:

In the Interaction Space, we collect information about the seven interaction methods proposed by Yi et al. [209]: select, explore, reconfigure, encode, abstract/elaborate, filter, and connect. Visualisation tasks (VisTasks) define how users interact with visualisations to derive insights. These tasks help users conduct visual analytics, transforming raw data into meaningful information that aligns with their specific goals. The key Visualisation Tasks are:

- **LookupT**: location and target know
- **LocateT**: location unknown, target known
- **BrowseT**: location known, target unknown
- **ExploreT**: location and target unknown

The **Interaction Style** can be classified based on the type of interaction used. It includes:

- **Basic (WIMP)**: Interaction using traditional input devices like a mouse, keyboard, and pointers.
- **Advanced (NL)**: Interaction using more advanced methods such as Natural Language (NL) interfaces.

Appendix B

Supplementary Information for Chapter 7

```
- intent: define_constructiveness
  examples: |
    - Can you tell me what is meant by [constructiveness]{"entity": "filter_name"}?
    - I was wondering if you could explain [constructiveness]{"entity": "filter_name"} to me?
    - Could you explain to me what [constructiveness]{"entity": "filter_name"} refers to?
    - I would like you to explain what [constructiveness]{"entity": "filter_name"} is
    - Would you be able to give me a tutorial for [constructiveness]{"entity": "filter_name"}?
    - I want you to give me an explanation about [constructiveness]{"entity": "filter_name"}
    - I was hoping you could help me understand [constructiveness]{"entity": "filter_name"}
    - I'd like you to define [constructiveness]{"entity": "filter_name"}
    - Would it be possible to have a tutorial for [constructiveness]{"entity": "filter_name"}?
    - Could you possibly go into more detail about what [constructiveness]{"entity": "filter_name"} is for?
    - Give me help on [constructiveness]{"entity": "filter_name"}
    - Guide for [constructiveness]{"entity": "filter_name"}
    - May you inform me on [constructiveness]{"entity": "filter_name"}
    - Learn more [constructiveness]{"entity": "filter_name"}
```

Figure B.1: Define Constructiveness Intent

```yaml
- intent: help_select_node_edge
  examples: |
    - Can you tell me what is meant by node and edge on left bar?
    - I was wondering if you could explain select node and edge on left menu to me?
    - Could you explain to me what node and edge refers to?
    - I would like you to explain what node and edge is
    - Would you be able to give me a tutorial for node and edge option?
    - I want you to give me an explanation about select node and edge menu
    - I was hoping you could help me understand node and edge feature
    - I'd like you to define node and edge bar
    - Would it be possible to have a tutorial for node and edge?
    - Could you possibly go into more detail about what node and edge is?
    - Give me help on node and edge
    - Guide for node and edge
    - May you inform me on node and edge
    - What is select node and edge
    - What is select node and edge on left bar
    - Tell me about nodes and edges
    - Help select node and edge
```

Figure B.2: Help for Select Node and Edge Intent

```yaml
- intent: highlight_check
  examples: |
    # features--> constructiveness, argumentation

     - I want to see nodes with [constructiveness]{"entity": "filter_name"}
     - Show me comments with [constructiveness]{"entity": "filter_name"}
     - Enable [constructive]{"entity": "filter_name"} nodes filtering
     - Turn on [constructive]{"entity": "filter_name"} nodes

     - I want to see comments with [argument]{"entity": "filter_name"}
     - Show me comments with [argumentation]{"entity": "filter_name"}
     - Check  [argumentation]{"entity": "filter_name"} node and edge
     - Highlight [argumentation]{"entity": "filter_name"} node and edge
```

Figure B.3: Highlight Check intent displaying Constructiveness and Argumentation examples.

User/Bot	Query/Response	Query Group
User	Show me argumentation comments	1
Bot	Here you have it	1
User	Display constructive comments	2
Bot	Here you have it	2
User	I want to see statistics of the features	3
Bot	I've shown you the statistics for the current visualisation in the pop-up window. You can change the variables and graph type using the drop-down menus. If you want more information about other types of statistics…	3

Figure B.4: VisTask1A: Expected conversation.

Table B.1: Expected intents for each task.

VisTask	**Expected Intents**
VisTask1A: In how many comments do we see Argumentation and Constructiveness together?	1) Select comments with Constructiveness 2) Select comments with Argumentation 3) Open statistics for selected comments
VisTask1B: What is the most common level of toxicity for comments tagged with the feature Constructiveness? Are there other features that appear for this level of toxicity?	1) Show comments with Constructiveness 2) Uncheck Constructiveness 3) Choose Not-Toxic comments 4) Show me statistics
VisTask2A: What is the most common level of toxicity for the feature Stereotype?	1) Select Stereotype comments
VisTask2B: What is the most common level of toxicity for the feature Intolerance in the most toxic subtree?	1) Display the most toxic subtree 2) Highlight Intolerance comments

User/Bot	Query/Response	Query Group
User	how many comments include argumentation and constructiveness together?	1
Bot	Which layout do you want to change to?	1
User	select argumentation and constructiveness filters	2
Bot	I'm sorry, I didn't quite understand that. Could you rephrase that please?	2
User	how many comments include argumentation?	3
Bot	I've shown you the statistics for the current visualisation in the pop-up window. You can change the variables and graph type using the drop-down menus. If you want more information about other types of statistics…	3
User	and constructiveness?	4
Bot	How do you want to display it?	4
User	(offered two options user selected highlight constructiveness)	4
Bot	Here you have it. I have automatically selected AND. You can also change to OR.	4
User	how many comments include argumentation?	5
Bot	I've shown you the statistics for the current visualisation in the pop-up window. You can change the variables and graph type using the drop-down menus. If you want more information about other types of statistics…	5
User	and constructiveness?	6
Bot	How do you want to display it?	6
User	(offered two options the user selected to see constructiveness using glyphs)	6
Bot	I've checked the constructiveness colour feature	6
User	Select argumentation node	7
Bot	Here you have it	7
User	and constructiveness filter	8
Bot	I'm sorry, I didn't quite understand that. Could you rephrase that please?	8
User	how do i select filters together?	9
Bot	With OR option, you will see the comments that are tagged with at least one of the features you have selected	9
User	how many comments include arguementation?	10
Bot	Here you have argumentation comments	10
User	how many include constructiveness?	11
Bot	A comment is constructive when it is respectful and polite (regardless of whether it is in favour or against the content of the article or of another comment), when it intends to create an enriching and useful dialogue, when it contributes with new knowledge, ideas and proposals and offers new perspectives and insights to approach the subject.	11
User	how many comments include constructiveness?	12
Bot	I'm sorry, I didn't quite understand that. Could you rephrase that please?	12
User	how many comments include constructiveness?	13
Bot	I'm sorry, I didn't quite understand that. Could you rephrase that please?	13
User	Select constructiveness filter	14
Bot	How do you want to display it?	14
User	(offered two options user selected highlight constructiveness)	14
Bot	Here you have it highlighted	14
User	how many?	15
Bot	I can help you with…	15
User	can i see it in a graph?	16
Bot	I can help you learn about:	16
	User selects to learn about statistics graphs	16
	Selected Comments Statistics shows you overall numbers of features for comments you've selected using the Select Node and Edge menu.	16

Figure B.5: VisTask1A: Example of a long conversation of User 12

User/Bot	Query/Response	Query Group
User	filter by argumentation	1
Bot	How do you want to display it?	1
User	(offered two options to the user selected to see it with glyphs)	1
Bot	I''ve checked the argumentation colour feature	1
User	plot selected comments	2
Bot	I've shown you the statistics for the current visualisation in the pop-up window. You can change the variables and graph type using the drop-down menus. If you want more information about other types of statistics…	2
User	count argumentation	3
Bot	Where do you want to uncheck it from?	3
User	filter argumentation	4
Bot	How do you want to display it?	4
User	(offered two options the user selected to see constructiveness by highlighting it.)	4
Bot	Here you have it	4

Figure B.6: VisTask1A: Example of a minimal conversation of User 11