UNIVERSITAT DE BARCELONA

Facultat de Matemàtiques i Informàtica

UNIVERSITAT DE BARCELONA

FUNDAMENTAL PRINCIPLES OF DATA SCIENCE MASTER'S THESIS

Explaining Word Interactions Using Integrated Directional Gradients

Author:
Marc BALLESTERO RIBÓ

Supervisors:
Dr. Daniel ORTIZ-MARTÍNEZ
Prof. Dr. Petia RADEVA

A thesis submitted in partial fulfillment of the requirements for the degree of MSc in Fundamental Principles of Data Science

in the

Facultat de Matemàtiques i Informàtica

June 17, 2025

UNIVERSITAT DE BARCELONA

Abstract

Facultat de Matemàtiques i Informàtica

MSc

Explaining Word Interactions Using Integrated Directional Gradients

by Marc BALLESTERO RIBÓ

Explainability methods are key for understanding the decision-making processes behind complex text models. In this thesis, we theoretically and empirically explore Integrated Directional Gradients (IDG), a method that can attribute importance to both individual features and their high-order interactions for deep neural network (DNN) models. We introduce evaluation metrics to quantitatively assess the quality of the generated explanations, and propose a framework to adapt word-level evaluation methods to high-order phrase-level interactions. Applying IDG to a BERT-based hate speech detection model, we compare its performance at the word level against well-established methods such as Integrated Gradients (IG) and Shapley Additive Explanations (SHAP). Our results indicate that, while IDG's word-level attributions are less faithful than those of IG and SHAP, they are the best-scoring ones in terms of plausibility. On the other hand, IDG's high-order importance attributions exhibit high faithfulness metrics, indicating that IDG can consider hierarchical dependencies that traditional methods overlook. Qualitative analyses further support the interpretability of IDG explanations. Overall, this thesis highlights the potential of high-order explanation methods for improving transparency in text models.

Acknowledgements

Marc Aureli (26 d'abril del 121, Roma – 17 de març del 180, Vindobona, l'actual Viena) comença les seves *Meditacions* amb una evocació dels aprenentatges rebuts de les seves figures familiars, seguint una fórmula anafòrica concisa però efectiva: "Del meu avi Ver, la bondat i el caràcter tranquil. De la reputació i el record del meu progenitor, la discreció i la fermesa [...]" [1]. Aquí, em proposo adaptar l'anàfora aureliana per expressar el meu agraïment.

D'una banda, aquest treball marca la fi d'una etapa formativa a Barcelona; de l'altra, representa l'inici d'un nou capítol vital i acadèmic a l'estranger. És per això que voldria fer extensiu aquest agraïment no només a aquells que m'han acompanyat en l'elaboració d'aquest treball concret, sinó també a totes les persones que, al llarg dels darrers anys, han estat un suport inestimable i constant en el meu camí de creixement acadèmic i personal.

***

A la Petia i a en Dani, pel suport, els bons consells i l'escolta activa constants durant el temps que he estat treballant en aquest projecte.

A la mare i el pare, per ser-hi sempre per recordar-me que la vida s'ha d'afrontar amb aplom, els peus a terra i la il·lusió als ulls.

A la Clara, per les hores diàries que ens passem rient —que mai són massa—, i per la virtut de sempre saber-me arrencar una riallada, tant en els moments plàcids com en els més durs.

Als amics de sempre —l'Ona, en Jan, en Pol, en Pau, la Mireia, en Jordi i en Biel— per recordar-me que per passar una bona tarda només cal una taula i una baralla de cartes.

Als amics de la Universitat —en Neil, la Meritxell, l'Alba i l'Arnau, entre d'altres—, per les hores compartides de converses acadèmiques i extraacadèmiques; per ser-hi i haver-hi estat.

A la meva estimada cosineta, l'Ares, per les hores i hores de consells, guiatge i suport a l'hora de sol·licitar una posició doctoral; per fer-me veure que, encara que jo no m'ho cregués, hi havia possibilitats.

A la resta de la meva família —de sang i política—, per ser el refugi de pau i sobrietat que sovint necessitem els que ens dediquem a l'acadèmia.

A Can Vilaró (Carrer del Comte Borrell, 61, Barcelona), per servir-me religiosament cada divendres al migdia un plat de cap-i-pota amb tripa al forn acompanyat d'una copa de vi de Gandesa —l'al·licient perfecte per acudir a classe un divendres a la tarda.

A la Mineta, la meva gata, per fer-me companyia —sempre des de la inherent distància emocional del felí— durant les hores que he passat estudiant tancat a l'habitació.

Contents

Chapter 1

Introduction

One of the reasons behind the unprecedented predictive power of deep neural networks (DNNs) is their ability to capture high-level interactions between features [2, 3]. In the field of linguistics, the *principle of compositionality* states that the meaning of a complex expression is determined by the meanings of its constituents and the rules used to combine them. This principle, albeit highly debated, remains as a foundational concept in the study of natural language [4, 5]. Consequently, to understand how semantic and syntactic structures contribute to a model's predictions, it is essential for explainability methods in natural language processing (NLP) to quantify the importance of feature interactions. For instance, in a sentiment classification task, the phrase "not bad" carries an entirely different sentiment than what one would infer by looking at the words "not" and "bad" separately. Capturing the interactions encoded in such combinations is essential for robust and interpretable explanations.

Traditional feature attribution methods, such as Local Interpretable Model-Agnostic Explanations (LIME) [6], Shapley Additive Explanations (SHAP) [7, 8] or Integrated Gradients (IG) [9], explain a model's prediction by assigning an importance value to each input feature depending on how much it has influenced the output. Although these approaches are helpful to understand which features are important, they fail to account for high-order dependencies and interactions.

There exist several approaches to explain the importance of feature interactions for a predictor which have been applied to NLP models:

- Murdoch *et al.* [10] propose Contextual Decomposition (CD), a method to explain the predictions of long short-term memory units (LSTM). Nonetheless, its application to state-of-the-art models, such as the ones based on self-attention layers, is not straightforward.
- Singh *et al.* [11] introduced a variation of CD, namely Agglomerative Contextual Decomposition (ACD), that can hierarchically quantify interaction importance for feed-forward and convolutional neural networks (CNN). However, as CD, it cannot be straightforwardly applied to more recent and powerful models.
- Lundberg *et al.* [8] introduced SHAP interaction values as a natural high-order extension of SHAP values, being inspired on the game-theoretical concept of the Shapley interaction index, first formalized by Grabisch and Roubens [12]. These are originally designed to quantify the contribution of the interaction between pairs of features to a model's prediction, and their extension to arbitrary groups of features is straightforward. Nevertheless, as with standard SHAP values, the exact computation of SHAP interaction values is NP-hard [13], and even approximate computations are typically

very resource-intensive. This computational limitation turns them infeasible for certain applications.

- Chen *et al.* [14] propose HEDGE, a method based on the SHAP interaction value to generate hierarchical explanations specifically for text classification models. Despite its interpretability, the method emerges from a non-axiomatic formulation, and thus fails to fulfil certain assumptions that are considered essential for feature importance attribution methods.
- Janizek *et al.* [15] introduce Integrated Hessians (IH), a second-order extension of Integrated Gradients (IG) designed to quantify pairwise feature interactions. While this method has been tested in the NLP context, from this author's perspective, its utility is limited by the fact that it can only explain interactions between pairs of features. In language models, predictions often rely on more complex, higher-order dependencies across multiple tokens, such as syntactic structures or discursive patterns. Therefore, limiting the analysis to pairwise interactions may fail to account for the full extent of the model's reasoning.

In this thesis, we explore and adopt Integrated Directional Gradients (IDG) [16] as the method for interaction-based explainability. IDG addresses many of the inherent limitations of the previously discussed approaches. In particular, it enables the attribution of importance to arbitrary feature groups, rather than being restricted to individual or pair-wise interactions. Moreover, it is computationally efficient when working with differentiable prediction functions, and thus well-suited to be applied to DNN based models.

Hate speech represents a challenging and troubling threat on online social media platforms, often leading to real-life consequences such as increased crimes against minority groups [17]. Thus, there is a growing interest in developing automated models capable of detecting such kinds of discourses. Numerous research efforts have addressed this problem by proposing various combinations of datasets and modeling approaches. Nevertheless, persistent issues remain, such as biased predictions against particular protected groups and lack of explainability [18–25].

To provide researchers with a standardized dataset to evaluate hate speech detection models, Mathew *et al.* [26] introduced the HateXplain dataset. This dataset covers multiple dimensions of hate speech, with data collected from different social media platforms. Most relevantly, each text sample is paired with a set of human-provided rationales, *i.e.* annotations which indicate the snippets of text that support the categorization of a sequence as being constitutive of hate speech. This feature is crucial because it not only enables the development of more sophisticated models, but also allows for the quantitative evaluation of explainability techniques by measuring how well the generated importance scores agree with the human annotated rationales.

The primary goals of this thesis are to provide a thorough theoretical description of IDG and to apply and evaluate it on a hate speech detection model. We explore and implement appropriate validation metrics to quantitatively assess the performance of IDG in explaining model predictions on a specific dataset. In particular, we develop a modified version of explanation quality metrics adapted for the evaluation of higher-order attributions. Subsequently, we develop a hate speech detection model trained on the HateXplain dataset, and then adapt and apply IDG to generate interaction-based explanations for its predictions.

Using this approach, we investigate how effectively IDG can capture complex feature interactions in the context of hate speech classification and evaluate the interpretability and reliability of these explanations.

Report Structure

This thesis report is structured as follows:

- **Chapter 1** introduces the context and motivation for the thesis work.
- **Chapter 2** develops the theoretical framework of IDG and discusses its specific application to text classification models.
- **Chapter 3** defines the metrics used to quantitatively evaluate the quality of the explanations generated by IDG, and introduces an adaptation of existing metrics to assess higher-order importance attributions.
- **Chapter 4** reports the methodology and empirical findings derived from the application of IDG to a hate speech detection model on the HateXplain dataset.
- **Chapter 5** summarizes the conclusions and outlines the future research lines that arise from this thesis.

Additionally, appendices are included to provide supplementary information, such as the computational environment used to run the project code, the public availability of the codebase and additional figures supporting the main text.

Notational Conventions

Throughout this thesis, we assume the existence of a feature space $\mathcal{X} \subseteq \mathbb{R}^N$ and a target space for a classification problem with M labels, denoted by $\mathcal{Y} \subseteq [0,1]^M$. Given a deep neural network (DNN), we denote by $f : \mathcal{X} \to \mathcal{Y}$ the predictor function that, for a given input vector, returns a probability distribution over the target classes. Unless otherwise specified, for any $x \in \mathcal{X}$, we denote by $[f(x)]_j$ the probability that the model assigns to instance x belonging to class j.

Disclaimer Regarding Harmful or Offensive Content

This thesis involves the use of the HateXplain dataset, which contains real-world examples of hate speech and offensive language. Such instances cannot be avoided due to the nature of the work. As a result, some examples included in this report contain explicit, offensive, or harmful content. These excerpts are presented for the unique purpose of scientific analysis and do not reflect the views of the author or affiliated institutions. Reader discretion is advised.

Chapter 2

Integrated Directional Gradients

This chapter is devoted to providing an exhaustive theoretical description of *integrated directional gradients*, a method for computing hierarchical feature group attributions, and discussing the particularities of its application to text sequence classification models.

2.1 The Feature Group Attribution Problem

2.1.1 Problem Formulation

Integrated Directional Gradients (IDG) [16] is a method designed to attribute importance scores to groups of features, computing their relevance to the output of a DNN given an input. Formally, the method aims at solving the *feature group attribution problem.*

Definition 2.1. The *feature group attribution problem* is defined as follows:

> Given an input $x = (x_1, \ldots, x_n) \in \mathcal{X}$, a baseline $b \in \mathcal{X}$, a predictor (DNN function) $f : \mathcal{X} \to \mathcal{Y}$ and a family of meaningful subsets $M \subseteq \mathcal{P}(\{x_1, \ldots, x_n\})$, assign to every meaningful subset of features $S \in M$ an *importance score* $v(S)$.

The main challenge for solving the problem is, then, the design of an interpretable and well-behaved *importance function* $v : M \to \mathbb{R}$. This function should attribute a real value to each meaningful subset of features in M that is a direct proxy of its importance for the predictor with respect to a baseline input b, which represents the absence of contribution from any feature.

This problem formulation goes along the line of previous research on computing feature importance values using game theory-inspired approaches, the most relevant of which being Shapley Additive Explanations (SHAP) [7, 8]. These approaches consider features to be players of a cooperative game, which collectively collaborate to reach the output of the DNN. Individual feature importances are then calculated using Shapley values, assuming the existence of a *value function* that represents the alteration in model output when some features values are known by the model and others are considered unknown and equal to some baseline value.

The IDG approach to computing feature importances is fundamentally different from previous game theory-derived schemes, but its axiomatic formulation takes inspiration from them, while its implementation framework stems from Integrated Gradients (IG) [9].

The *family of meaningful subsets* M captures the notion that, in some applications, not all combinations of features represent meaningful parts of the input. In other words, in some cases, to effectively interact, features need to form structured groups that respect inherent structural dependencies or coherence. Text analysis is one of such cases: according to compositionality, when constructing explanations for NLP models, it is important to restrict them to subsets of features that agree with the syntactic and semantic structure of language [4, 5].

2.1.2 Axiomatic Requirements

This section presents the properties that the solution $v : M \to \mathbb{R}$ to the feature group attribution problem should desirably fulfil. Axiomatic formulations are mainly based on [16]. Before that, an auxiliary definition is needed.

Definition 2.2. Given two features, with indices i and j, we can define the two following notions of equivalence:

- **Functional Equivalence.** The i-th and j-th features are *functionally equivalent* if, for every pair of input vectors x, y such that $x_i = y_j$, $x_j = y_i$, and $x_k = y_k$, $\forall k \notin \{i, j\}$, then $f(x) = f(y)$.
- **Structural Equivalence.** The i-th and j-th features are *structurally equivalent* regarding a family of meaningful feature subsets M if, for all $S \in M$ and an arbitrary input vector x, $x_i \in S$ and $S \neq \{x_i\}$ implies that $x_j \in S$ and vice versa.

This definition is a variation of the notion of feature equivalence necessary for the axiomatic formalization of IG [9]. The concepts presented here extend the ones reported in [9] introducing the necessary details to solve the feature group attribution problem.

Two features are said to be *functionally equivalent* if swapping the values of those features in the input does not alter the output of the predictor. On the other hand, two features are said to be *structurally equivalent* if they occupy equivalent positions in the structure derived from the family of meaningful feature subsets M.

With this auxiliary definition, the desired properties of the solution to the feature group attribution can be postulated.

Definition 2.3. Given a solution to the feature group attribution problem $v : M \to \mathbb{R}$, we define the following properties.

(1) **Non-negativity**: The importance of any relevant feature subset should be non-negative.

$$\forall S \in M,\ v(S) \geq 0 \tag{2.1}$$

(2) **Normality**: The value of the empty set of features is zero.

$$v(\varnothing) = 0 \tag{2.2}$$

(3) **Monotonicity**: The importance of a set of features is greater than or equal to the importance of its subsets.

$$\forall S, T \in M,\ S \subseteq T \implies v(S) \leq v(T) \tag{2.3}$$

(4) **Superadditivity**: The value of the disjoint union of two feature subsets is greater than or equal to the sum of values of the two subsets.

$$\forall S, T \in M,\ S \cap T = \varnothing,\ v(S \sqcup T) \geq v(S) + v(T) \tag{2.4}$$

(5) **Sensitivity (a)**: Let there be a feature, with index i, such that $f(x) \neq f(b)$ for every input x and baseline b that only differ in the i-th feature. Then, $v(\{x_i\}) > 0$ and $v(S) > 0$ for every set of features S containing the value of the i-th feature.

(6) **Sensitivity (b)**: Let there be a feature, with index j, such that $f(x) = f(b)$ for every input x and baseline b that only differ in the j-th feature. Then, $v(\{x_j\}) = 0$ and $v(S) = v(S \setminus \{x_j\})$ for every set of features S containing the value of the j-th feature.

(7) **Symmetry Preservation**: If two features, with indices i and j, are both functionally and structurally equivalent, and given an input x and baseline b such that $x_i = x_j$ and $b_i = b_j$, then, for every subset of features $S \subseteq \{x_1, \ldots, x_n\} \setminus \{x_i, x_j\}$, we have that $v(S \cup \{x_i\}) = v(S \cup \{x_j\})$.

(8) **Implementation Invariance**: Given two neural networks, DNN_1 and DNN_2, let us denote their corresponding predictors f_1 and f_2. Let us also denote the solutions for the feature group attribution problem applied to each network by $v_1, v_2 : M \to \mathbb{R}$, respectively. Assume that the two neural networks are *functionally equivalent*, *i.e.* $f_1(x) = f_2(x)$, $\forall x \in \mathcal{X}$. Then, the solutions are *implementation invariant* if, for any input x, we have that $v_1(S) = v_2(S)$ for all $S \in M$.

The *non-negativity* axiom comes from the fact that the importance of a feature subset should be a directionless quantity. Other implementations of feature importance attribution explainers do not enforce this property and hence can attribute importances with negative values. In classification tasks, such negative values typically indicate a tendency toward predicting alternative classes rather than the one being explained or inferred [6–9]. The view behind IDG is that feature importance should be a proxy of the amount of information (statistical entropy) contained in the feature values that is effectively used by the model, and thus a non-negative figure.

Properties **2** to **4** emanate from some of the foundational axioms of cooperative game theory [27, 28]. *Normality* ensures that the importance of the empty set of features (that is, no information at all provided to the model) should be null. *Monotonicity* captures the fact that adding more features to a subset adds more information, therefore attributed importance should not decrease. Finally, *superadditivity* represents the fact that, in the IDG framework, feature collaboration is assumed to never be detrimental. Hence, collaboration among features should not lead to a lower combined importance attribution.

Properties **5** to **8** are variations of the axioms for IG presented in [9]. *Sensitivity (a)* makes sure that a feature that does effectively affect the output of a predictor is attributed a non-zero importance. On the other hand, *sensitivity (b)* assures that any feature being irrelevant for the predictor is attributed a null importance, and that does not alter the importance of any feature group containing it. *Symmetry preservation* ensures that features that are both functionally and structurally equivalent, apport the same amount of importance to all feature subsets there may be included in. Finally, the *implementation invariance* property is just a guarantee that different implementations of a same predictor lead to equal importance attribution functions.

The following result states that non-negativity together with superadditivity automatically implies monotonicity. This will become important later in the theoretical development.

Lemma 2.4. A non-negative and superadditive solution of the feature group attribution problem $v : M \to \mathbb{R}$ is monotonic.

PROOF: Let $S, T \in M$, $S \subseteq T$. Then, we can construct T as a disjoint union

$$T = S \sqcup (T \setminus S), \quad S \cap (T \setminus S) = \varnothing. \tag{2.5}$$

By superadditivity, we have that

$$v(T) = v\left(S \sqcup (T \setminus S)\right) \geq v(S) + v(T \setminus S), \tag{2.6}$$

and, since non-negativity implies that $v(T \setminus S) \geq 0$, we conclude that

$$v(T) \geq v(S), \tag{2.7}$$

and hence v is monotonic. □

2.1.3 The IDG Solution

The authors of [16] present IDG as a solution for the feature group attribution problem, mainly inspired by IG [9] and Harsanyi dividends [29]. Harsanyi dividends are a concept in cooperative game theory that assigns a value to the marginal contribution of players, reflecting the value uniquely attributable to their cooperation [29, 30].

The main idea behind IDG is to construct a value function in terms of the dividends generated by each meaningful feature subset. In this framework, any meaningful feature group $S \in M$ contributes additional value to the model. This additional value is represented in the dividend of the group, which is denoted by $\mathrm{d}(S)$ and satisfies $0 \leq \mathrm{d}(S) < 1$.

One of the first and simplest measures of the importance of a feature for a DNN model is the partial derivative of the prediction function with respect to the input feature. This approach is sometimes referred to in the literature as *Vanilla Gradients* [31–33]. Gradients are a natural analog of the coefficients of a linear model for a DNN, therefore seem a good starting point for feature importance attribution. Nonetheless, this simple formulation breaks the sensitivity properties, which are claimed to be essential for any attribution method [9]. The authors of Integrated Gradients solve the issue by taking a path integral of the partial derivative of the predictor with respect to the feature through the straight line path in the feature space $\mathcal{X}$ connecting the input x to the baseline b. To compute the dividend of a single feature, IDG takes an equivalent approach and computes the absolute value of the path integral of the partial derivative connecting x and b.

The dividend of a group of features should measure the interaction between the features present in the group. IDG considers the directional derivative of the predictor in the direction of the given set of features as a representative surrogate of the importance of the feature interactions, and the dividend is then computed as the normalized absolute value of the path integral of this directional derivative over the straight path connecting x and b. The sign of the path integral shall be further interpreted as the nature of the contribution, positive or negative, to the output.

Let us mathematically formalize this high-level description of the IDG solution. Let $x = (x_1, \ldots, x_n)$ be an input vector, $b = (b_1, \ldots, b_n)$ the baseline, f the predictor, and $S \in M$ a family of meaningful feature subsets. Let $c = \arg\max_k [f(x)]_k$ and $f_c(x) := [f(x)]_c$, hence $f_c(x)$ denotes the probability assigned by f to the predicted class for x.

The direction of the straight line path in the feature space connecting x and b, given the features in S, is given by the vector

$$z^S = (z_1^S, \dots z_n^S), \text{ with } z_i^S = \begin{cases} x_i - b_i, & \text{if } x_i \in S \\ 0, & \text{otherwise} \end{cases}, \quad i \in \{1, \dots n\}. \tag{2.8}$$

In other words, z^S is a vector that has non-zero components only in the directions of the features in the group S, representing the change from b to x along the subset of features S.

The directional derivative of the predictor in the direction given by z^S can then be computed as

$$\nabla_S f_c(x) = \nabla f_c(x)^T \hat{z}^S, \text{ where } \hat{z}^S = \frac{z^S}{\|z^S\|}. \tag{2.9}$$

We now can define the integrated directional gradient, the dividend and the importance value of the feature group S.

Definition 2.5. With the notation given in equations (2.8) and (2.9), we define the following:

- The *integrated directional gradient* of the feature group S is defined as

$$\mathrm{IDG}\,(S) = \int_0^1 \nabla_S f_c\,(b + \alpha\,(x - b))\,\mathrm{d}\alpha. \tag{2.10}$$

  That is, the path integral of the directional derivative of the predictor between the baseline and the input in the direction given by the feature group S.

- The *dividend* of S is given by

$$\mathrm{d}(S) = \begin{cases} \dfrac{|\mathrm{IDG}\,(S)|}{\sum_{T \in M} |\mathrm{IDG}\,(T)|} & \text{if } S \in M \\ 0 & \text{otherwise} \end{cases} \tag{2.11}$$

  i.e., the normalized absolute value of IDG (S) over all the meaningful feature subsets in M, such that the sum of dividends of all meaningful feature groups adds up to 1.

- The *importance value* of S is calculated as

$$v(S) = \sum_{\substack{T \subseteq S \\ T \in M}} \mathrm{d}(T). \tag{2.12}$$

  In other words, the value of a feature subset is the result of adding up the dividends of all meaningful subsets contained in it, including itself.

The most important aspect of this solution is that it fulfils the eight desired properties stated in definition 2.1.

Theorem 2.6. The IDG importance attribution function v, as defined in definition 2.5, fulfils the properties of definition 2.3.

PROOF: We prove each property in turn.

(1) **Non-negativity.** Since the dividend of any meaningful feature group S is defined as a non-negative value (either a normalized sum of absolute values or zero) and $v(S)$ is defined as a sum of dividends, $v(S) \geq 0$ for any S by construction.

(2) **Normality.** We have that

$$v(\varnothing) = \sum_{\substack{T \subseteq \varnothing \\ T \in M}} \mathrm{d}(T) = \mathrm{d}(\varnothing). \tag{2.13}$$

Note that, following equation (2.8),

$$z^{\varnothing} = (0, \dots, 0) \implies \nabla_{\varnothing} f_c(x) = \nabla f_c(x)^T \hat{z}^{\varnothing} = 0, \tag{2.14}$$

which leads to

$$\mathrm{IDG}\,(\varnothing) = \int_0^1 0 \, \mathrm{d}\alpha = 0, \tag{2.15}$$

which finally yields $\mathrm{d}(\varnothing) = 0$.

Before proving monotonicity, we will prove superadditivity, and then use lemma 2.4.

(4) **Superadditivity.** Given $S, T \in M$, $S \cap T = \varnothing$, we have that

$$v(S \sqcup T) = \sum_{\substack{R \subseteq S \sqcup T \\ R \in M}} \mathrm{d}(R) = \underbrace{\sum_{\substack{R \subseteq S \\ R \in M}} \mathrm{d}(R)}_{v(S)} + \underbrace{\sum_{\substack{R \subseteq T \\ R \in M}} \mathrm{d}(R)}_{v(T)} + \underbrace{\sum_{\substack{R \subseteq S \sqcup T \\ R \cap S \neq \varnothing,\, R \cap T \neq \varnothing \\ R \in M}} \mathrm{d}(R)}_{\geq 0}, \tag{2.16}$$

since all meaningful subsets of $S \sqcup T$ can be decomposed into those that are subsets of S, those that are subsets of T and those that intersect both S and T. Using the definition of the dividend (2.11) and non-negativity, this yields that

$$v(S \sqcup T) \geq v(S) + v(T), \tag{2.17}$$

as desired.

(3) **Monotonicity.** We have already proved non-negativity and superadditivity. Applying lemma 2.4, we get that v is also monotonic.

(5) **Sensitivity (a).** Suppose that the i-th feature is such that $f(x) \neq f(b)$ for an input x and baseline b that only differ in the i-th component. It is sufficient to see that $\mathrm{IDG}\,(\{x_i\}) \neq 0$. Using equation (2.8), we have that

$$\hat{z}^{\{x_i\}} = (0, \dots, 0, \overset{i}{1}, 0, \dots, 0). \tag{2.18}$$

In this case, x_i is the only feature that varies on the path connecting x and b, hence we can write f_c as a single-variable function: $f_c(x) = g(x_i)$. Then,

$$\nabla_{\{x_i\}} f_c(x) = \frac{\mathrm{d}}{\mathrm{d}x_i} g(x_i), \tag{2.19}$$

from where it follows that

$$\mathrm{IDG}\,(\{x_i\}) = \int_0^1 \frac{\mathrm{d}}{\mathrm{d}x_i} g\left(b_i + \alpha(x_i - b_i)\right) \mathrm{d}\alpha = \frac{1}{x_i - b_i} \int_{b_i}^{x_i} \frac{\mathrm{d}}{\mathrm{d}x_i} g(x_i) \, \mathrm{d}x_i =$$

$$= \frac{g(x_i) - g(b_i)}{x_i - b_i} = \frac{f(x) - f(b)}{x_i - b_i} \neq 0. \tag{2.20}$$

(6) **Sensitivity (b).** Suppose that the j-th feature is such that $f(x) = f(y)$ for every pair of inputs x and y that only differ in the j-th component. It is sufficient to see that, for

all $S \in M$ such that $x_i \in S$, then $\text{IDG}(S) = \text{IDG}(S \setminus \{x_i\})$. By hypothesis, since f is constant in the j-th component,

$$\frac{\partial}{\partial x_i} f_c(x) = 0, \tag{2.21}$$

which implies that

$$\nabla_S f_c(x) = \nabla_{S \setminus \{x_i\}} f_c(x), \tag{2.22}$$

that finally leads to $\text{IDG}(S) = \text{IDG}(S \setminus \{x_i\})$. Moreover, equation (2.21) also implies that $\nabla_{\{x_i\}} f(x) = 0$, thus $v(\{x_i\}) = 0$.

(7) **Symmetry Preservation.** Assume that two features, with indices i and j are both functionally and structurally equivalent. Let x and b be, respectively, an input and a baseline such that $x_i = x_j$ and $b_i = b_j$. Let S be a feature subset such that $S \subseteq \{x_1, \dots, x_n\} \setminus \{x_i, x_j\}$. Functional equivalence implies that

$$\frac{\partial}{\partial x_i} f(x) = \frac{\partial}{\partial x_j} f(x). \tag{2.23}$$

Moreover, considering the equality condition between the i-th and j-th components of x and b, it follows that

$$\nabla_{S \cup \{x_i\}} f_c(x) = \nabla_{S \cup \{x_j\}} f_c(x), \tag{2.24}$$

which means that $x_i = x_j$ on every point of the path connecting x and b, from where it follows that $\text{IDG}(S \cup \{x_i\}) = \text{IDG}(S \cup \{x_j\})$.

(8) **Implementation Invariance.** The importance function v is defined in such a way that depends entirely on the functional form of the gradients of the predictor and its evaluations, and not on the particular DNN architecture that is being considered. Hence, v is implementation invariant.

□

2.2 IDG Applied to Text Classification

This section is devoted to describing the application of IDG to explaining an embedding-based deep model for text classification. The section delves into three challenges: (1) choosing a baseline feature vector b, (2) determining the structure of meaningful feature subsets M, and (3) visualizing and interpreting the output of IDG.

2.2.1 Choosing a Baseline b

The baseline input b should ideally represent the absence of any meaningful contribution from the input features. In other words, it should correspond to an input such that the model receives no relevant information for making a prediction. Most importantly, the baseline choice is known to have an impact on the performance and reliability of the computed attributions [34].

Aghababaei *et al.* [34] and Bastings *et al.* [35] propose the following options for the baseline choice:

- **Zero Baseline:** the baseline embedding is filled with zero values.
- **Mask Baseline:** the baseline embedding is filled with `[MASK]` token embeddings.
- **Padding Baseline:** the baseline embedding is filled with `[PAD]` token embeddings.

- **Unknown Baseline:** the baseline embedding is filled with `[UNK]` token embeddings.
- **Mean Baseline:** the baseline embedding is filled with the mean value calculated on the whole set of the model input embeddings.

One of the most commonly used approaches is the padding baseline [14–16, 36], where the input is replaced with the `[PAD]` token to simulate the absence of information. However, recent work by Enguehard *et al.* [37] argues in favor of using a mask baseline instead.

Although this may seem counterintuitive (the `[PAD]` token is typically assumed to represent a complete lack of content, while `[MASK]` is a trained token) the authors point out a key aspect: the `[PAD]` token is untrained and therefore may be arbitrarily close in the embedding space to any particular word. In contrast, the `[MASK]` token is trained to simulate random absence of information, making it a more semantically consistent choice as a baseline input for attribution methods.

Despite these arguments, comprehensive comparative studies such as Aghababaei *et al.* [34] have found that a zero baseline is the most consistent and reliable option. Similarly, the proponents of IDG also use the zero baseline in their approach. Accordingly, we adopt the zero baseline for the attribution analyses conducted in this thesis.

It is worth noting that, in the case of BERT, the zero vector embedding corresponds to the encoding of the `[PAD]` token. As a result, using a zero baseline is effectively equivalent to employing a padding token baseline. A similar situation occurs with XLNet, where the zero vector corresponds to the embedding of the `[UNK]` token, making these two approaches interchangeable in practice for that model as well.

2.2.2 The Family of Meaningful Feature Subsets M

As defined in §2.1.1, the *family of meaningful subsets* M should account for the fact that not all combinations of features represent meaningful parts of the input. In the case of text analysis, it is important to restrict explanations to subsets of features that agree with the syntactic and semantic structure of language [4, 5].

Common linguistic formalisms, such as Chomskyan Principles and Parameters (P&P) [38] and Head-Driven Phrase Structure Grammar (HDPS) [39] are based on the rule of compositionality [5], which materializes in the idea that the semantics of a sentence can be read off its constituency parse tree [40].

Syntactic constituency is the idea that groups of words can behave as single units or *constituents* [41]. Constituency parse trees are hierarchical structures that represent the syntactic structure of a sentence according to phrase structure rules. Each node in the tree corresponds to a syntactic constituent and reflects a semantically meaningful span of text. Figure 2.1 shows an instance of such a tree.

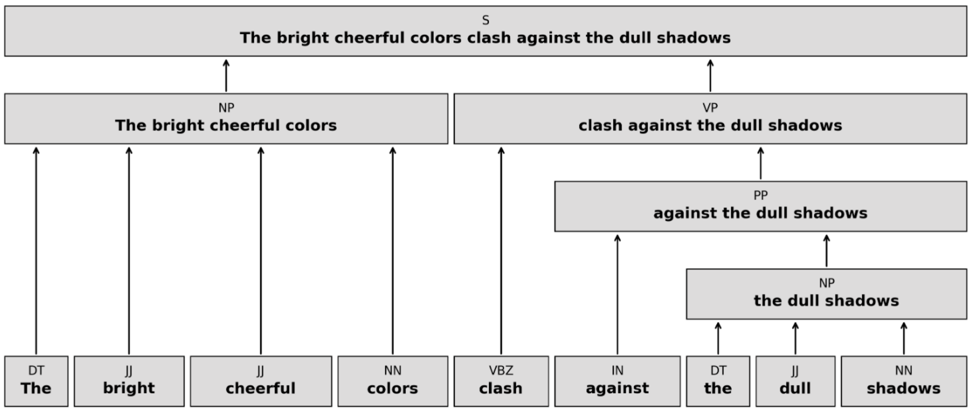

FIGURE 2.1: Example of the constituency tree for the sequence "The bright cheerful colors clash against the dull shadows", generated using the `Stanza` constituency parser [42]. Part-of-speech (POS) and syntactic tags for the different constituents are indicated above the words. Abbreviations are according to the Penn treebank [43] (`DT` $\equiv$ determiner, `JJ` $\equiv$ adjective, `NN` $\equiv$ noun, `VBZ` $\equiv$ verb in the 3rd person of singular present, `IN` $\equiv$ preposition, `NP` $\equiv$ noun phrase, `PP` $\equiv$ prepositional phrase, `VP` $\equiv$ verb phrase, `S` $\equiv$ simple declarative clause).

In our framework, the family of meaningful subsets M will be therefore defined as the collection of all text spans corresponding to internal nodes in the constituency parse tree of a given sentence. This will enable the attribution method to focus on linguistically meaningful substructures, thus ensuring that the resulting explanations are interpretable from a syntactic and semantic perspective.

2.2.3 Output Data: Visualization and Interpretation

Recall that the output of IDG is precisely a map $v : M \to \mathbb{R}$ that assigns an absolute importance score to each meaningful subset of features belonging to the family M. In our framework, this corresponds to assigning an importance score to each part of the constituency parse tree of a sentence.

Generally, constituency parse trees are generated as unbalanced trees, meaning that branches can have arbitrary depths. To facilitate the visualization and interpretation of the explanations, we have developed an algorithm to pad the tree and convert it into a depth-balanced structure. This transformation allows the tree to be viewed as a structured aggregation process, where a text sequence is systematically constructed from its constituents, as shown in figure 2.2.

Algorithm 1 describes the pseudocode implementation[1] of a recursive procedure to depth-balance a tree. We use the following notation: given a tree `T` and a node `node`, `Children(node)` denotes the set of children of `node`.

[1]Some standard Python functions and methods, such as `zip` or `append` have been used to simplify notation.

Algorithm 1: Equalize Depths of a Tree.

```
Input: A tree T with its root node root.
Output: A depth-balanced tree T' and its height h(T')
Function EqualizeDepths(node):
    if Children(node) = ∅ then
        return node, 1;
    new_children ← [];
    heights ← [];
    foreach child ∈ Children(node) do
        new_child, h ← EqualizeDepths(child);
        new_children.append(new_child);
        heights.append(h);
    max_height ← max(heights);
    padded_children ← [];
    foreach child, h ∈ zip(new_children, heights) do
        padded_child ← child;
        for i ← 0 to max_height - h - 1 do
            padded_child ← new node with padded_child as child;
        padded_children.append(padded_child);
        Children(node) ← padded_children;
    return node, max_height + 1;
To balance the tree: Call EqualizeDepths(root) for the root of the tree T.
```

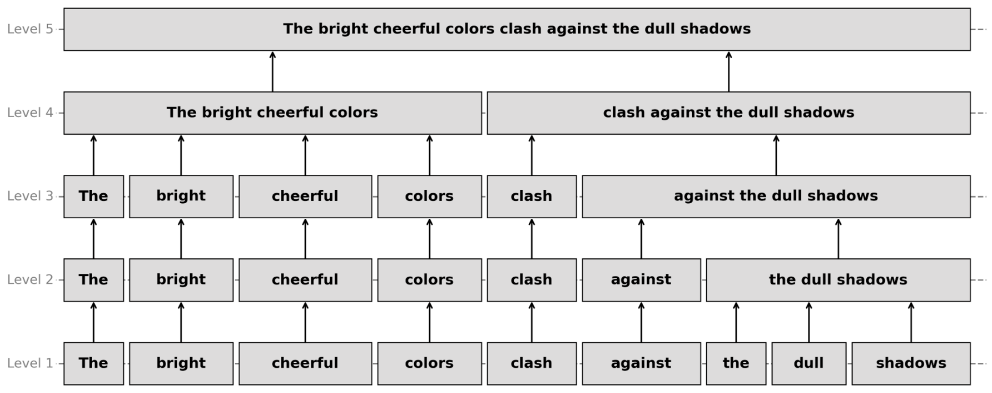

FIGURE 2.2: Depth-balanced constituency tree for the sequence displayed in figure 2.1, obtained applying algorithm 1.

The visualization of scores in the depth-balanced constituency parse tree provides an intuitive understanding of how different parts of a text contribute to a model's prediction. Each node in the tree is assigned a normalized importance score, where the root (which represents the entire sentence) always receives a score of 1, indicating that it contains 100% of the information necessary for the prediction.

To further enhance interpretability, for a given set of features $S \in M$ within the tree, each score $v(S)$ is represented multiplied by the *dividend direction* of S, defined as the sign of IDG (S). This accounts for the fact that nodes can have positive or negative contributions. Specifically, a positive score indicates that the corresponding text segment supports the predicted class, while a negative score suggests that it opposes the prediction. The magnitude of the score captures the strength of this influence.

By visualizing these signed scores on the padded, depth-balanced tree, the hierarchical structure of the sentence and the relative impact of each constituent become easily apparent. Branches that accumulate higher scores (positive or negative) can be visually identified as critical for the model's decision-making process. This balanced representation ensures that all branches are displayed at a uniform depth, making it easier to compare the contributions of different segments and understand how the model processes the text.

Figure 2.3 shows an example of the visualization of the IDG output as described above. A color map is used to represent the value of the product between the dividend direction and the attribution score for each text span.

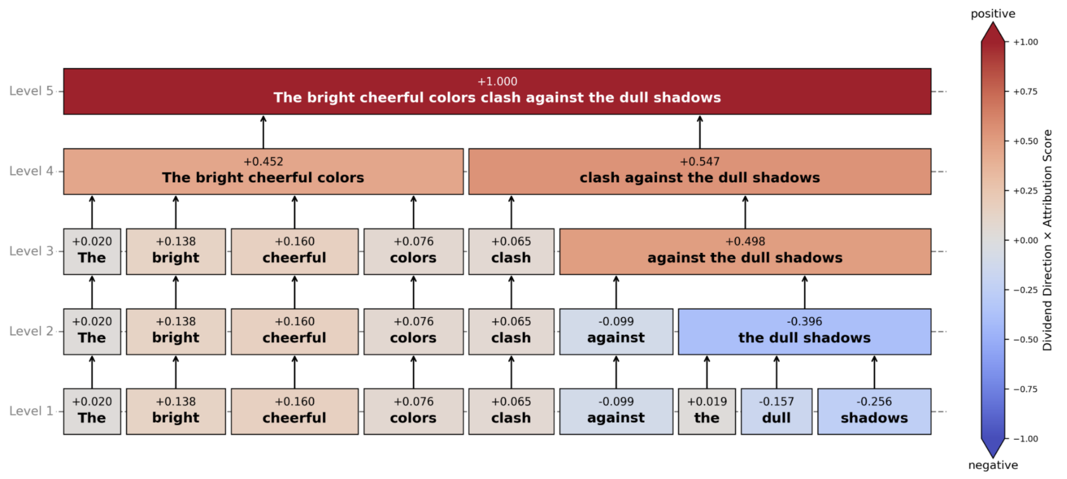

FIGURE 2.3: Results given by IDG on a sentiment classification task. Note how the different text constituents interact, with "bright cheerful colors" suggesting positivity while "dull shadows" introduces negative connotations, illustrating how the model balances these interactions to reach its final decision.

Chapter 3

Validation Metrics

This chapter defines the metrics used to evaluate the quality of the explanations generated by IDG, and introduces extensions of existing faithfulness metrics to assess importance attributions. These metrics are further adapted for global evaluation of high order feature importance attributions through a bias-corrected aggregation process.

3.1 Word-level Attributions

This section is centered on the evaluation of the feature attributions generated by an importance attribution method applied to a text classification model at the word-level.

We focus the evaluation on two different aspects of interpretability, following the line of DeYoung *et al.* [44] and Jacobi and Goldberg [45]:

- **Agreement with Human Rationales** or **Plausibility**. Assuming a rationalized dataset, *i.e.*, a dataset where the importance of each word in a text sequence has been manually annotated by a human, we measure how well the importance scores produced by the explainability method align with these human-provided scores.
- **Agreement with internal Model Behavior** or **Faithfulness**. Here, we want to evaluate how well the importance scores provided by the explainability method reflect the internal behavior of the model. In other words, we want to capture to which extent tokens marked as important actually influenced the model output.

3.1.1 Agreement with Human Rationales: Plausibility

To compute plausibility metrics, we assume the availability of a rationalized dataset. That is, given a text sequence $s = (w_1, \dots, w_n)$, where $w_i,\ i \in \{1, \dots, n\}$ are words, there exists a vector $r = (r_1, \dots, r_n)$ that encodes feature importance scores according to human annotators.

Often, human-annotated rationales are sparse and consist of hard importance scores. For instance, a rationale may take the form of a binary vector, where each component is either 1 or 0, indicating whether a word is considered important or not by the annotator. When multiple annotators have scored the same sequence, *soft rationales* can be computed by averaging their individual annotations. This results in a continuous-valued importance score that captures the degree of agreement among annotators.

We will define two distinct metrics to evaluate plausibility: *agreement@k* and the *area under the precision-recall curve* (AUPRC).

3.1.1.1 Agreement@k

There are studies that evaluate feature importance attribution as a ranking task [35, 46], restricting the evaluation to the top k ranking features. However, there is no consensus about how k should be selected. In fact, selecting a fixed value of k is that it can result in excluding tokens with scores close to the top-k and including tokens with low scores, not accounting for high importance gaps [47].

Kamp *et al.* [47] address the issue by introducing agreement@k as a metric to concurrently evaluate plausibility across multiple values of k. Before defining agreement@k, we need to introduce relevance@k as an auxiliary concept.

Definition 3.1. Let $s = (w_1, \ldots, w_n)$ be a sequence of words, $r = (r_1, \ldots, r_n)$ be a vector of soft-scored human rationales for s, and $\phi = (\phi_1, \ldots, \phi_n)$ be a vector of feature importance attributions generated by some explainability method. We shall assume that both r and ϕ are ℓ^1-normalized[1]. Let $\text{top}_k(v)$ denote the set of the k highest scoring component indices in a vector $v = (v_1, \ldots, v_n)$. We define the *relevance@k* of a word feature w_i as

$$\text{relevance}@k(w_i) = \frac{1}{2}\left[\mathbb{1}_{\{i \in \text{top}_k(\phi)\}} + \mathbb{1}_{\{i \in \text{top}_k(r)\}}\right]. \tag{3.1}$$

In other words, relevance@$k(w_i)$ is equal to the ratio of methods (explainability method or human scoring) that include w_i in the top k scoring features.

Definition 3.2. Let $s = (w_1, \ldots, w_n)$ be a sequence of words, $r = (r_1, \ldots, r_n)$ be a vector of soft-scored human rationales for s, and $\phi = (\phi_1, \ldots, \phi_n)$ be a vector of feature importance attributions generated by some explainability method, both ℓ^1-normalized. We define the *agreement@k* of the sequence s relative to the attributions of ϕ and r as

$$\text{agreement}@k(s) = \frac{\sum_{i=1}^{n} \text{relevance}@k(w_i)}{\sum_{i=1}^{n} \mathbb{1}_{\{\text{relevance}@k(w_i)>0\}}}. \tag{3.2}$$

By including the term in the denominator, we ensure that features not included neither in $\text{top}_k(r)$ nor $\text{top}_k(\phi)$ are not considered in the calculation of agreement@$k(s)$ and thus the metric is not artificially inflated by having high agreement on irrelevant tokens.

Definition 3.3. Let $\mathcal{X} = \{s_1, \ldots, s_m\}$ be a dataset of sentences. The agreement@k of the whole dataset $\mathcal{X}$ is calculated by averaging the sentence-level scores:

$$\text{agreement}@k(\mathcal{X}) = \frac{1}{m}\sum_{i=1}^{m} \text{agreement}@k(s_i). \tag{3.3}$$

This metric allows for the quantification of the agreement between an explainability method and human annotators at a whole range of values of k. It is worth noting that it is expected for the agreement score to increase the closer k gets to the length of the word sequence. This is referred to as *sentence length bias* [47]. Thus, it is particularly interesting and insightful to analyze the behavior of this metric at low to mid-range values of k.

[1]A vector $v = (v_1, \ldots, v_n)$ is ℓ^1-normalized if $\sum_{i=1}^{n} |v_i| = 1$.

3.1.1.2 Area under the Precision-Recall Curve

While agreement @k captures how well an explainability method ranks tokens in alignment with human rationales, it emphasizes ordinal agreement rather than raw identification ability. In other words, it does not directly assess whether the method is effective at identifying any token that humans consider important, regardless of its exact position in a ranked list.

To address this, we complement the agreement @k evaluation with a precision-recall analysis over token importance, following the approach proposed by DeYoung *et al.* [44].

Specifically, we treat human rationales binarized ground-truth relevance labels[2]. The attribution scores produced by the explainability method are then treated as predictions. This allows us to compute the Area Under the Precision-Recall Curve (AUPRC), by sweeping a threshold over token scores. This provides a threshold-independent measure of the method's ability to identify relevant tokens according to human annotations.

A high AUPRC is an indication that the method assigns high importance scores to tokens that humans also consider relevant, even if it does not rank them precisely in the same order. Thus, this metric captures a complementary notion of explainability quality: the mere ability to detect importance rather than ranking it correctly.

An AUPRC globalized across an entire dataset can be obtained by computing the mean of the AUPRC for each of the instances belonging to the dataset.

3.1.2 Agreement with Model Behavior: Faithfulness

The computation of faithfulness metrics does not rely on the availability of a rationalized dataset: importance scores are evaluated regarding their ability to reflect the true reasoning process of the model. Measuring faithfulness beyond plausibility is important because an explanation method might provide explanations that are agreeable to humans (plausible) but are not functionally relevant for the model.

We adopt the notions of *comprehensiveness* and *sufficiency* as defined by DeYoung et al. [44], which evaluate the faithfulness of explanations through input perturbation. Specifically, comprehensiveness measures the drop in model confidence when the most important features according to the explanation method are removed, while sufficiency measures how much of the original confidence is conserved when only the most important features are kept.

Nonetheless, these definitions rely on a fixed feature importance threshold to compute these metrics. From this perspective, such a fixed-threshold approach may limit the ability to assess how faithfulness varies across different levels of feature importance.

Variants that calculate a globalized metric have been proposed, such as the Area Over the Perturbation Curve (AOPC). This metric, inspired by previous work in the computer vision domain by Samek *et al.* [48], measures the average change in model output as top-ranked input features are progressively removed according to their importance. It typically evaluates the effect of perturbation over a predefined set of bins for top-ranked features (e.g., top 1%, 5%, 10%, etc.), thus offering a more comprehensive view of explanation faithfulness than single-threshold methods.

[2]In cases where the original rationales are soft scores, such as when they result from averaging multiple annotators' decisions, they are converted into binary labels through thresholding, effectively binarizing the rationales for evaluation purposes.

While AOPC provides a useful discretized approximation of attribution impact, it remains limited by the choice and granularity of bins. To address this limitation, we introduce two threshold-independent faithfulness metrics which integrate the effects of removing or retaining top-ranked features across all possible feature importance thresholds: area under the comprehensiveness perturbation curve (AUCPC) and area over the sufficiency perturbation curve (AOSPC).

We believe that, by treating comprehensiveness and sufficiency as continuous functions over feature importance thresholds, these area-based metrics can provide a finer-grained view of faithfulness, enabling a richer comparison between explanation methods.

3.1.2.1 Comprehensiveness

Comprehensiveness measures to which extent were all the features needed to make a prediction marked as important by the explainability method. Let us define it formally.

Definition 3.4. Let $s = (w_1, \dots, w_n)$ be a sequence of words, $\phi = (\phi_1, \dots, \phi_n)$ a ℓ^1-normalized vector of feature importance attributions, and $f : \mathcal{X} \to \mathcal{Y}$ the model predictor. Let $t \in [0, 1]$ be a threshold value. We shall also assume a baseline input vector $b = (b_1, \dots, b_n)$ in the sense of definition 2.1 and §2.2.1. We define a *contrast example* as $\tilde{s}^t = (\tilde{s}^t_1, \dots, \tilde{s}^t_n)$ from s given ϕ and t, with components given by

$$\tilde{s}^t_i = \begin{cases} b_i, & \text{if } \phi_i \geq t \\ s_i, & \text{otherwise} \end{cases} \qquad i \in \{1, \dots n\}. \tag{3.4}$$

Suppose the prediction of the sequence s favors class c. We define the *comprehensiveness score* of ϕ for the sequence s and the threshold t as

$$\text{comprehensiveness}_t(s) = [f(s)]_c - \left[f(\tilde{s}^t)\right]_c, \tag{3.5}$$

where $[f(s)]_c$ indicates the output probability assigned by the predictor for the sequence s belonging to class c.

To compute comprehensiveness, given a sequence and its explanation, we construct a counterexample where the tokens marked as relevant by the explanation (according to some threshold) are removed and substituted by a baseline value. We then measure the drop in the model's confidence for the originally predicted class, computed as the difference between the model's probability assigned to the predicted class on the original input and that assigned to the same class on the counterexample input.

A large drop (*i.e.*, a high comprehensiveness score) indicates that the removed tokens were indeed influential in the prediction, and thus that the feature importance attributions were faithful to the model's behavior. A negative comprehensiveness score means that the model became more confident in its prediction after the relevant tokens were removed, something completely counter-intuitive if the tokens were indeed relevant for the prediction.

Previous approaches to comprehensiveness define a fixed importance threshold for its computation. Here, to avoid relying on a specific threshold $t \in [0, 1]$ for selecting relevant tokens, we construct a *comprehensiveness-threshold curve* by sweeping t across its domain. Since the x component of the curve corresponds to t and the y component to the comprehensiveness score (*i.e.*, the change in predicted probability), all points of the curve are contained in the $[0, 1] \times [-1, 1]$ rectangle of the plane.

As a threshold-independent comprehensiveness measure, we then compute the signed area under the comprehensiveness-threshold curve, which we will call the *area under the comprehensiveness perturbation curve* (AUCPC). Note that AUCPC is bounded between -1 and 1, with higher values indicating greater overall comprehensiveness across all thresholds.

For a dataset consisting of several instances of text sequences, the global AUCPC can be computed by computing the area under the average comprehensiveness-threshold curve across all instances.

3.1.2.2 Sufficiency

Sufficiency measures to which extent the features marked as important by the explainability method contain enough signal for the model to reach its prediction. It can be understood as the complementary metric to comprehensiveness. Let us define it formally.

Definition 3.5. Let $s = (w_1, \dots, w_n)$ be a sequence of words, $\phi = (\phi_1, \dots, \phi_n)$ a ℓ^1-normalized vector of feature importance attributions, and $f : \mathcal{X} \to \mathcal{Y}$ the model predictor. Let $t \in [0, 1]$ be a threshold value. We shall also assume a baseline input vector $b = (b_1, \dots, b_n)$ in the sense of definition 2.1 and §2.2.1. We define a *reduced example* as $\bar{s}^t = (\bar{s}^t_1, \dots, \bar{s}^t_n)$ from s given ϕ and t, with components given by

$$\bar{s}^t_i = \begin{cases} s_i, & \text{if } \phi_i \geq t \\ b_i, & \text{otherwise} \end{cases} \qquad i \in \{1, \dots n\}. \tag{3.6}$$

Suppose the prediction of the sequence s favors class c. We define the *sufficiency score* of ϕ for the sequence s and the threshold t as

$$\text{sufficiency}_t(s) = [f(s)]_c - \left[f(\bar{s}^t)\right]_c, \tag{3.7}$$

where $[f(s)]_c$ indicates the output probability assigned by the predictor for the sequence s belonging to class c.

To compute sufficiency, given a sequence and its explanation, we construct a reduced example where the tokens not marked as relevant by the explanation (according to some threshold) are removed and substituted by a baseline value. We then measure the drop in the model's confidence for the originally predicted class, computed as the difference between the model's probability assigned to the predicted class on the original input and that assigned to the same class on the reduced input.

A small drop (*i.e.*, a low sufficiency score) indicates that the tokens identified as important preserve most of the information needed by the model to reach its prediction, suggesting that the explanation captures a minimal, but sufficient subset of important features. On the other hand, a large probability drop indicates that the supposedly irrelevant tokens were actually contributing to the model's decision, hence undermining the faithfulness of the explanation.

As before, to avoid relying on a specific threshold for selecting relevant tokens, we construct a *sufficiency-threshold curve* by sweeping t across its domain. As with the comprehensiveness -threshold curve, all are contained in the $[0, 1] \times [-1, 1]$ rectangle of the plane.

Then, as a global, threshold-independent sufficiency measure, we compute the signed area under the sufficiency-threshold curve. Note that lower values of this area correspond to globally lower sufficiency scores, and thus more faithful explanations. To construct a metric such that higher values reflect greater faithfulness (consistent with the interpretation of the previously defined AUCPC), we define the *area over the sufficiency perturbation curve* (AOSPC) as one minus the signed area under the sufficiency-threshold curve.

With this definition, AOSPC is bounded between -1 and 1, with higher values indicating greater overall sufficiency across all thresholds.

For a dataset consisting of several instances of text sequences, the global AOSPC can be computed by computing the area over the average sufficiency-threshold curve across all instances.

3.2 High-order Interactions

We evaluate the importance attributions for higher order interactions from the faithfulness perspective, as defined in §3.1, by introducing a multi-level approach for the computation of AUCPC and AOSPC.

Recall that a solution to the feature group attribution problem (particularly IDG) assigns to each meaningful subset of features an importance score, reflecting their contribution to the underlying model prediction. In our framework, that means assigning an importance value to each node within the constituency parse tree of a sentence.

As mentioned in §2.2.3, constituency parse trees are, in general, unbalanced trees. To facilitate high-order evaluation, we will use the procedure described in algorithm 1 to transform the attribution trees to depth-balanced structures.

Once the tree is balanced, each level corresponds to a sequence of importance attributions over the text spans represented by that level's nodes. At this point, comprehensiveness and sufficiency can be evaluated for the attributions at each level. Following the methodology described in §3.1.2, AUCPC and AOSPC can be computed using the respective threshold curves.

Global high-order AUCPC and AOSPC can be computed by averaging the level-wise threshold curves that are first aggregated over the dataset at each tree level. However, this naïve averaging approach fails to account for two critical sources of bias:

- **Support bias:** The maximum height of a constituency parse tree is inherently tied to the length of the sentence it represents. Consequently, at any given level, the number of instances that contribute to the faithfulness evaluation varies depending on the distribution of sentence lengths in the dataset. Levels closer to the root may be underrepresented due to shorter sequences that do not reach such heights.
- **Span coverage bias:** Constituency tree nodes at higher levels correspond to longer text spans. Feature attribution methods, particularly IDG, tend to artificially assign higher importance scores to these spans, as they inherently contain more information and thus more potential influence on the model's output. This introduces an undesired inflation in attribution values.

To address these biases, the averaging of perturbation curves must be performed in a weighted manner. Specifically, we introduce correction factors that account for uneven support across levels and normalize for the inflated importance of longer spans. These adjustments allow for the definition of global faithfulness metrics that better reflect the actual quality of the attributions, regardless of structural imbalances in the dataset.

Let $t \in [0, 1]$ be a threshold value and $f_l(t)$ be a faithfulness metric (either comprehensiveness or sufficiency) evaluated at level l and threshold t (thus, $f_l(t)$ stands for either the comprehensiveness-threshold or sufficiency-threshold curve at level l). Let L be the maximum level reached by the sequences in the dataset. We then compute the corrected global average curve as

$$F(t) = \frac{1}{\Omega} \sum_{l=1}^{L} \omega_l f_l(t), \tag{3.8}$$

where ω_l is a correction factor accounting for the support and span coverage imbalances in the dataset and $\Omega = \sum_{l=1}^{L} \omega_l$.

In practice, datasets generally contain a larger proportion of shorter sequences, which results in lower-level tree nodes being more frequently represented than those lying at higher levels. At the same time, nodes at higher levels in constituency trees correspond to longer text spans, which are more likely to receive inflated attribution scores because they artificially contain more information. Therefore, a correction factor that simultaneously adjusts for these two imbalances can be defined as

$$\omega_l := \frac{\text{support at level } l}{\text{mean maximum span coverage at level } l}, \quad l \in \{1, \ldots, L\}. \tag{3.9}$$

The corrected curve $F(t)$ can finally be interpreted as a global faithfulness curve that accurately reflects the true quality of the high-order importance attributions, independently of structural biases specific to the dataset. Using this curve, global AUCPC and AOSPC metrics for the high-order importance attributions can then be computed, which we will denote AUCPC_{HO} and AOSPC_{HO}.

3.3 Global Qualitative Insights

Beyond quantifying the method's agreement with human annotators or model functional dependencies, the generated explanations can also provide global qualitative insights. For instance, Saeteros et al. [49] used word clouds [50] to visualize the tokens that were most globally relevant for the model when classifying instances into specific classes.

Here, we adopt this approach by generating word clouds based on the mean importance scores assigned to individual words, in order to represent their global relevance. To capture the global importance of low-level interactions, we apply the same method to 2-grams (word pairs).

Chapter 4

IDG Applied to Hate Speech Detection

This chapter is devoted to explaining the procedure and results derived from the application of IDG to a hate speech detection model trained on the HateXplain dataset.

4.1 Materials and Methods

4.1.1 Dataset

Our practical work focuses on the HateXplain dataset, a benchmark dataset for explainable hate speech detection developed by Mathew *et al.* [26]. HateXplain is the first benchmark dataset for the detection of hate speech with word annotations that capture the human rationales for labelling.

4.1.1.1 Dataset Overview

The data compiled in HateXplain was sourced from two social media platforms:

- **Twitter**[1]. The authors filtered posts from a 1% random sample of tweets in the period from January 2019 to June 2020.
- **Gab**[2]. The authors use a previously curated dataset [52], which contains filtered hateful posts.

Following the approach of previous studies on hate speech on social media platforms [18, 52–54], the authors behind HateXplain build a corpus of posts using lexicons of unigrams and bigrams that are usually associated with hate.

Reposts and duplicates were removed, and it was ensured that posts did not contain links or audiovisual materials, since they could carry additional information not available to annotators. Emojis were not excluded, as they might be relevant for the labelling task. All posts were anonymized by replacing usernames with the `<user>` token. Further text normalization was applied to replace certain entities with abstract placeholders:

- `<number>`: numeric values.
- `<percent>`: percentages.

[1]Former name for the social network 𝕏. Link: `https://x.com`.

[2]Gab is a social media platform known for its far-right user base, described as a haven for neo-Nazis, white supremacists, white nationalists, antisemites, the alt-right, supporters of Donald Trump, conservatives, right-libertarians, and believers in conspiracy theories [51]. Link: `https://gab.com`.

- `<money>`: monetary values.
- `<date>`, `<time>`: temporal expressions.
- `<email>`, `<phone>`: personal data.
- `<laugh>`, `<sad>`, `<happy>`, `<surprise>`, `<kiss>`, `<wink>`, `<annoyed>`, etc.: non-textual emotive and expressive content (glyphs or non-alphanumeric character combinations).

Human annotators performed two distinct annotation tasks on the dataset:

- **Target class annotation:** annotators were asked to determine whether a post was either hateful, offensive or neither of the two, *i.e.*, normal. Each post was independently assessed by three annotators, and the ground truth class was determined by majority voting. The cases where the three annotators disagreed on the assessment were not considered for the study.
- **Rationale annotation**: for each post labeled as hateful or offensive, annotators were asked to justify their decision by highlighting the specific tokens that contributed to their judgment.

4.1.1.2 Data Selection and Preprocessing

The latest version of the whole, unprocessed, HateXplain dataset was downloaded from its GitHub repository[3]. All selection and preprocessing procedures were carried out using the Polars Python package, an open-source library for fast and efficient data manipulation [55].

First, data instances in which the three annotators disagreed on the target class assessment were excluded, only keeping posts for which more than two (out of three) annotators agreed. The target class of such instances was determined by majority voting.

Subsequently, to focus on a binary classification task, only instances labeled as either *hateful* or *normal* were preserved, while those labeled as *offensive* were discarded.

In the original dataset, each rationale was encoded as a boolean vector, where each element indicates whether a token was part of the rationale. The final (global) rationale for a post was obtained by averaging the boolean vectors provided by different annotators.

To ensure further compatibility with tokenizer models, special Unicode characters, such as `u200d`, `u200f` or accented characters, were replaced with special placeholders. Moreover, all text excerpts were systematically lowercased to ensure consistency.

Finally, the dataset was split into train, validation, and test segments for the subsequent analyses, and saved in `parquet` format for further use. The splitting was done in a stratified manner, ensuring similar class distributions in each split.

After filtering and preprocessing, the dataset comprised a total of $N = 13749$ samples. Table 4.1 shows the number of instances for each split of the dataset and each target class, as well as the counts for the total dataset and the ratio of normal to hate speech samples. Note that this ratio is preserved across all splits, confirming that the stratified sampling procedure was effective.

[3]GitHub repository for the HateXplain dataset.

TABLE 4.1: Instance counts for each split of the dataset and target class, as well as the ratio of normal to hate speech samples.

	Class			
Split	Normal	Hate Speech	**Total**	Ratio (N:HS)
Train	6251	4748	10 999	1.32
Validation	781	593	1374	1.32
Test	782	594	1376	1.32
Total	7814	5935	13 749	1.32

Table 4.2 presents the number of samples per platform for each split, as well as for the entire dataset. It also includes the ratio of Twitter to Gab samples. A small residual number of samples could not be attributed to either Twitter or Gab due to the lack of metadata in the original dataset. Note that the Twitter-to-Gab ratio is preserved across all splits except for the validation set, where it is slightly higher.

TABLE 4.2: Instance counts for each split of the dataset and post platform, as well as the ratio of Twitter to Gab samples.

	Platform			
Split	Twitter	Gab	Unknown	Ratio (T:Gab)
Train	5149	5831	19	0.88
Validation	663	709	2	0.94
Test	643	731	2	0.88
Total	6455	7271	23	0.89

Figure 4.1 shows the distribution of the number of tokens per sample across dataset splits and target classes. As can be seen, there are no notable differences in the distributions for each class. Additional figures regarding further descriptive statistics of the dataset can be consulted in §B.1. Overall, no significant differences in the studied parameters are appreciated.

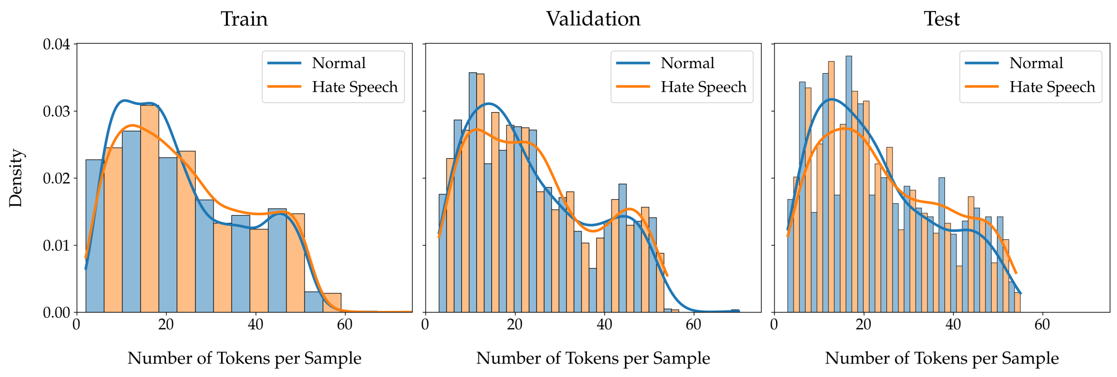

FIGURE 4.1: Distribution of the number of tokens per sample for each dataset split and target class. A kernel density estimation (KDE) curve is overlaid facilitate interpretation.

4.1.2 Model Selection

To develop a hate speech detection algorithm, two transformer-based state-of-the-art deep learning text models were implemented and evaluated: Bidirectional Encoder Representations from Transformers (BERT) [56] and XLNet [57]. These models were selected due to their strong performance on a wide range of natural language understanding tasks and their ability to capture contextual information in text through self-attention mechanisms.

BERT is a bidirectional transformer model pretrained using masked language modeling and next sentence prediction tasks, allowing it to learn deep bidirectional representations for text by jointly considering both left and right contexts. In contrast, XLNet uses autoregressive language modeling, incorporating permutation-based factorization to capture bidirectional dependencies without relying on masking, which can lead to improved performance in certain downstream tasks.

Both models have been implemented and fine-tuned using the Hugging Face Transformers library [58] and Pytorch [59], which jointly offer access to a wide range of pretrained models as well as a unified interface for managing and fine-tuning transformer-based architectures. For details regarding the computational environment and hardware specifications used to run the code, please refer to Appendix A.

The fine-tuning for both models was carried out using equivalent training strategies. First, task-specific pre-trained models and tokenizers were downloaded from the Hugging Face repositories. In the case of BERT, the `BertForSequenceClassification` implementation was employed, with weights initialized from the pre-trained `bert-base-uncased` model. For XLNet, the `XLNetForSequenceClassification` implementation was used, with weights initialized from the `xlnet-base-cased` pre-trained model.

Input text sequences were tokenized and encoded before being fed into the models for fine-tuning. Specifically, sequences were tokenized using the respective model tokenizers, converting text into input IDs and attention masks for each transformer architecture. Padding and truncation were applied to ensure consistent input lengths within each batch. In particular, a maximum sequence length of 128 elements was selected for both models. Additionally, placeholders for special elements described in section §4.1.1.1 and emojis were manually added to the tokenizers as special tokens to ensure these elements were correctly recognized and processed by the models.

The models were fine-tuned in a supervised way using the labeled data from the training split. For BERT, fine-tuning was performed with a batch-size of 32, a learning rate of 2×10^{-5} and a maximum of 5 training epochs. The AdamW optimizer was used, with a weight decay of 0.1 and a linear learning rate scheduler with 100 warmup steps. Early stopping was applied by monitoring the validation loss, with training halted if no improvement was observed. The XLNet model was fine-tuned under the same settings.

The final model for inference was selected based on its performance on the validation dataset, considering multiple metrics such as the confusion matrix, accuracy, F_1-score, and area under the receiver operating characteristic curve (AUROC).

4.1.3 Explainability

4.1.3.1 Generation of Explanations

To interpret the model's predictions at the interaction level, we employed IDG. All the explainability analysis was done using the samples of the test split of the dataset.

The official implementation of IDG, publicly available on GitHub[4], is built on PyTorch and the Captum library [60]. However, the codebase depended on outdated versions of these libraries and, in particular, was found to be incompatible with the latest release of the Transformers library. We therefore conducted a thorough revision and refactoring of the code to ensure compatibility with the latest dependencies and to adapt the implementation to our specific requirements.

In line with the discussion in §2.2.1, we adopted a zero baseline approach for the attribution computation. For BERT, this is equivalent to defining the baseline as a vector completely constituted of `[PAD]` tokens.

As detailed in §2.2.2, we selected constituency parse trees as the family of meaningful subsets for defining feature importance. These trees were generated systematically using the Stanza constituency parser [42]. We used an standardly configured pipeline, which relies on the Penn Tree Bank (PTB) [43], updated following the guidelines of Mott *et al.* [61].

Additionally, we generated word-level feature importance attributions using IG and SHAP as reference baselines. These methods, widely adopted for feature importance estimation, served as a benchmark for assessing the quality of the first-order explanations produced by IDG.

We employed the Transformers-Interpret library implementation of IG [62]. For SHAP, we used the official library released by Lundberg et al. [7], and specifically applied the Partition Explainer algorithm to generate explanations.

4.1.3.2 Quantitative Evaluation

The quality of the generated explanations was assessed using the metrics described in §3.

Word-level Attributions

The attributions at the word level generated by IDG were quantitatively evaluated from both the plausibility and faithfulness perspective, and compared with the values corresponding to the explanations generated by IG and SHAP.

Regarding plausibility, agreement@k and the area under the precision-recall curve (AUPRC) were computed. Since both metrics require samples with human-provided rationales, these evaluations were limited to instances from the hate speech class, as they are the only ones annotated with importance labels in our dataset. Only correctly predicted samples were considered.

For consecutive values of k between 1 and 15, agreement@k was computed for each of the considered samples. The global agreement@k for all the samples was computed using the averaging formula described in §3.1.1.1.

To compute the AUPRC, each soft-scored human rationale was binarized into a ground-truth vector by marking a token as important if it had a positive soft score (*i.e.*, if at least one of the three annotators considered it relevant). Then, the precision-recall curve comparing the explanation method's attributions to the human annotations was computed for each sample. A global AUPRC was obtained by averaging the individual sample AUPRCs.

[4]GitHub repository for the IDG implementation.

Globally averaged precision-recall curves were also computed by interpolating all individual precision-recall curves over a fixed range of recall values and then averaging the corresponding precision values.

As for faithfulness, we evaluated the explanations using the comprehensiveness and sufficiency metrics. Specifically, each feature importance attribution vector was ℓ^1-normalized before computing the comprehensiveness-threshold and sufficiency-threshold curves, as defined in §3.1.2. The padding baseline approach was applied by masking the appropriate tokens in each perturbed example with `[PAD]` tokens. Consistent with the plausibility evaluation, only correctly predicted samples were considered. Global AUCPC and AOSPC values were obtained by averaging the individual sample AUCPCs and AOSPCs, respectively.

High-order Interactions

The importance of high-order feature interactions given by IDG was solely evaluated from the faithfulness perspective. A plausibility analysis would have required rationalized constituency parse trees, which were not available.

First, the constituency parse tree of each sequence was depth-balanced using algorithm 1. The importance attribution vector associated to each level of the depth-balanced tree was ℓ^1-normalized. Then, comprehensiveness-threshold and sufficiency-threshold curves for each level within each sample were computed.

Next, to obtain a global representation, level-wise curves were averaged across all samples in the dataset, resulting in a global curve per tree depth level. Finally, as discussed in §3.2, a corrected averaging procedure was applied to these level-wise curves to compute global faithfulness metrics that account for support and span-coverage biases.

For each tree level, we computed the support (i.e., the number of samples whose depth-balanced constituency trees reached that level) and the mean maximum span coverage, defined as the average maximum fraction of the sentence length covered by nodes at that level. This analysis revealed that support is approximately inversely proportional to the tree level, while the mean maximum span coverage tends to increase roughly in direct proportion to the level. Hence, we introduced correction factors to compensate for these dependencies, following the guidelines described in §3.2 and equation (3.9). Finally, using the globally corrected curves, we computed AUCPC_{HO} and AOSPC_{HO}.

4.1.3.3 Qualitative Evaluation

To visualize the explainability trees generated by IDG, a custom visualization framework was developed using Matplotlib [63], following the guidelines described in §2.2.3. This allowed us to intuitively inspect the hierarchical structure of importance attributions and assess the coherence and interpretability of the explanations.

Global qualitative explanations, in the sense of §3.3, were generated by creating word clouds for the word-level importances assigned by IDG, IG and SHAP, using the Word Cloud Python library [64]. In particular, the global importance score for each token was computed as the average of its importance scores across all sentences in which the token appeared. Common English stop words were filtered using the list provided in the Scikit-learn library [65]. Word clouds for 2-grams were generated similarly using the IDG data.

4.2 Results

4.2.1 Model Performance

As described in §4.1.2, after the fine-tuning phase, models were evaluated based on their performance metrics on the validation split, and the best performing model was selected based on these metrics.

Table 4.3 shows the performance of the fine-tuned models on the validation set. Figures 4.2 and 4.3 show the confusion matrix and ROC curve for BERT and XLNet, respectively. Training curves for both models are reported on §B.2.

Overall, BERT outperforms XLNet across all evaluation metrics except for precision, where XLNet shows a slight advantage. Therefore, BERT was selected as the final model for subsequent inference and explainability analyses.

TABLE 4.3: Performance metrics of fine-tuned BERT and XLNET on the validation split for the hate speech class. Boldface is used to highlight the best values.

	Metric				
Model	Accuracy	Precision	Recall	F_1-score	AUROC
BERT	**0.878**	0.855	**0.865**	**0.860**	**0.948**
XLNet	0.870	**0.862**	0.833	0.847	0.937

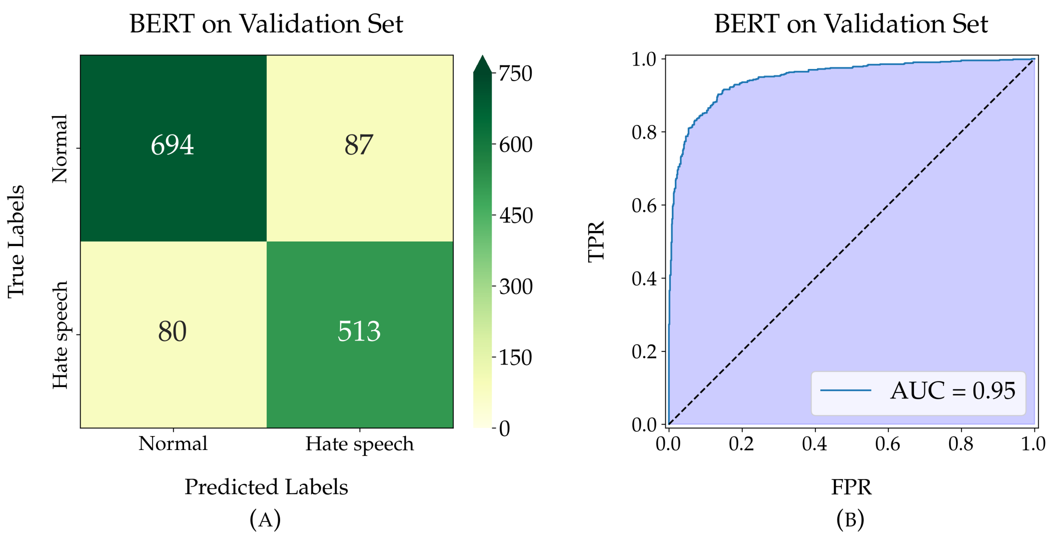

FIGURE 4.2: (A) Confusion matrix for BERT on the validation dataset. (B) ROC curve display for BERT on the validation set.

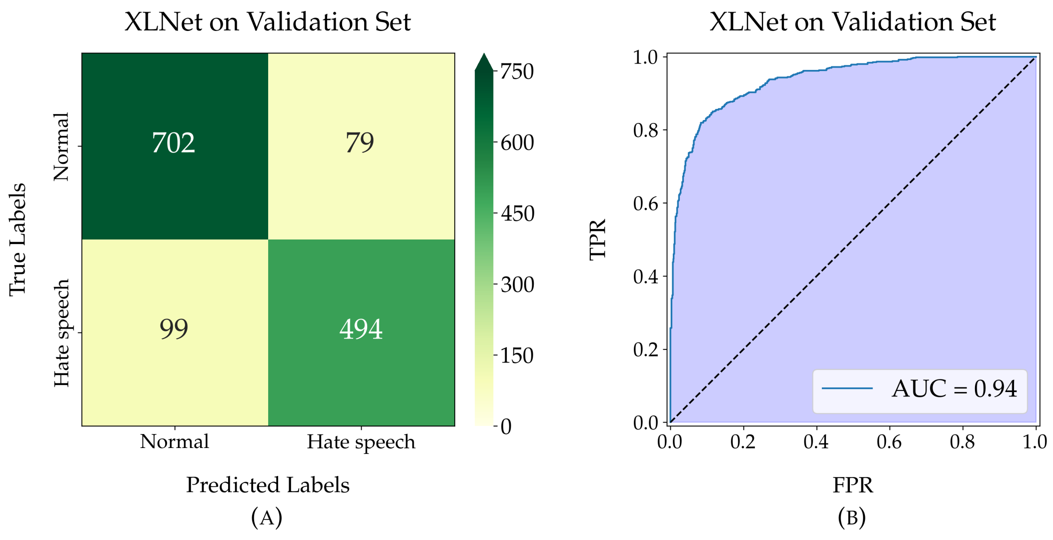

FIGURE 4.3: (A) Confusion matrix for XLNet on the validation dataset. (B) ROC curve display for XLNet on the validation set.

Table 4.4 shows the evaluation metrics of the selected BERT model on the test dataset. Figure 4.4 complements the previous information with the confusion matrix and ROC curve display for the model.

TABLE 4.4: Performance metrics of fine-tuned BERT on the test split for the hate speech class.

	Metric				
Model	Accuracy	Precision	Recall	F_1-score	AUROC
BERT	0.903	0.881	0.896	0.888	0.954

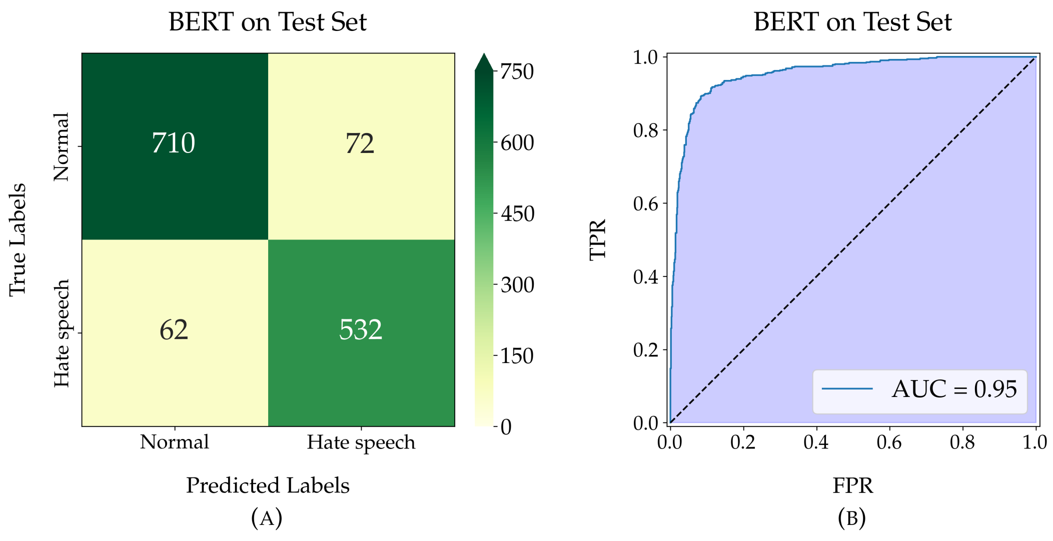

FIGURE 4.4: (A) Confusion matrix for BERT on the test dataset. (B) ROC curve display for BERT on the test set.

4.2.2 Explainability

4.2.2.1 Quantitative Evaluation

Word-level Attributions

Here, we focus on the quantitative analysis of the word-level importance attributions generated by IDG, IG and SHAP.

Figure 4.5 shows the analyzed plausibility metrics for the word-level importance attributions given by IDG, IG and SHAP. Note that IDG achieves a better agreement@*k* score at low-mid values of *k* than SHAP, which in turn achieves better scores than IG. Regarding the AUPRC, IDG achieves a value of 0.81, which outweighs the 0.77 value reached by SHAP, which in turn is greater than the 0.75 attributed to IG.

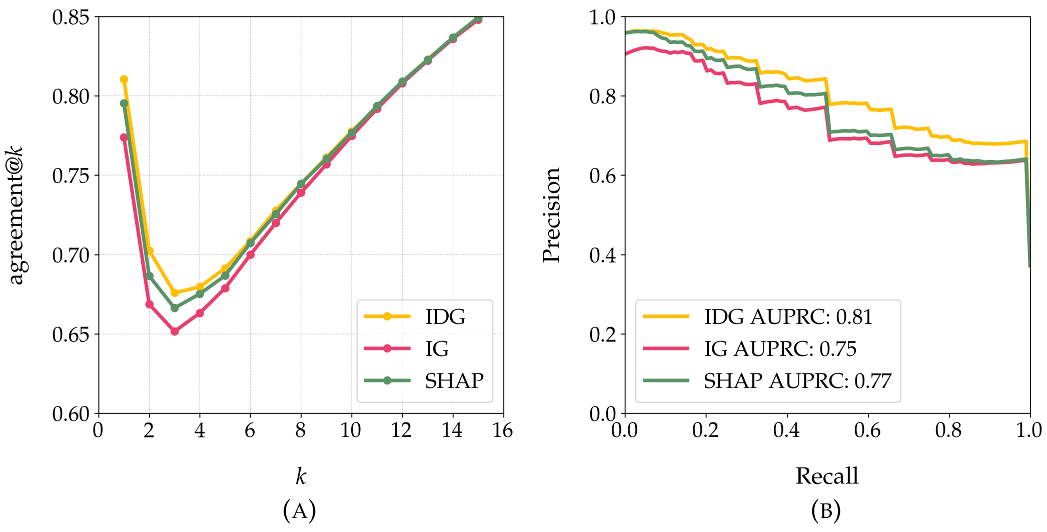

FIGURE 4.5: (A) Mean agreement@*k* for the three evaluated methods. (B) Precision-recall curve for the three evaluated methods.

Regarding faithfulness, figure 4.6 presents the comprehensiveness-threshold and sufficiency-threshold curves for the analyzed methods.

The comprehensiveness-threshold curve shows the drop in model confidence for each importance threshold. A higher area under the curve (AUCPC) indicates greater faithfulness, as it means that the explainability method identifies features that are truly relevant to the model's decision across more importance thresholds.

The sufficiency-threshold curve displays the variation in model confidence when, at a given importance threshold, only the relevant features are retained, and the rest of the features are removed. A higher Area Over the Sufficiency-threshold Curve (AOSPC) indicates that the explanation highlights a set of features that is sufficient for the model to maintain a similar level of confidence, thus demonstrating higher faithfulness.

For the studied data, the best performing method is SHAP, which achieves an AUCPC of 0.13 and an AOSPC of 0.73. IG follows with an AUCPC of 0.10 and an AOSPC of 0.70. Finally, IDG achieves the lowest values, with an AUCPC of 0.08 and an AOSPC of 0.68.

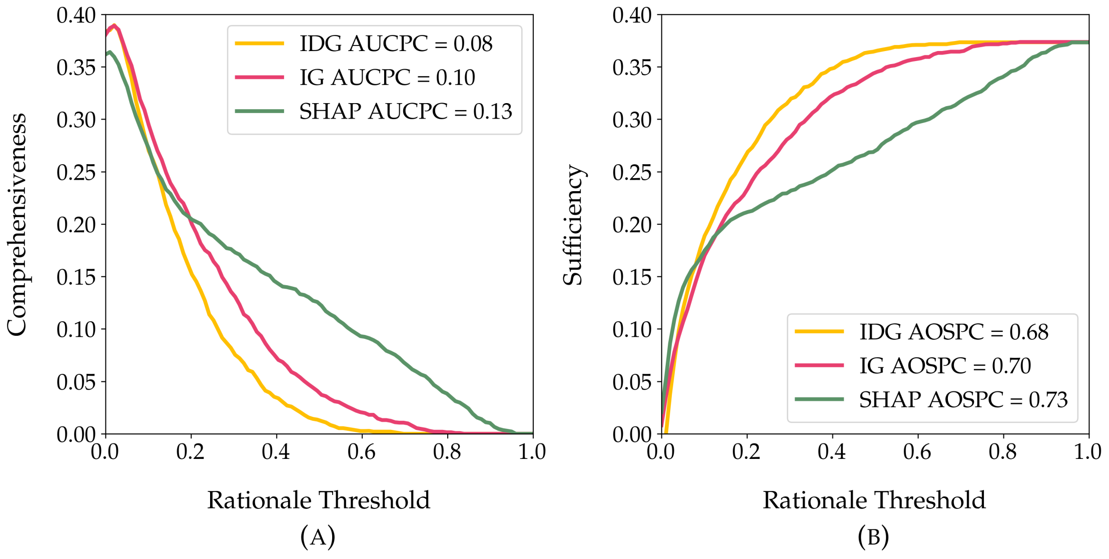

FIGURE 4.6: (A) Comprehensiveness-threshold curve for the three evaluated methods. (B) Sufficiency-threshold curve for the three evaluated methods.

High-order Interactions

This part is devoted to the qualitative analysis of the importance attributions for high-order interactions generated by IDG.

Figure 4.7 shows the support of each tree level across the test dataset, along with the mean maximum span coverage at each level. We can see that, indeed, support is approximately inversely proportional to the level, while the mean maximum span coverage tends to increase with the tree level.

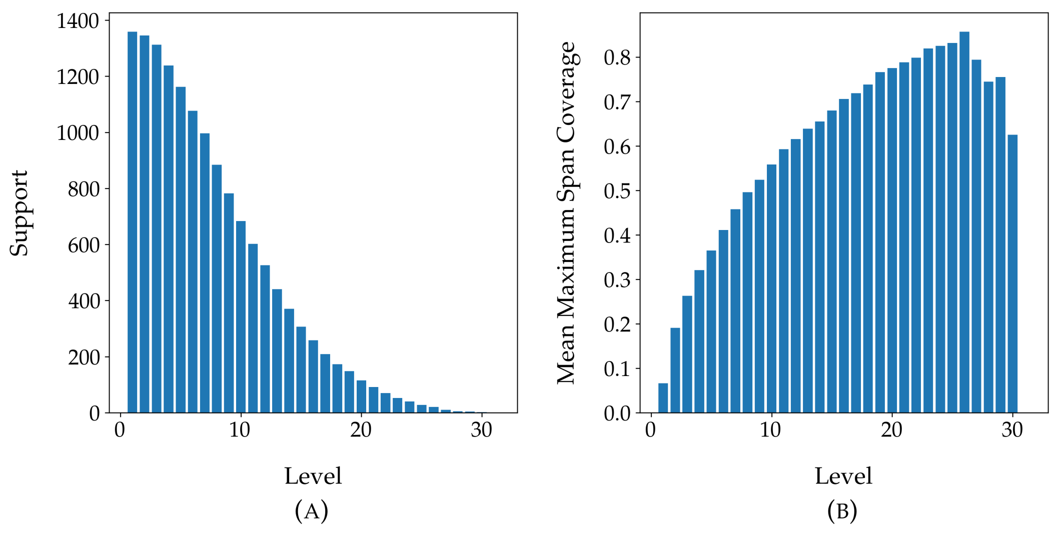

FIGURE 4.7: (A) Support at each tree depth-level across the test dataset. (B) Mean maximum span coverage at each tree-depth level across the test dataset.

Figures 4.8 and 4.9 present the results of the comprehensiveness and sufficiency analyses for the high-order interactions detected by IDG. Specifically, plot (A) of each figure shows the raw curves for each level, without any correction factor applied. Plot (B) shows each level curve multiplied by its correction factor, along with the global curve resulting from applying a weighted average with the corrections. The area in the plot corresponding to AUCPC_{HO} and AOSPC_{HO} is shaded. IDG achieves a global corrected AUCPC_{HO} of 0.13, and an AOSPC_{HO} of 0.72.

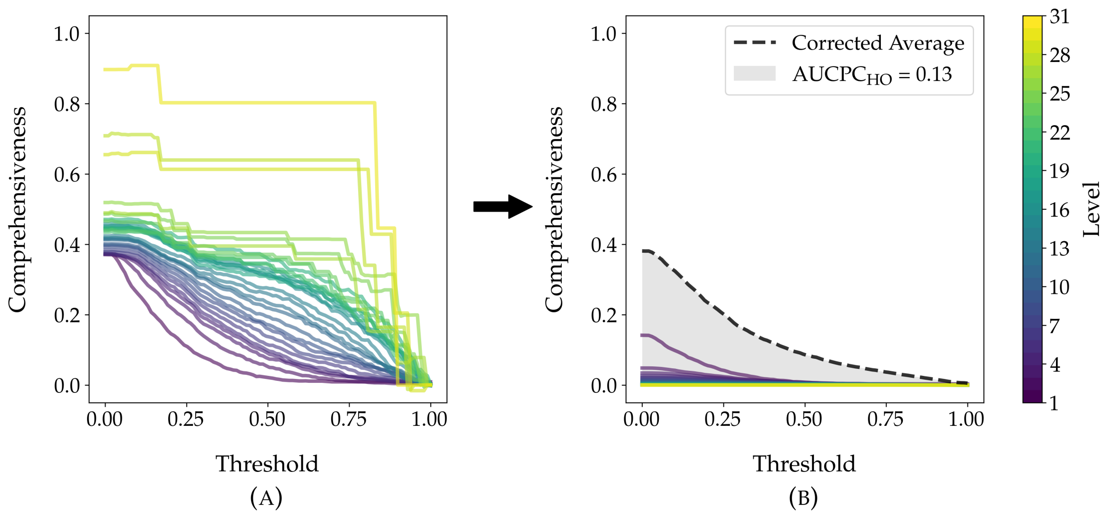

FIGURE 4.8: (A) Comprehensiveness-threshold curves for each level. (B) Corrected comprehensiveness-threshold curves along with their weighted average.

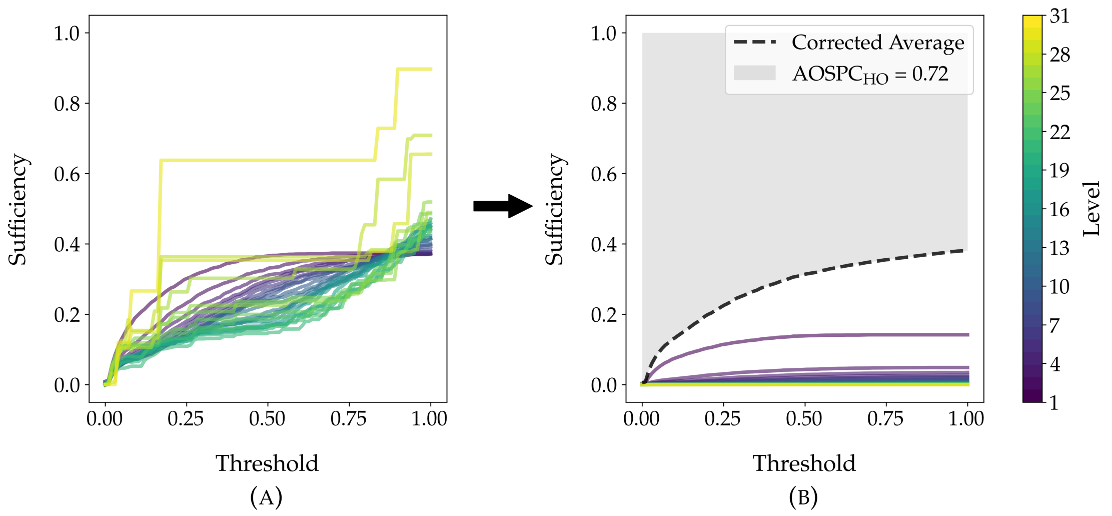

FIGURE 4.9: (A) Sufficiency-threshold curves for each level. (B) Corrected sufficiency-threshold curves along with their weighted average.

4.2.2.2 Qualitative Evaluation

In this section, we focus on the qualitative analysis of the generated explanations.

Figure 4.10 shows the explanation tree generated by IDG for the sequence “My ching chong is so cutie”, which the model correctly classified as hateful with 97% confidence. The tree is presented post-processed according to the visualization framework described in §2.2.3[5]. At the word level, IDG correctly identifies “ching” and “chong” as contributing to the hateful classification, although the token “cutie” is incorrectly attributed as hateful as well. Moving

[5]In particular, it has been depth-balanced using algorithm 1.

up the tree, we can see that the phrase "my ching chong" is attributed a high importance score toward the hate speech class, whereas "is so cutie" receives an opposite attribution. In the end, the whole sentence is correctly identified as hateful.

This example highlights how IDG can capture both fine-grained word-level attributions and higher-order interactions between phrases. By correctly identifying the key hateful terms and their context, IDG provides a richer explanation of the model's decision. Although the misaligned attribution of "cutie" suggests limitations at the word level, the overall hierarchical structure helps understand how positive and negative contributions combine to give the final prediction.

Additional complementary examples of explainability trees are shown in §B.3.

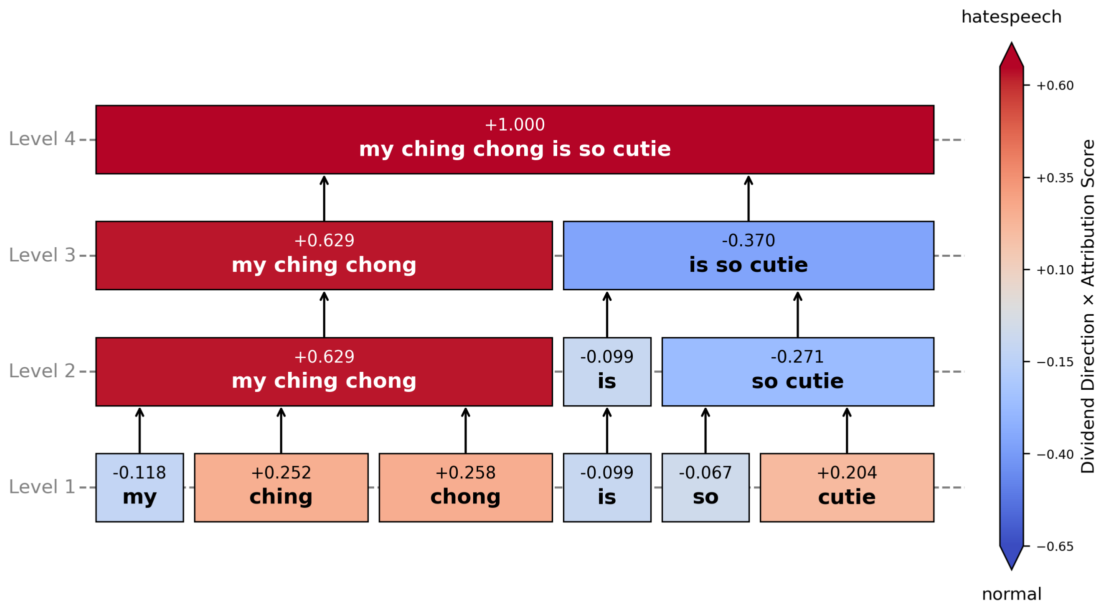

FIGURE 4.10: Explainability tree for the sequence with ID `1178101883602272256_-twitter`.

Figure 4.11 displays word clouds representing the globally most important tokens for predictions of the hate speech class, as identified by each explainability method. Each word cloud is color-coded with the palette used in previous figures to distinguish between the three methods. At first glance, the most prominent tokens are clearly aligned with highly offensive and discriminatory English terms, thus showing that all three methods succeed in highlighting toxic discursive patterns associated with hate speech.

Figure 4.12 shows a word cloud illustrating the globally most relevant 2-grams (combinations of two words) for predictions of the hate speech class, as computed by IDG. As before, the most important terms contain highly hateful and malicious vocabulary.

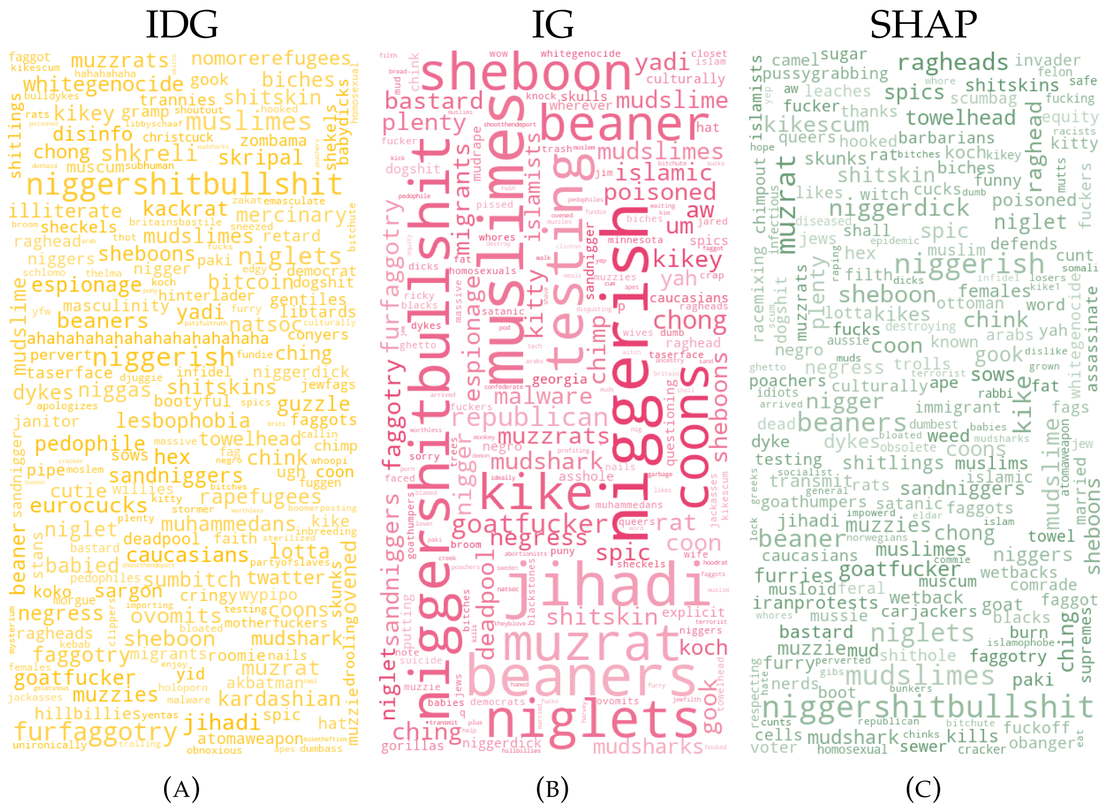

FIGURE 4.11: Wordclouds showing the globally most important tokens for the hate speech class, according to (A) IDG, (B) IG, and (C) SHAP.

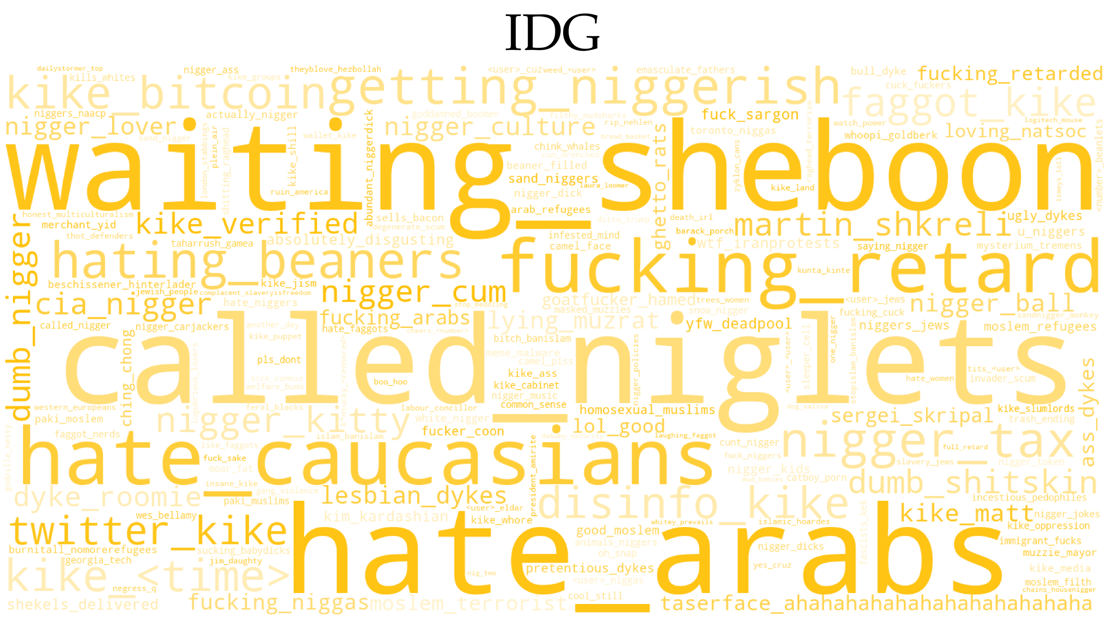

FIGURE 4.12: Wordcloud showing the globally most important 2-grams for the hate speech class, according to IDG. A 2-gram is represented by its constituting tokens separated by an underscore.

4.3 Discussion

Our study presents a comprehensive evaluation of IDG applied to a hate speech BERT-based detection model. We have processed and described our working dataset, assessed model performance and analyzed both quantitatively and qualitatively the quality of IDG explanations. The discussion below reflects on our main findings.

Dataset. Our descriptive analysis confirmed that our preprocessing and sampling strategy preserved class and platform distributions across the training, validation, and test splits. The consistent distributions ensured no that no major biases or structural inconsistencies affected downstream evaluations. The minor deviation observed in the Twitter-to-Gab ratio for the validation set has unlikely impacted the analysis, but it is worth noting it for future work.

Model Performance. BERT consistently outperformed XLNet in all evaluated metrics except for precision. Most importantly, given the detection nature of our task, the higher recall and F1-score of BERT, along with its training stability, made it the most suitable candidate for final evaluation and explainability analysis.

Word-level Attributions. We evaluated the quality of the word-level attributions given by IDG from the plausibility and faithfulness perspectives, and compared them with similar metrics for IG and SHAP. Our analysis revealed that IDG was the highest scoring method in terms of plausibility, thus suggesting that IDG better captures what humans consider relevant for hate speech detection, something particularly important in such a socially sensitive domain. On the other hand, IDG scored the lowest faithfulness, whereas IG and SHAP exhibited superior scores. This implies that IG and SHAP explanations can more accurately reflect the actual functional dependence of the model.

High-order Interactions. We evaluated the quality of the high-order interactions importance attributions given by IDG by using the modified plausibility metrics introduced in the theoretical framework of this study. The corrected AUCPC_{HO} and AOSPC_{HO} values show that IDG's high-order attributions can better capture the true functional dependencies of the model. Notably, these values exceed those of IDG's word-level faithfulness metrics and are comparable to the word-level faithfulness scores of SHAP.

Qualitative Assessment. Case studies of specific text sequences reinforced quantitative findings. In particular, IDG's hierarchical attributions for a number of examples revealed how hateful terms were integrated into the model's classification reasoning. Word clouds confirmed that all methods successfully highlight toxic discursive patterns associated with hate speech, with IDG's 2-gram analysis showing phrase-level toxicity.

Chapter 5

Conclusions and Future Work

In this chapter, we present the conclusions derived from this thesis, as well as the limitations of our work and the future work lines that might stem from it.

5.1 Conclusions

In this work, we have theoretically and empirically explored IDG as an explainability method capable of attributing importance values not only to individual input features, but also to their high-order interactions. Moreover, we have explored different metrics for the quantitative evaluation of the quality of such attributions, and proposed a corrected framework to benchmark the faithfulness of high-order interaction importance attributions for text data. Finally, we have investigated the applicability and effectiveness of IDG for explaining a BERT-based hate speech detection model.

Our results suggest that, while IDG lags slightly behind IG and SHAP in terms of word-level attribution faithfulness, it surpasses both techniques in terms of plausibility, indicating a closer alignment with human intuition when it comes to identifying hateful content. Furthermore, the ability of IDG to explain high-order interactions improved faithfulness scores, rivaling traditional word-level interactions. This showcases the potential of high-order importance attributions for capturing the complex reasoning behind text model predictions.

Qualitative analyses proved that IDG can offer interpretable and meaningful explanations, that align with human expectations and known toxic discursive patterns. Most importantly, IDG allows extracting insights on how the hierarchical structure of text and word combinations contribute to the output of a model, giving a more exhaustive view of the model's decision-making process.

To sum up, IDG represents a promising method for improving explainability in NLP.

5.2 Future Work

One key limitation of our analysis is the reliance on plausibility metrics that depend on annotator rationales, that are assumed to be complete and accurate. Despite considering the outputs from at least two independent annotators, human rationales may not accurately capture all relevant tokens. Furthermore, annotations were only available at the word level, which prevented the analysis of plausibility at higher orders. Further studies should incorporate data from a larger pool of annotators and, ideally, include structured annotations in constituency parse trees.

Another methodological limitation relates to faithfulness metrics. Both comprehensiveness and sufficiency rely on perturbations of the input text, which can result in unrealistic examples or instances that lie too far from the data manifold. This may lead to misleading quality assessments, especially when the model tends to behave unpredictably on out-of-distribution inputs. Future work could explore evaluation methodologies that preserve semantic integrity and proximity to the data distribution by, for instance, using causal, counterfactual or generative approaches.

Another relevant limitation of our study is the fact that evaluation has been performed on a single dataset. To better understand the generalizability and robustness of the explainability methods, a broader quantitative assessment on different datasets should be performed.

Beyond, it would be interesting to expand the qualitative assessment of IDG by performing user studies to determine whether explanations are generally agreed upon by different audiences. Such evaluations could provide more in-depth insights into the interpretability and usefulness of the explanations in real-world scenarios.

Future studies could also investigate the performance of detection models and explainability methods in a multilingual context. Extending analyses to languages beyond English would help evaluate the generalizability of the methods and explore the unique linguistic challenges associated with different languages and cultural contexts.

Our dataset was entirely composed of text data; links and audiovisual content were removed or replaced with textual placeholders. However, content in social media goes far beyond textual data, and includes links, audio and video content. Future work could explore how to perform multimodal detection and explainability analysis, and how could IDG integrate in such frameworks.

Bibliography

[1] Marc Aureli. *Meditacions*. Llibres de l'Índex, 2008.

[2] Ian Goodfellow et al. *Deep learning*. MIT Press Cambridge, 2016.

[3] Michael Tsang, Dehua Cheng, and Yan Liu. *Detecting Statistical Interactions from Neural Network Weights*. 2018. arXiv: `1705.04977 [stat.ML]`.

[4] Jianbo Chen and Michael Jordan. "Ls-tree: Model interpretation when the data are linguistic". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. 2020.

[5] Theo MV Janssen and Barbara H Partee. "Compositionality". In: *Handbook of logic and language*. Elsevier, 1997.

[6] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. ""Why should i trust you?" Explaining the predictions of any classifier". In: *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 2016.

[7] Scott M Lundberg and Su-In Lee. "A unified approach to interpreting model predictions". In: *Advances in neural information processing systems* (2017).

[8] Scott M. Lundberg et al. "From local explanations to global understanding with explainable AI for trees". In: *Nature Machine Intelligence* (2020).

[9] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. "Axiomatic attribution for deep networks". In: *International conference on machine learning*. PMLR. 2017.

[10] W. James Murdoch, Peter J. Liu, and Bin Yu. *Beyond Word Importance: Contextual Decomposition to Extract Interactions from LSTMs*. 2018. arXiv: `1801.05453 [cs.CL]`.

[11] Chandan Singh, W. James Murdoch, and Bin Yu. *Hierarchical interpretations for neural network predictions*. 2019. arXiv: `1806.05337 [cs.LG]`.

[12] Michel Grabisch and Marc Roubens. "An axiomatic approach to the concept of interaction among players in cooperative games". In: *International Journal of game theory* (1999).

[13] Edith Elkind et al. "On the computational complexity of weighted voting games". In: *Annals of Mathematics and Artificial Intelligence* (2009).

[14] Hanjie Chen, Guangtao Zheng, and Yangfeng Ji. "Generating Hierarchical Explanations on Text Classification via Feature Interaction Detection". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2020.

[15] Joseph D. Janizek, Pascal Sturmfels, and Su-In Lee. "Explaining explanations: Axiomatic feature interactions for deep networks". In: *Journal of Machine Learning Research* (2021).

[16] Sandipan Sikdar, Parantapa Bhattacharya, and Kieran Heese. "Integrated Directional Gradients: Feature Interaction Attribution for Neural NLP Models". In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, 2021.

[17] Matthew .L Williams et al. "Hate in the machine: Anti-Black and Anti-Muslim social media posts as predictors of offline racially and religiously aggravated crime". In: *The British Journal of Criminology* (2020).

[18] Nedjma Ousidhoum et al. "Multilingual and Multi-Aspect Hate Speech Analysis". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, 2019.

[19] Jing Qian et al. "Learning to Decipher Hate Symbols". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, 2019.

[20] Ona de Gibert et al. "Hate Speech Dataset from a White Supremacy Forum". In: *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*. Association for Computational Linguistics, 2018.

[21] Jing Qian et al. "Leveraging Intra-User and Inter-User Representation Learning for Automated Hate Speech Detection". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. Association for Computational Linguistics, 2018.

[22] Valerio Basile et al. "Semeval-2019 task 5: Multilingual detection of hate speech against immigrants and women in twitter". In: *Proceedings of the 13th international workshop on semantic evaluation*. 2019.

[23] Cristina Bosco et al. "Overview of the evalita 2018 hate speech detection task". In: *Ceur workshop proceedings*. CEUR. 2018.

[24] Aymé Arango, Jorge Pérez, and Barbara Poblete. "Hate speech detection is not as easy as you may think: A closer look at model validation". In: *Proceedings of the 42nd international acm sigir conference on research and development in information retrieval*. 2019.

[25] Tommi Gröndahl et al. "All You Need is "Love": Evading Hate Speech Detection". In: *Proceedings of the 11th ACM Workshop on Artificial Intelligence and Security*. AISec '18. Association for Computing Machinery, 2018.

[26] Binny Mathew et al. "Hatexplain: A benchmark dataset for explainable hate speech detection". In: *Proceedings of the AAAI conference on artificial intelligence*. 2021.

[27] L. S. Shapley. "17. A Value for n-Person Games". In: *Contributions to the Theory of Games, Volume II*. Princeton University Press, 1953.

[28] Imma Curiel. *Cooperative Game Theory and Applications. Cooperative Games Arising from combinatorial Optimization Problems*. Springer New York, 1997.

[29] John C. Harsanyi. "A Simplified Bargaining Model for the n-Person Cooperative Game". In: *International Economic Review* (1963).

[30] Pierre Dehez. "On Harsanyi Dividends and Asymmetric Values". In: *International Game Theory Review* (2017).

[31] David Baehrens et al. "How to explain individual classification decisions". In: *The Journal of Machine Learning Research* (2010).

[32] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. *Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps*. 2014. arXiv: `1312.6034 [cs.CV]`.

[33] Marco Ancona et al. "Gradient-Based Attribution Methods". In: *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*. Springer-Verlag, 2022.

[34] Ali Aghababaei et al. *Application of integrated gradients explainability to sociopsychological semantic markers*. 2025. arXiv: `2503.04989 [cs.CL]`.

[35] Jasmijn Bastings et al. ""Will You Find These Shortcuts?" A Protocol for Evaluating the Faithfulness of Input Salience Methods for Text Classification". In: *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2022.

[36] Soumya Sanyal and Xiang Ren. "Discretized Integrated Gradients for Explaining Language Models". In: *Proceedings of the 2021 Conference on Empirical Methods in Natural

Language Processing. Ed. by Marie-Francine Moens et al. Association for Computational Linguistics, 2021.

[37] Joseph Enguehard. "Sequential Integrated Gradients: a simple but effective method for explaining language models". In: *Findings of the Association for Computational Linguistics: ACL 2023*. Association for Computational Linguistics, 2023.

[38] Noam Chomsky and Howard Lasnik. "The Theory of Principles and Parameters". In: *An International Handbook of Contemporary Research*. De Gruyter Mouton, 1993.

[39] C. Pollard and I.A. Sag. *Head-Driven Phrase Structure Grammar*. Studies in Contemporary Linguistics. University of Chicago Press, 1994.

[40] A. Carnie. *Syntax: A Generative Introduction*. Introducing Linguistics. Wiley, 2013.

[41] Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models*. 3rd. 2025.

[42] Peng Qi et al. "Stanza: A Python Natural Language Processing Toolkit for Many Human Languages". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Association for Computational Linguistics, 2020.

[43] Mitch Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. "Building a large annotated corpus of English: The Penn Treebank". In: *Computational linguistics* (1993).

[44] Jay DeYoung et al. "ERASER: A Benchmark to Evaluate Rationalized NLP Models". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2020.

[45] Alon Jacovi and Yoav Goldberg. "Towards Faithfully Interpretable NLP Systems: How Should We Define and Evaluate Faithfulness?" In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2020.

[46] Pepa Atanasova. "A Diagnostic Study of Explainability Techniques for Text Classification". In: *Accountable and Explainable Methods for Complex Reasoning over Text*. Springer Nature Switzerland, 2024.

[47] Jonathan Kamp, Lisa Beinborn, and Antske Fokkens. "Dynamic Top-k Estimation Consolidates Disagreement between Feature Attribution Methods". In: *The 2023 Conference on Empirical Methods in Natural Language Processing*. 2023.

[48] Wojciech Samek et al. "Evaluating the Visualization of What a Deep Neural Network Has Learned". In: *IEEE Transactions on Neural Networks and Learning Systems* (2017).

[49] David Saeteros et al. *The Written Self: Decoding Personality and Sex Differences Through Explainable AI*. 2025.

[50] Martin J Halvey and Mark T Keane. "An assessment of tag presentation techniques". In: *Proceedings of the 16th international conference on World Wide Web*. 2007.

[51] Wikipedia contributors. *Gab (social network)*. 2025.

[52] Binny Mathew et al. "Spread of hate speech in online social media". In: *Proceedings of the 10th ACM conference on web science*. 2019.

[53] Thomas Davidson et al. "Automated hate speech detection and the problem of offensive language". In: *Proceedings of the international AAAI conference on web and social media*. 2017.

[54] Paula Fortuna and Sérgio Nunes. "A Survey on Automatic Detection of Hate Speech in Text". In: *ACM Comput. Surv.* (2018).

[55] *Polars: Blazingly fast DataFrames in Rust, Python, Node.js, R, and SQL*. `https://pola.rs/`.

[56] Jacob Devlin et al. "BERT: Pre-training of deep bidirectional transformers for language understanding". In: *Proceedings of the 2019 conference of the North American chapter of the*

association for computational linguistics: human language technologies, volume 1 (long and short papers). 2019.

[57] Zhilin Yang et al. "XLNet: Generalized autoregressive pretraining for language understanding". In: *Advances in neural information processing systems* (2019).

[58] Thomas Wolf et al. "Transformers: State-of-the-Art Natural Language Processing". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, 2020.

[59] Adam Paszke et al. "PyTorch: an imperative style, high-performance deep learning library". In: *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Curran Associates Inc., 2019.

[60] Narine Kokhlikyan et al. *Captum: A unified and generic model interpretability library for PyTorch*. 2020. arXiv: `2009.07896 [cs.LG]`.

[61] Justin Mott et al. "Supplementary guidelines for ETTB 2.0". In: *University of Pennsylvania* (2009).

[62] Charles Pierse. *Transformers Interpret*. `https://github.com/cdpierse/transformers-interpret`. Version 0.5.2. 2021.

[63] J. D. Hunter. "Matplotlib: A 2D graphics environment". In: *Computing in Science & Engineering* (2007).

[64] Andreas C. Mueller. *Wordcloud*. `https://github.com/amueller/wordcloud`. Version 1.9.1. 2023.

[65] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* (2011).

[66] *Conda: OS-agnostic, system-level binary package and environment manager.* `https://anaconda.org/anaconda/conda`.

[67] Arvind Subramaniam, Aryan Mehra, and Sayani Kundu. *Exploring Hate Speech Detection with HateXplain and BERT*. 2022. arXiv: `2208.04489 [cs.CL]`.

List of Figures

List of Tables

List of Algorithms

Appendices

Appendix A

Computational Environment and Hardware Specifications

This appendix provides an overview of the computational environment and hardware specifications used to execute the experiments and run the code developed throughout the project.

A.1 Hardware Specifications

All the experiments were conducted on a local machine with the following hardware configuration:

- **CPU:** Intel® Core™ Ultra 9 185H × 22 @ 5.1 GHz.
- **RAM:** 32.0 GiB.
- **GPU:** NVIDIA® GeForce RTX™ 4060 Laptop GPU with 8.0 GB VRAM.
- **Storage:** 1.0 TB SSD.
- **Operating System:** Ubuntu 24.04.2 LTS.

A.2 Python Environment

The Python environment was managed using Conda [66], and includes the packages listed in table A.1.

TABLE A.1: Conda packages and their versions.

Package	Version	Package	Version
_libgcc_mutex	0.1	_openmp_mutex	4.5
absl-py	2.1.0	adwaita-icon-theme	47.0
aiohappyeyeballs	2.4.4	aiohttp	3.11.11
aiosignal	1.3.2	annotated-types	0.7.0
anyio	4.8.0	aom	3.9.1
argon2-cffi	23.1.0	argon2-cffi-bindings	21.2.0
arrow	1.3.0	asttokens	3.0.0
astunparse	1.6.3	async-lru	2.0.4
at-spi2-atk	2.38.0	at-spi2-core	2.40.3
atk-1.0	2.38.0	attr	2.5.1
attrs	24.3.0	aws-c-auth	0.8.0
aws-c-cal	0.8.0	aws-c-common	0.9.31
aws-c-compression	0.3.0	aws-c-event-stream	0.5.0
aws-c-http	0.9.0	aws-c-io	0.15.0
aws-c-mqtt	0.11.0	aws-c-s3	0.7.0
aws-c-sdkutils	0.2.0	aws-checksums	0.2.0
aws-crt-cpp	0.29.0	aws-sdk-cpp	1.11.407

Continued on next page

Table A.1 – continued from previous page

Package	Version	Package	Version
azure-core-cpp	1.14.0	azure-identity-cpp	1.10.0
azure-storage-blobs-cpp	12.13.0	azure-storage-common-cpp	12.8.0
azure-storage-files-datalake-cpp	12.12.0	babel	2.16.0
backcall	0.2.0	beautifulsoup4	4.12.3
bleach	6.2.0	bleach-with-css	6.2.0
blosc	1.21.6	bokeh	3.6.2
branca	0.8.1	brotli	1.1.0
brotli-bin	1.1.0	brotli-python	1.1.0
brunsli	0.1	bzip2	1.0.8
c-ares	1.34.4	c-blosc2	2.15.2
ca-certificates	2025.4.26	cached-property	1.5.2
cached_property	1.5.2	cachetools	5.5.0
cairo	1.18.2	captum	0.7.0
catalogue	2.0.10	certifi	2025.4.26
cffi	1.17.1	charls	2.4.2
charset-normalizer	3.4.1	click	8.1.8
cloudpathlib	0.21.0	cloudpickle	3.1.0
colorama	0.4.6	colorcet	3.1.0
colour	0.1.5	comm	0.2.2
confection	0.1.5	contourpy	1.3.1
cpython	3.12.8	cucim	24.12.00
cuda-cccl_linux-64	12.8.55	cuda-crt-dev_linux-64	12.8.61
cuda-crt-tools	12.8.61	cuda-cudart	12.8.57
cuda-cudart-dev	12.8.57	cuda-cudart-dev_linux-64	12.8.57
cuda-cudart-static	12.8.57	cuda-cudart-static_linux-64	12.8.57
cuda-cudart_linux-64	12.8.57	cuda-cupti	12.8.57
cuda-nvcc-dev_linux-64	12.8.61	cuda-nvcc-impl	12.8.61
cuda-nvcc-tools	12.8.61	cuda-nvrtc	12.8.61
cuda-nvtx	12.8.55	cuda-nvvm-dev_linux-64	12.8.61
cuda-nvvm-impl	12.8.61	cuda-nvvm-tools	12.8.61
cuda-profiler-api	12.8.55	cuda-python	12.6.0
cuda-version	12.8	cudf	24.12.00
cudf-polars	24.12.00	cudf_kafka	24.12.00
cudnn	9.3.0.75	cugraph	24.12.00
cuml	24.12.00	cuproj	24.12.00
cupy	13.3.0	cupy-core	13.3.0
cuspatial	24.12.00	custreamz	24.12.00
cuvs	24.12.00	cuxfilter	24.12.00
cycler	0.12.1	cymem	2.0.11
cyrus-sasl	2.1.27	cython-blis	1.0.1
cytoolz	1.0.1	daal4py	2024.6.0
dal	2024.6.0	dask	2024.11.2
dask-core	2024.11.2	dask-cuda	24.12.00
dask-cudf	24.12.00	dask-expr	1.1.19
datasets	2.14.4	datashader	0.16.3
dav1d	1.2.1	dbus	1.13.6
debugpy	1.8.11	decorator	5.1.1
defusedxml	0.7.1	dill	0.3.7
distributed	2024.11.2	distributed-ucxx	0.41.00
distro	1.9.0	dlpack	0.8
emoji	2.14.1	en-core-web-sm	3.8.0
entrypoints	0.4	epoxy	1.5.10
exceptiongroup	1.2.2	executing	2.1.0
expat	2.6.4	fastrlock	0.8.3
filelock	3.16.1	flatbuffers	24.3.25
fmt	11.0.2	folium	0.19.4
font-ttf-dejavu-sans-mono	2.37	font-ttf-inconsolata	3.000
font-ttf-source-code-pro	2.038	font-ttf-ubuntu	0.83
fontconfig	2.15.0	fonts-conda-ecosystem	1
fonts-conda-forge	1	fonttools	4.55.3
fqdn	1.5.1	freetype	2.12.1
freexl	2.0.0	fribidi	1.0.10
frozenlist	1.5.0	fsspec	2024.12.0
gast	0.6.0	gdk-pixbuf	2.42.12
geopandas	1.0.1	geopandas-base	1.0.1
geos	3.13.0	geotiff	1.7.3
gflags	2.2.2	giflib	5.2.2
glib-tools	2.82.2	glog	0.7.1
gmp	6.3.0	gmpy2	2.1.5

Continued on next page

Table A.1 – continued from previous page

Package	Version	Package	Version
google-pasta	0.2.0	graphite2	1.3.13
graphviz	12.2.1	grpcio	1.65.5
gtk3	3.24.43	gts	0.7.6
h11	0.14.0	h2	4.1.0
h5py	3.12.1	harfbuzz	10.4.0
hdf5	1.14.4	hicolor-icon-theme	0.17
holoviews	1.20.0	hpack	4.0.0
httpcore	1.0.7	httpx	0.28.1
huggingface_hub	0.27.1	hyperframe	6.0.1
icu	75.1	idna	3.10
imagecodecs	2024.9.22	imageio	2.36.1
importlib-metadata	8.5.0	importlib_resources	6.5.2
ipykernel	6.29.5	ipython	7.34.0
ipywidgets	8.1.5	isoduration	20.11.0
jedi	0.19.2	jinja2	3.1.5
jiter	0.8.2	joblib	1.4.2
json-c	0.18	json5	0.10.0
jsonpointer	3.0.0	jsonschema	4.23.0
jsonschema-specifications	2024.10.1	jsonschema-with-format-nongpl	4.23.0
jupyter	1.1.1	jupyter-lsp	2.2.5
jupyter-server-proxy	4.4.0	jupyter_client	8.6.3
jupyter_console	6.6.3	jupyter_core	5.7.2
jupyter_events	0.11.0	jupyter_server	2.15.0
jupyter_server_terminals	0.5.3	jupyterlab	4.3.4
jupyterlab_pygments	0.3.0	jupyterlab_server	2.27.3
jupyterlab_widgets	3.0.13	jxrlib	1.1
keras	3.8.0	keyutils	1.6.1
kiwisolver	1.4.8	krb5	1.21.3
langcodes	3.4.1	language-data	1.3.0
lazy-loader	0.4	lazy_loader	0.4
lcms2	2.16	ld_impl_linux-64	2.43
lerc	4.0.0	libabseil	20240722.0
libaec	1.1.3	libarchive	3.7.7
libarrow	17.0.0	libarrow-acero	17.0.0
libarrow-dataset	17.0.0	libarrow-substrait	17.0.0
libavif16	1.1.1	libblas	3.9.0
libbrotlicommon	1.1.0	libbrotlidec	1.1.0
libbrotlienc	1.1.0	libcap	2.71
libcblas	3.9.0	libcrc32c	1.1.2
libcublas	12.8.3.14	libcublas-dev	12.8.3.14
libcucim	24.12.00	libcudf	24.12.00
libcudf_kafka	24.12.00	libcufft	11.3.3.41
libcufile	1.13.0.11	libcufile-dev	1.13.0.11
libcugraph	24.12.00	libcugraph_etl	24.12.00
libcugraphops	24.12.00	libcuml	24.12.00
libcumlprims	24.12.00	libcups	2.3.3
libcurand	10.3.9.55	libcurand-dev	10.3.9.55
libcurl	8.11.1	libcusolver	11.7.2.55
libcusolver-dev	11.7.2.55	libcusparse	12.5.7.53
libcusparse-dev	12.5.7.53	libcuspatial	24.12.00
libcuvs	24.12.00	libde265	1.0.15
libdeflate	1.22	libedit	3.1.20240808
libev	4.33	libevent	2.1.12
libexpat	2.6.4	libfabric	1.22.0
libfabric1	1.22.0	libffi	3.4.2
libgcc	14.2.0	libgcc-ng	14.2.0
libgcrypt-lib	1.11.0	libgd	2.3.3
libgdal-core	3.10.0	libgfortran	14.2.0
libgfortran5	14.2.0	libglib	2.82.2
libgoogle-cloud	2.30.0	libgoogle-cloud-storage	2.30.0
libgpg-error	1.51	libgrpc	1.65.5
libheif	1.18.2	libhwloc	2.11.2
libhwy	1.1.0	libiconv	1.17
libjpeg-turbo	3.0.0	libjxl	0.11.1
libkml	1.3.0	libkvikio	24.12.01
liblapack	3.9.0	libllvm14	14.0.6
liblzma	5.6.3	liblzma-devel	5.6.3
libmagma	2.8.0	libmagma_sparse	2.8.0
libnghttp2	1.64.0	libnl	3.11.0

Continued on next page

Table A.1 – continued from previous page

Package	Version	Package	Version
`libnsl`	2.0.1	`libntlm`	1.8
`libnvjitlink`	12.8.61	`libnvjpeg`	12.3.5.57
`libopenblas`	0.3.28	`libparquet`	17.0.0
`libpng`	1.6.45	`libprotobuf`	5.27.5
`libraft`	24.12.00	`libraft-headers`	24.12.00
`libraft-headers-only`	24.12.00	`librdkafka`	2.5.3
`libre2-11`	2024.07.02	`librmm`	24.12.01
`librsvg`	2.58.4	`librttopo`	1.1.0
`libsentencepiece`	0.2.0	`libsodium`	1.0.20
`libspatialite`	5.1.0	`libsqlite`	3.47.2
`libssh2`	1.11.1	`libstdcxx`	14.2.0
`libstdcxx-ng`	14.2.0	`libsystemd0`	256.9
`libthrift`	0.21.0	`libtiff`	4.7.0
`libtorch`	2.4.1	`libucxx`	0.41.00
`libudev1`	257.2	`libutf8proc`	2.8.0
`libuuid`	2.38.1	`libuv`	1.49.2
`libwebp-base`	1.5.0	`libxcb`	1.17.0
`libxcrypt`	4.4.36	`libxgboost`	2.1.2
`libxkbcommon`	1.8.0	`libxml2`	2.13.5
`libzlib`	1.3.1	`libzopfli`	1.0.3
`lime`	0.2.0.1	`linkify-it-py`	2.0.3
`llvm-openmp`	19.1.6	`llvmlite`	0.43.0
`locket`	1.0.0	`lz4`	4.3.3
`lz4-c`	1.9.4	`lzo`	2.10
`mapclassify`	2.8.1	`marisa-trie`	1.2.1
`markdown`	3.6	`markdown-it-py`	3.0.0
`markupsafe`	3.0.2	`matplotlib-base`	3.10.0
`matplotlib-inline`	0.1.7	`mdit-py-plugins`	0.4.2
`mdurl`	0.1.2	`minizip`	4.0.7
`mistune`	3.1.0	`mkl`	2023.2.0
`ml_dtypes`	0.4.0	`mpc`	1.3.1
`mpfr`	4.2.1	`mpi`	1.0.1
`mpich`	4.2.3	`mpmath`	1.3.0
`msgpack-python`	1.1.0	`multidict`	6.1.0
`multipledispatch`	0.6.0	`multiprocess`	0.70.15
`munkres`	1.1.4	`murmurhash`	1.0.10
`namex`	0.0.8	`nbclient`	0.10.2
`nbconvert-core`	7.16.5	`nbformat`	5.10.4
`nccl`	2.24.3.1	`ncurses`	6.5
`nest-asyncio`	1.6.0	`networkx`	3.4.2
`nltk`	3.9.1	`nodejs`	22.12.0
`notebook`	7.3.2	`notebook-shim`	0.2.4
`nspr`	4.36	`nss`	3.107
`numba`	0.60.0	`numba-cuda`	0.0.17.1
`numpy`	1.26.4	`nvcomp`	4.1.0.6
`nvtx`	0.2.10	`nx-cugraph`	24.12.00
`openai`	1.61.0	`openjpeg`	2.5.3
`openssl`	3.5.0	`opt_einsum`	3.4.0
`optree`	0.13.1	`orc`	2.0.2
`overrides`	7.7.0	`packaging`	24.2
`pandas`	2.2.3	`pandocfilters`	1.5.0
`panel`	1.5.5	`pango`	1.56.1
`param`	2.2.0	`parso`	0.8.4
`partd`	1.4.2	`patsy`	1.0.1
`pcre2`	10.44	`pdf2image`	1.17.0
`pexpect`	4.9.0	`pickleshare`	0.7.5
`pillow`	11.1.0	`pip`	24.3.1
`pixman`	0.44.2	`pkgutil-resolve-name`	1.3.10
`platformdirs`	4.3.6	`polars`	1.14.0
`poppler`	24.12.0	`poppler-data`	0.4.12
`preshed`	3.0.9	`proj`	9.5.1
`prometheus_client`	0.21.1	`prompt-toolkit`	3.0.48
`prompt_toolkit`	3.0.48	`propcache`	0.2.1
`protobuf`	5.27.5	`psutil`	6.1.1
`pthread-stubs`	0.4	`ptyprocess`	0.7.0
`pure_eval`	0.2.3	`py-xgboost`	2.1.2
`pyarrow`	17.0.0	`pyarrow-core`	17.0.0
`pycparser`	2.22	`pyct`	0.5.0
`pydantic`	2.10.6	`pydantic-core`	2.27.2

Continued on next page

Table A.1 – continued from previous page

Package	Version	Package	Version
pydot	3.0.4	pygments	2.19.1
pygraphviz	1.14	pylibcudf	24.12.00
pylibcugraph	24.12.00	pylibraft	24.12.00
pynvjitlink	0.5.0	pynvml	11.4.1
pyogrio	0.10.0	pyparsing	3.2.1
pyproj	3.7.0	pysocks	1.7.1
python	3.12.8	python-confluent-kafka	2.5.3
python-dateutil	2.9.0.post0	python-fastjsonschema	2.21.1
python-flatbuffers	24.12.23	python-graphviz	0.20.3
python-json-logger	2.0.7	python-tzdata	2024.2
python-xxhash	3.5.0	python_abi	3.12
pytorch	2.4.1	pytz	2024.1
pyviz_comms	3.0.4	pywavelets	1.8.0
pyyaml	6.0.2	pyzmq	26.2.0
qhull	2020.2	raft-dask	24.12.00
rapids	24.12.00	rapids-dask-dependency	24.12.00
rapids-xgboost	24.12.00	rav1e	0.6.6
rdma-core	55.0	re2	2024.07.02
readline	8.2	referencing	0.35.1
regex	2024.11.6	requests	2.32.3
rfc3339-validator	0.1.4	rfc3986-validator	0.1.1
rich	13.9.4	rmm	24.12.01
rpds-py	0.22.3	s2n	1.5.6
safetensors	0.5.2	scikit-image	0.24.0
scikit-learn	1.6.1	scikit-learn-intelex	2024.6.0
scipy	1.15.1	seaborn	0.13.2
seaborn-base	0.13.2	send2trash	1.8.3
sentencepiece	0.1.99	sentencepiece-python	0.2.0
sentencepiece-spm	0.2.0	setuptools	75.8.0
shap	0.46.0	shapely	2.0.6
shapiq	1.2.2	shellingham	1.5.4
simpervisor	1.0.0	six	1.17.0
sleef	3.7	slicer	0.0.8
smart-open	7.1.0	smart_open	7.1.0
snappy	1.2.1	sniffio	1.3.1
sortedcontainers	2.4.0	soupsieve	2.5
spacy	3.8.2	spacy-legacy	3.0.12
spacy-loggers	1.0.5	spdlog	1.14.1
sqlite	3.47.2	srsly	2.5.1
stack_data	0.6.3	stanza	1.10.1
statsmodels	0.14.4	streamz	0.6.4
svt-av1	2.3.0	sympy	1.13.3
tbb	2021.13.0	tblib	3.0.0
tensorboard	2.17.1	tensorboard-data-server	0.7.0
tensorflow	2.17.0	tensorflow-base	2.17.0
tensorflow-estimator	2.17.0	termcolor	2.5.0
terminado	0.18.1	tf-keras	2.17.0
thinc	8.3.2	threadpoolctl	3.5.0
tifffile	2024.12.12	tinycss2	1.4.0
tk	8.6.13	tokenizers	0.21.0
tomli	2.2.1	toolz	1.0.0
tornado	6.4.2	tqdm	4.67.1
traitlets	5.14.3	transformers	4.48.0
transformers-interpret	0.10.0	treelite	4.3.0
typer	0.15.2	typer-slim	0.15.2
typer-slim-standard	0.15.2	types-python-dateutil	2.9.0.20241206
typing-extensions	4.12.2	typing_extensions	4.12.2
typing_utils	0.1.0	tzdata	2024b
uc-micro-py	1.0.3	ucx	1.17.0
ucx-proc	1.0.0	ucx-py	0.41.00
ucxx	0.41.00	ujson	5.10.0
unicodedata2	16.0.0	uri-template	1.3.0
uriparser	0.9.8	urllib3	2.3.0
wasabi	1.1.3	wayland	1.23.1
wcwidth	0.2.13	weasel	0.4.1
webcolors	24.11.1	webencodings	0.5.1
websocket-client	1.8.0	werkzeug	3.1.3
wheel	0.45.1	widgetsnbextension	4.0.13
wordcloud	1.9.4	wrapt	1.17.2

Continued on next page

Table A.1 – continued from previous page

Package	Version	Package	Version
x265	3.5	xarray	2025.1.1
xerces-c	3.2.5	xgboost	2.1.2
xkeyboard-config	2.43	xorg-libice	1.1.2
xorg-libsm	1.2.5	xorg-libx11	1.8.11
xorg-libxau	1.0.12	xorg-libxcomposite	0.4.6
xorg-libxcursor	1.2.3	xorg-libxdamage	1.1.6
xorg-libxdmcp	1.1.5	xorg-libxext	1.3.6
xorg-libxfixes	6.0.1	xorg-libxi	1.8.2
xorg-libxinerama	1.1.5	xorg-libxrandr	1.5.4
xorg-libxrender	0.9.12	xorg-libxtst	1.2.5
xxhash	0.8.2	xyzservices	2024.9.0
xz	5.6.3	xz-gpl-tools	5.6.3
xz-tools	5.6.3	yaml	0.2.5
yarl	1.18.3	zeromq	4.3.5
zfp	1.0.1	zict	3.0.0
zipp	3.21.0	zlib	1.3.1
zlib-ng	2.2.3	zstandard	0.23.0
zstd	1.5.6		

Appendix B

Supplementary Figures

In this appendix, additional figures that support the analyses and results presented in the main text are provided. These include extended visualizations, distribution plots, and other relevant graphical information referenced throughout the thesis.

B.1 Dataset Descriptive Statistics

The figures of this section are meant to extend and complement the description of the dataset done in §4.1.1.2.

Figure B.1 shows the distribution of the number of characters per sample across splits and target classes. Figure B.2 presents the distribution of word lengths. Figure B.3 shows the distribution of the type-token ratio (TTR), defined as the total number of unique words divided by the total number of words for each sample. Figure B.4 presents the distribution of the number of emojis per sample. Finally, figure B.5 shows the distribution of the number of special characters (non-alphanumeric).

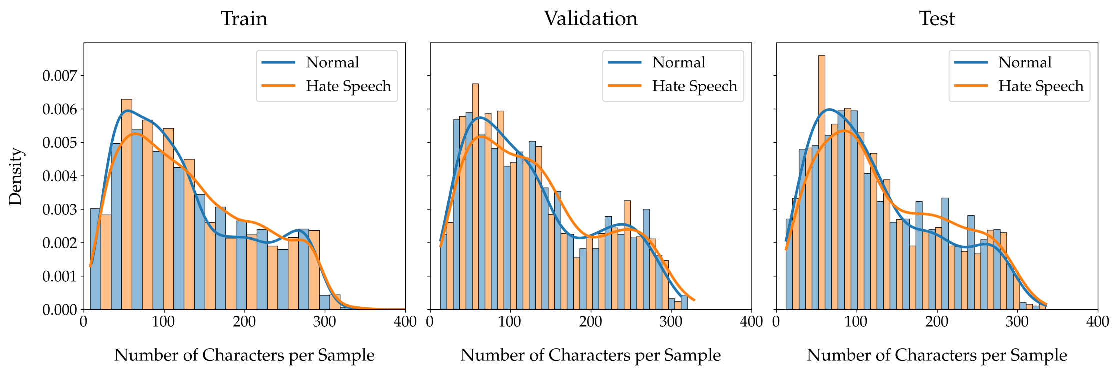

FIGURE B.1: Distribution of the number of characters per sample for each dataset split and target class. A kernel density estimation (KDE) curve is overlaid facilitate interpretation.

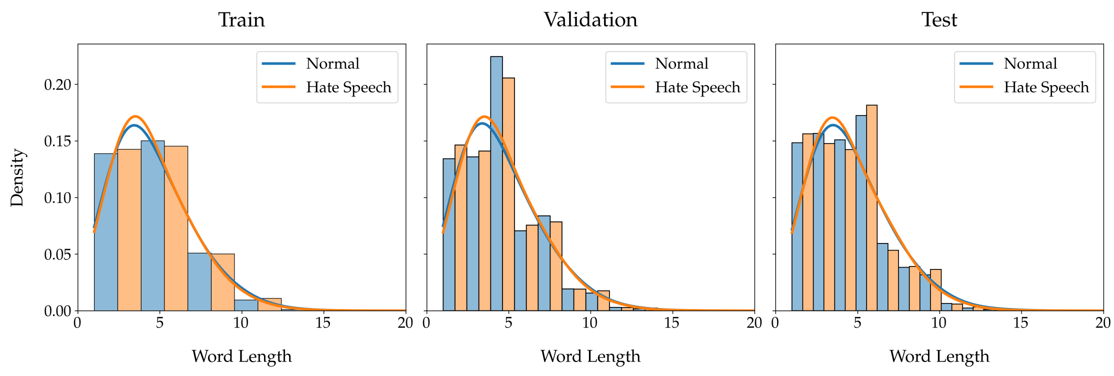

FIGURE B.2: Distribution of the length of words per sample for each dataset split and target class. A kernel density estimation (KDE) curve is overlaid facilitate interpretation.

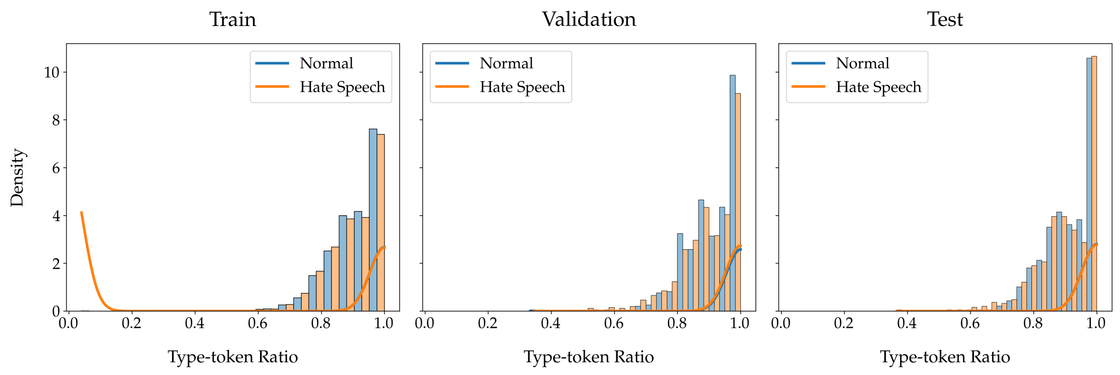

FIGURE B.3: Distribution of the type-token ratio of the samples for each dataset split and target class. A kernel density estimation (KDE) curve is overlaid facilitate interpretation.

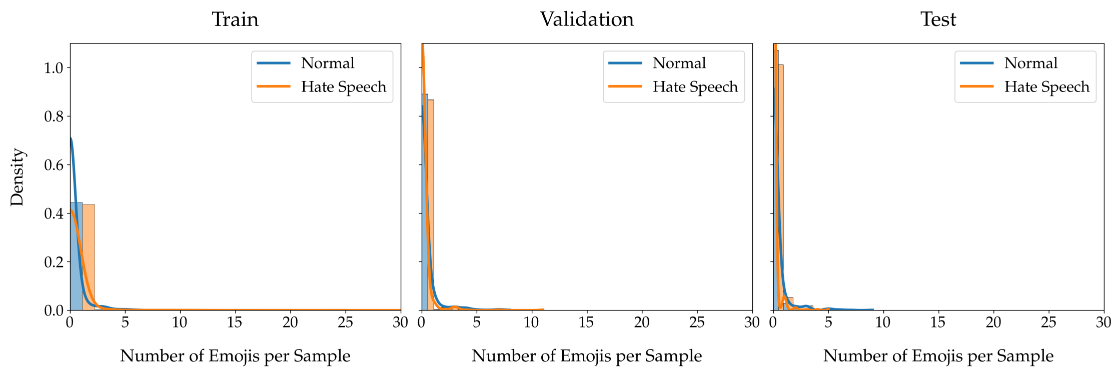

FIGURE B.4: Distribution of the number of emojis per sample for each dataset split and target class. A kernel density estimation (KDE) curve is overlaid facilitate interpretation.

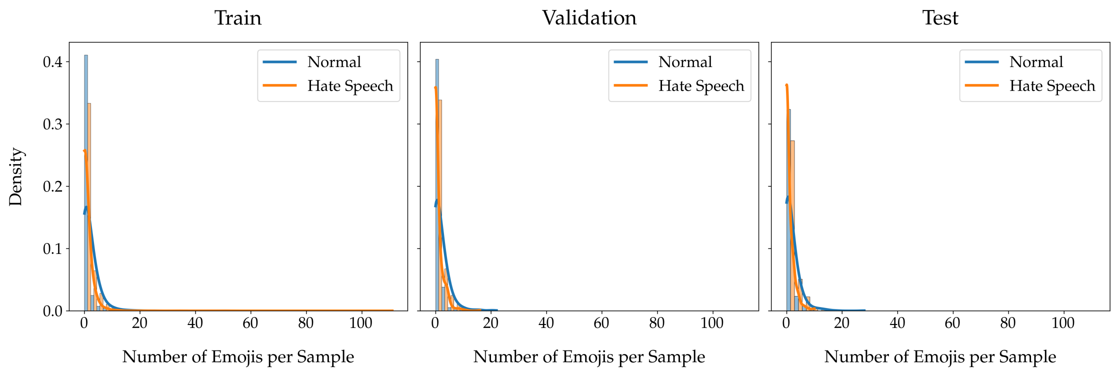

FIGURE B.5: Distribution of the number of special characters (non-alphanumeric) per sample for each dataset split and target class. A kernel density estimation (KDE) curve is overlaid facilitate interpretation.

B.2 Model Training Curves

In this section, the training curves for the BERT and XLNet models are reported. For each model, the evolution of the loss function and the F_1-score on the validation dataset are represented. Figure B.6 shows the data for BERT, and figure B.7 does it for XLNet.

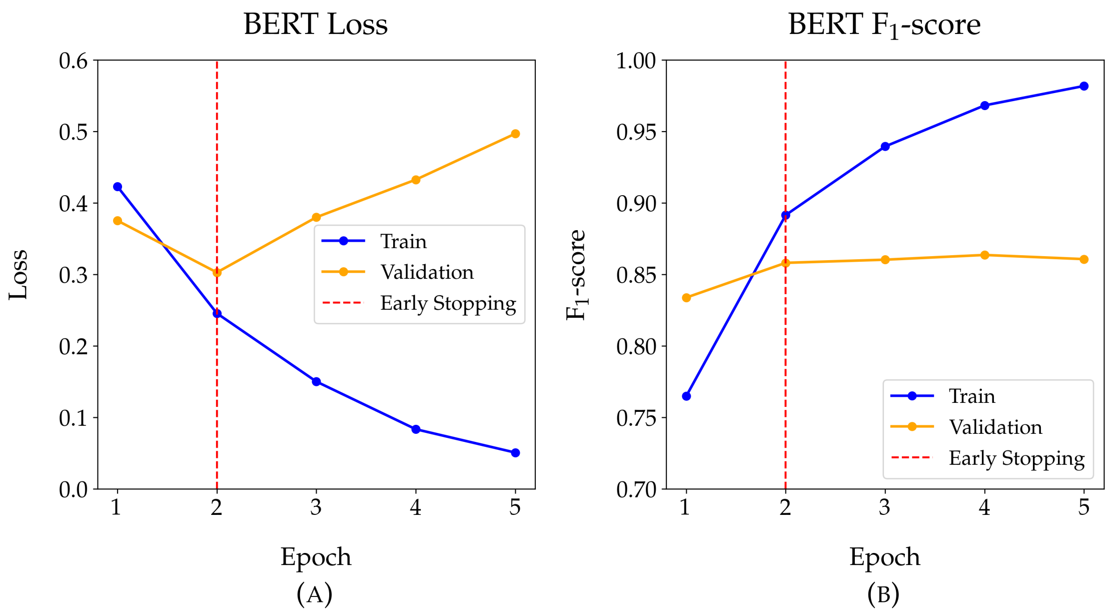

FIGURE B.6: (A) Evolution of BERT loss during the training epochs. (B) Evolution of BERT F_1-score during the training epochs.

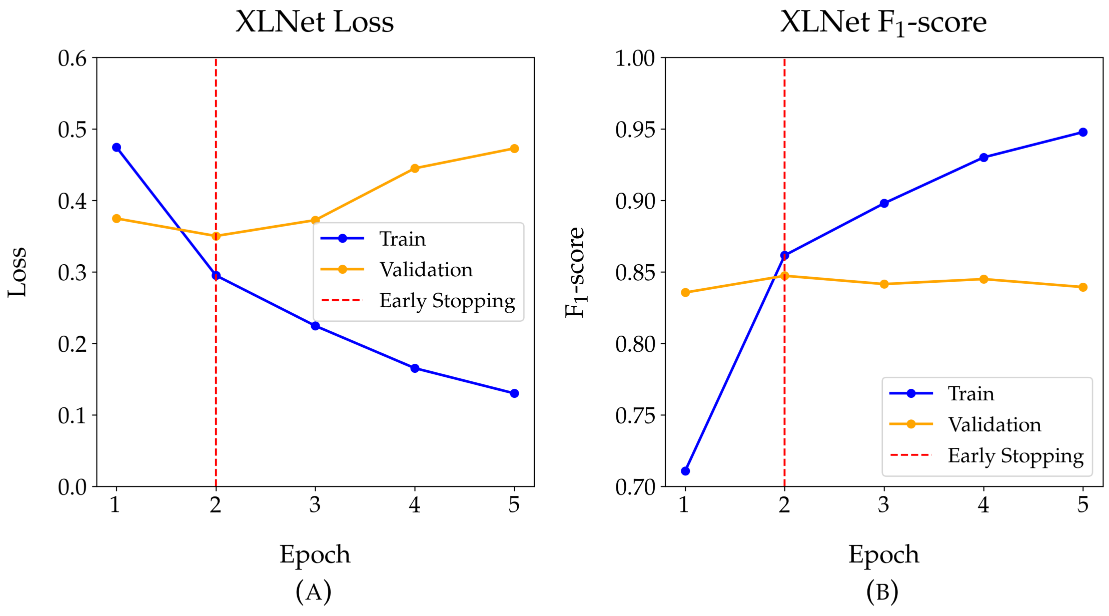

FIGURE B.7: (A) Evolution of XLNet loss during the training epochs. (B) Evolution of XLNet F_1-score during the training epochs.

B.3 Hierarchical Explainability Trees

In this section, additional hierarchical explainability trees generated by IDG are shown, to complement the results of the qualitative analysis presented in §4.2.2.2.

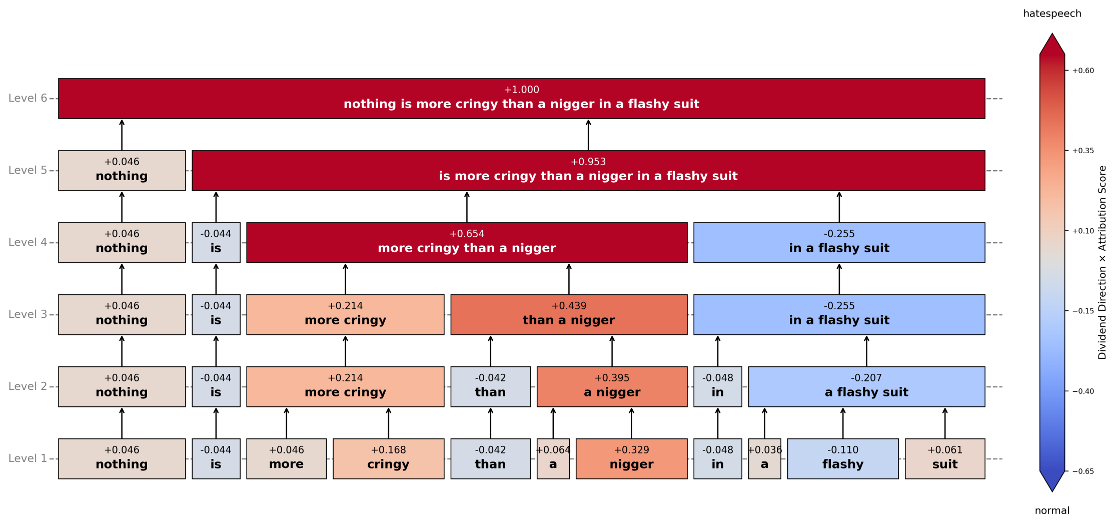

FIGURE B.8: Explainability tree for the sequence with ID 13730242_gab. The model correctly predicts the hate speech class, with a 94% confidence.

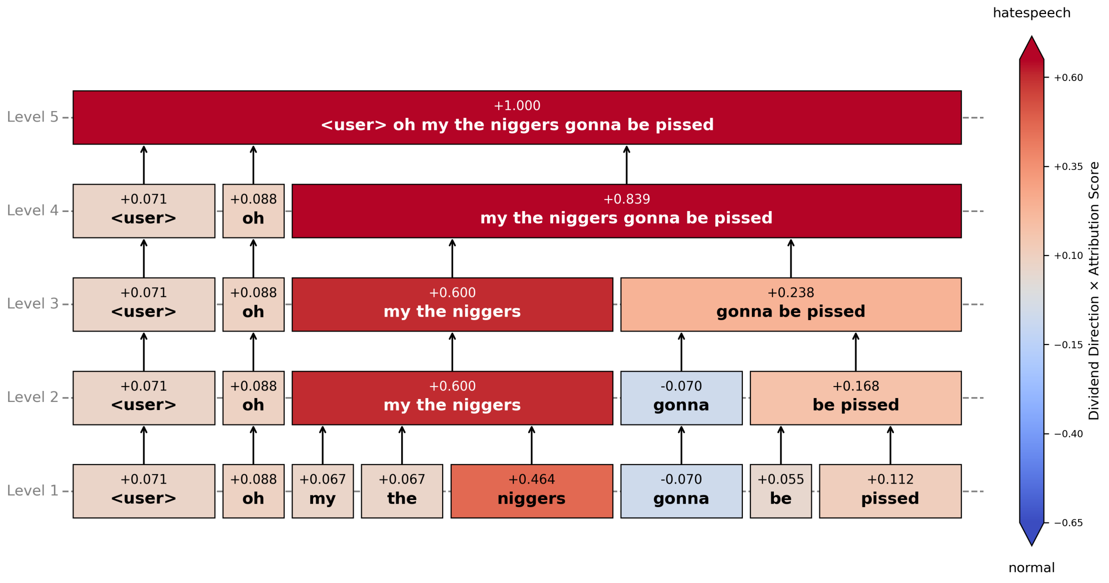

FIGURE B.9: Explainability tree for the sequence with ID 1178934864684470273_-twitter. The model correctly predicts the hate speech class, with 85% confidence.

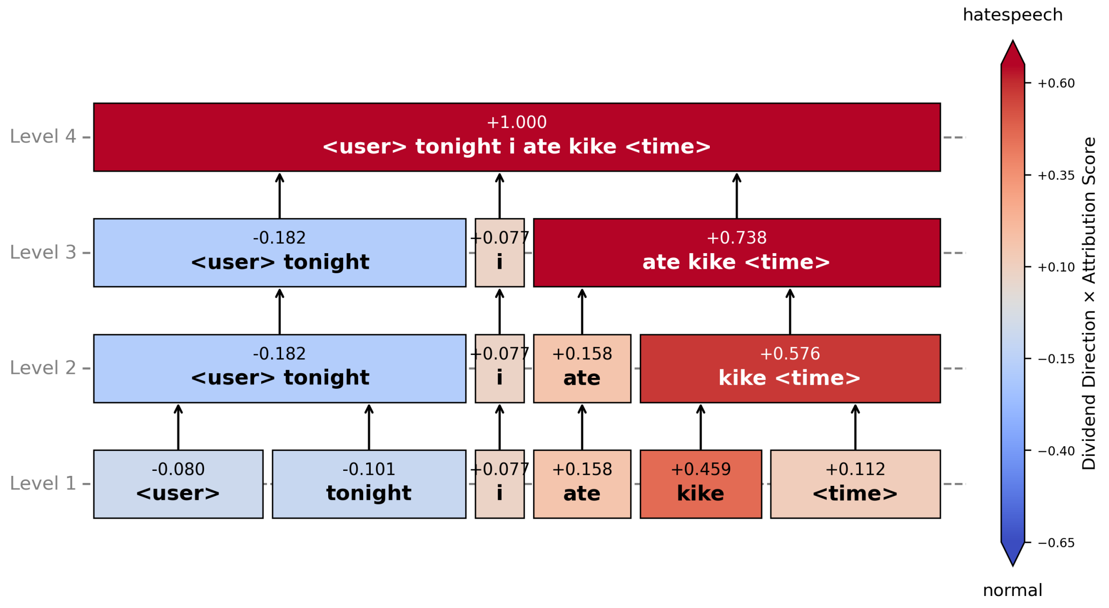

FIGURE B.10: Explainability tree for the sequence with ID 1266939334772523009_-twitter. The model correctly predicts the hate speech class, with 99% confidence.

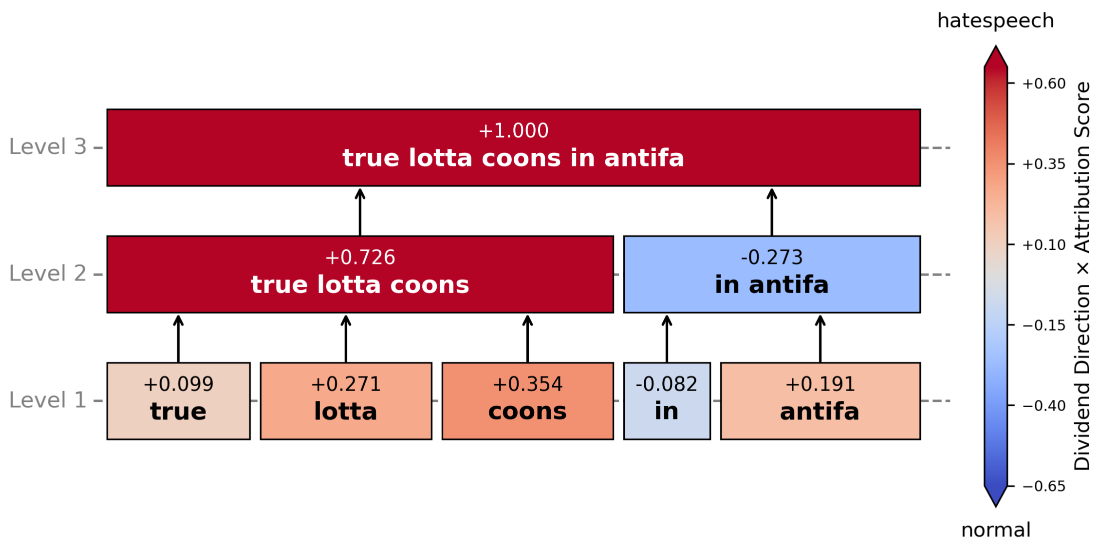

FIGURE B.11: Explainability tree for the sequence with ID 13925301_gab. The model correctly predicts the hate speech class, with 98% confidence.

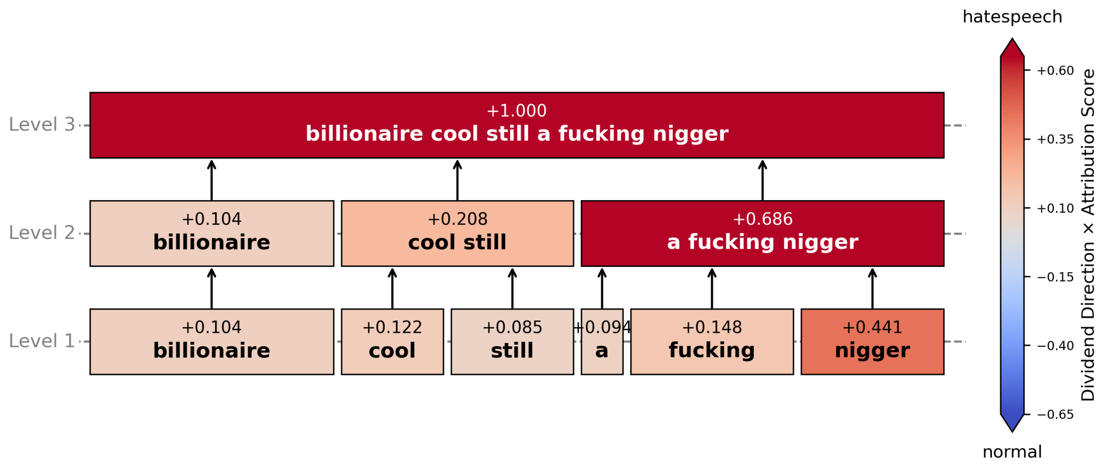

FIGURE B.12: Explainability tree for the sequence with ID 17593622_gab. The model correctly predicts the hate speech class, with 98% confidence.

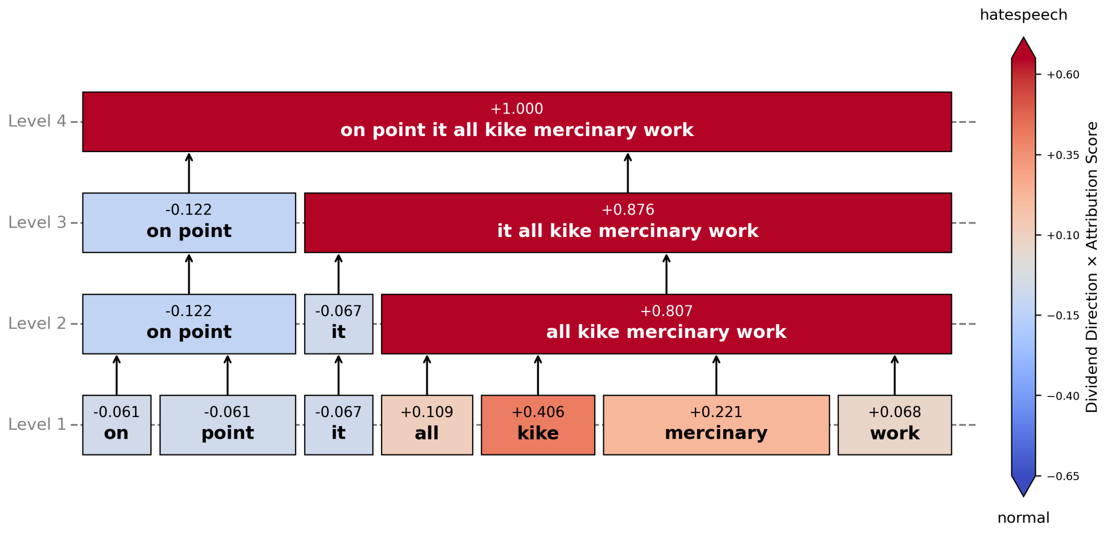

FIGURE B.13: Explainability tree for the sequence with ID 22398230_gab. The model correctly predicts the hate speech class, with 99% confidence.

Appendix C

Code Availability

The code developed and used throughout this thesis is publicly available in the following GitHub repository: `github.com/srmarcballestero/IDG_HateXplain`.