UNIVERSITAT DE BARCELONA

Treball final de grau

GRAU D'ENGINYERIA INFORMÀTICA

Facultat de Matemàtiques i Informàtica
Universitat de Barcelona

SELF-QUESTIONNAIRE: AUTOMATIC GENERATION OF QUESTIONS USING ARTIFICIAL INTELLIGENCE TECHNIQUES FOR SELF-REGULATED LEARNING

Autor: Lluís Roca Román

Tutors: Eloi Puertas Prats i Daniel Ortiz Martínez

Realitzat a: Departament de Matemàtiques i Informàtica

Barcelona, 4 de juliol de 2023

Resum

Durant aquests últims anys s'ha vist com el processament de llenguatge natural ha progressat de manera considerable, i cada cop el tenim més present en el nostre dia a dia, encara que no siguem totalment conscients. En l'àmbit educatiu ja se li està donant ús, com per exemple en el control de plagi.

En aquest treball s'implementarà una eina de Natural Language Processing d'avaluació, amb l'objectiu que l'alumnat pugui autoavaluar-se de manera independent, amb un generador de preguntes i les seves corresponents respostes.

Resumen

Durante estos últimos años se ha visto cómo el procesamiento de lenguaje natural ha progresado de forma considerable, y cada vez lo tenemos más presente en nuestro día a día, aunque no seamos totalmente conscientes. En el ámbito educativo ya se le está dando uso, como por ejemplo en el control de plagio.

En este trabajo se implementará una herramienta de Natural Language Processing de evaluación, con el objetivo de que el alumnado pueda autoevaluarse de forma independiente, con un generador de preguntas y sus correspondientes respuestas.

Abstract

In recent years, natural language processing has made considerable progress, and it is increasingly present in our daily lives, even if we are not fully aware of it. In the educational field it is already being used, for example in plagiarism control.

In this work we will implement a Natural Language Processing evaluation tool, so that students can independently self-evaluate themselves, with a generator of questions and their corresponding answers.

Agraïments

Voldria agrair a l'Eloi Puertas i al Daniel Ortiz pel seu constant seguiment del projecte i la seva ajuda.

Índex

1 Introducció

Per tal que sigui entesa i ubicada la problemàtica que vol ser resolta, en primer lloc, serà exposada una contextualització de tot el problema, on hi seran introduïts tots els conceptes teòrics relacionats amb aquest, juntament amb les línies de recerca més rellevants que han estat seguides per a la seva solució. Posteriorment, seran indicades quines han estat les justificacions de la investigació que ha estat realitzada i, per acabar, hi seran enumerats els objectius del present treball i seran exposades les motivacions per a la realització d'aquest.

1.1 Contextualització

Avui en dia, la intel · ligència artificial està cada cop més present en el nostre dia a dia, sent utilitzada per un ampli públic, ja sigui pel processament d'imatges, vehicles amb autonomia, detecció de malalties, etc. Però també es veu molt sovint a l'hora de processar textos, en usar motors de cerca, en fer servir el traductor, els assistents virtuals, etc. Aquestes intel · ligències, d'alguna manera, aconsegueixen entendre la semàntica d'un text i donar una sortida, en funció de la tasca especifica assignada.

Un dels llocs on aquestes aplicacions de processament de text amb intel · ligència artificial han estat àmpliament usades és l'àmbit acadèmic. És aquí on s'han trobat trobades una gran quantitat d'aplicacions per aquestes eines automatitzades, des de la generació de treballs o exàmens fins a la detecció de frau de treballs de l'alumnat. Per tant, fora convenient considerar la implementació d'unes eines, també amb base a la intel · ligència artificial, més completes i sofisticades per a la realització de tasques més complexes dins d'aquest àmbit, que estarien relacionades amb l'avaluació d'alumnes, tant perquè l'alumne pugui autoavaluar-se com perquè el professorat pugui avaluar a l'alumne.

Dins d'aquest context introduirem el sistema que serà implementat per al present treball. D'una forma genèrica i abstracte, a aquest sistema li serà introduït un text, com a *input* i, amb aquest text serà entrenat un model de generació de text, amb el qual seran obtingudes diferents definicions de termes del text. Aquest mateix text d'entrada també li serà subministrat a un model, a partir del qual serà extreta una llista de conceptes claus i els relacionarà en diferents grups, sent aconseguida finalment la definició de tots aquests conceptes similars de cada grup. Finalment, es jugarà amb aquestes definicions per generar les preguntes per a l'estudiant

A manera de visualització de la topologia global d'aquest sistema, és recomanada la visualització de la Figura 14.

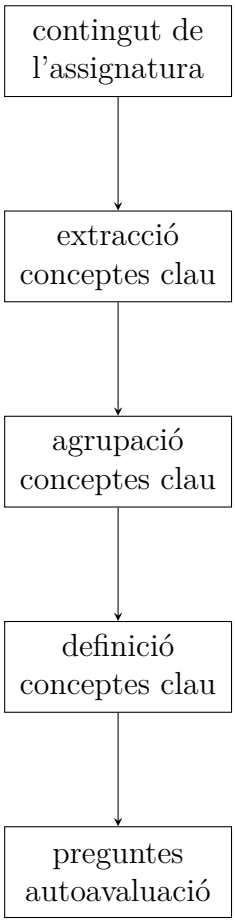

Figura 1: Esbós de l'estructura, respecte a blocs, del sistema on puguin ser obtingudes les definicions d'un concepte i múltiples conceptes claus relacionats.

1.2 Objectius i motivació

Vista la problemàtica d'obtenció simultània de definicions de múltiples conceptes, juntament amb la seva solució, que han estat exposats prèviament a la secció 1.1, per a aquest treball han estat establerts els següents objectius:

- **Entrenar un model de generació de text amb contingut de una assignatura**
- **Implementar un sistema a partir del qual puguem obtenir una sèrie de preguntes amb les quals l'estudiant pugui autoavaluar-se**

- **Entendre com operen els models nlp, i avaluar la manera de fer un d'aquests**

En últim lloc, pel que fa a les motivacions per a la realització del present treball, destaquem les següents:

- Gran interès en tot el funcionament, tan algorísmica com arquitectònicament, de sistemes d'intel · ligència artificial per al processament del llenguatge natural.
- Percepció que l'eina desenvolupada solucionarà bastants problemes que podrien facilitar la feina tant de l'estudiant com del professorat en l'avaluació.
- Esperança que els resultats d'aquest treball puguin ser utilitzats en un futur per tothom que estigui interessat en aquesta eina.

1.3 Estructura del document

Per tal que sigui aclarida l'estructura amb la qual ha estat escrit el present document, a continuació són enumerades totes les seves parts principals amb una explicació del contingut de cadascuna d'elles.

1. **Introducció**: Per tal que sigui entès el problema a resoldre en el present treball, es farà una introducció al problema i s'exposarà la naturalesa de la solució a implementar. Posteriorment, presentarem les principals motivacions per a la realització del treball, i, en últim lloc, enumerarem els objectius que han estat proposats.
2. **Estat de l'art**: Abans de començar el treball, han estat consultats una sèrie d'estudis i publicacions anteriors de temàtiques similars i en aquesta secció serà realitzada una enumeració d'aquelles que han estat de major rellevància.
3. **Anàlisi del problema**: Havent acabat la investigació prèvia de l'últim nivell de desenvolupament de les tecnologies a ser utilitzades en aquest treball, aquí s'explicarà el problema i la proposta feta, arquitectònicament parlant, juntament amb les metodologies i conceptes teòrics associats que han estat empleats. Finalment, tenint ja els materials i mètodes a ser usats, aquí s'exposarà el procés d'implementació de totes les funcionalitats a incloure dins del sistema, fent èmfasi en alguns segments d'algunes funcionalitats importants, mostrant ocasionalment el seu codi font.
4. **Resultats i discussió**: Un cop conclosa la implementació del sistema, aquest

és provat i, és aquí on seran documentades les proves a les quals aquest ha estat sotmès, juntament amb els resultats i conclusions extretes. A més també s'inclouran possibles dissertacions sobre el compliment de la hipòtesi inicial.

2 Estat de l'art

Per tal que la posterior implementació pugui ser enfocada en la direcció correcta des d'un primer moment, prèviament a l'inici d'aquesta serà realitzada una investigació d'un subconjunt de treballs de temàtiques similars al d'aquest treball. A partir de cada anàlisi d'un treball anterior s'articularan unes succintes conclusions, a partir de les quals puguin ser articulades les subseqüents recerques a més treballs.

Aquesta naturalesa incremental i progressiva de la redacció de l'estat de l'art ha estat ideada en compatibilitat amb la metodologia de creació del sistema d'intel · ligència artificial que ha estat implementat. Més concretament, el disseny del sistema ha estat concebut com una juxtaposició de diferents grans blocs abstractes i, a partir de les conclusions que aquí han estat extretes, aquest mateix disseny serà polit i especificat amb més detall.

2.1 Introducció

Sempre ha estat una qüestió a resoldre si la computació podria emular el funcionament d'un cervell humà. Dintre de la intel · ligència artificial, el Natural Language Procesing (NLP), és una àrea que estudia com una màquina pot entendre el llenguatge humà, interpretar-ho i el fet de processar-ho. Aquest camp l'hem vist en diferents llocs de la nostra vida quotidiana, ja sigui per la traducció de textos, els charbots, la recuperació d'informació,etc.

Els models de llenguatge neuronal tenen com a objectiu predir la següent paraula donat un context. Aquests diuen que el llenguatge consisteix en seqüències de simbols atòmics (paraules) que formen frases, i que al final de la frase l'últim símbol fa un important rol. Fins que no es va intentar aplicar intel · ligència artificial, va haver-hi molts intents fent servir models estadístics molt específics per llenguatges concrets. Si tenim en compte la millora en la capacitat de predir dades seqüencials, veurem que han millorat considerablement amb aquests models. Si, en canvi, ho tenim en compte amb la seva aplicació pràctica sí que s'ha de ser molt més realista, ja que en casos pràctics com ara traducció, reconeixement de veu, etc. els models es fan a partir de grans quantitats de dades, i són molt més simples que els que venen de la recerca, pel fet que els models procedents de la investigació solen ser complexos i sovint funcionen bé només per a sistemes basats en quantitats de dades d'entrenament molt limitades.

2.2 Models basats en Recurrent Neural Networks

Primer es va proposar utilitzar xarxes neuronals feedforward. Es basen en un conjunt de nodes connectats anomenats neurones artificials, en les que cadascuna d'aquestes neurones transmet un senyal a les altres neurones, i aquestes la processen. En aquestes

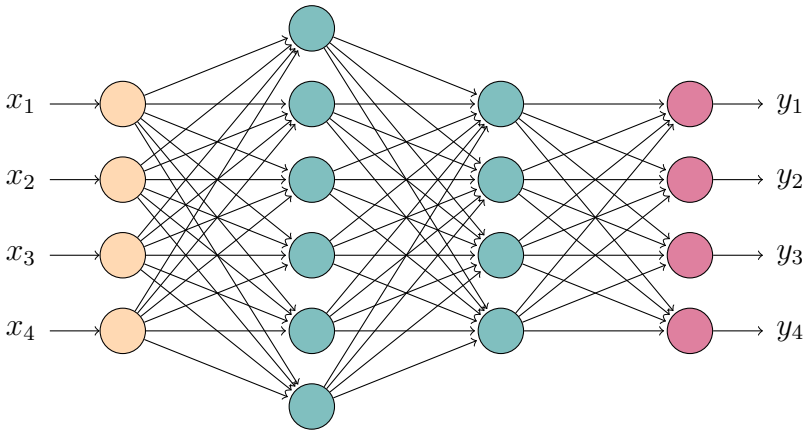

Figura 2: Esquema d'una xarxa neuronal feed forward amb dues capes

xarxes es poden diferenciar tres parts:

- La capa d'entrada, l'encarregada de processar les dades d'entrada.
- La capa o les capes ocultes, formada per neurones, amb una funció de predicció encarregada de transformar l'input que rebi de la capa d'entrada en un output per la capa de sortida.
- La capa de sortida, representarà l'output de la xarxa neuronal.

El funcionament de la xarxa neuronal, pot ser definit somerament de la següent forma:

1. A la capa d'entrada li són subministrades les dades inicials del problema.
2. Les dades d'entrada són subministrades a un *pipeline* de capes ocultes, on, a la primera d'elles li són subministrades les dades de la capa d'entrada, mentre que, a les subseqüents capes del *pipeline* li són subministrades les dades de sortida de l'anterior capa oculta.
3. Finalment output de l'última capa oculta serà el valor que retornarà la capa de sortida

Les xarxes feedforward sempre avancen en la mateixa direcció, són unidireccionals.

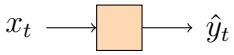

Figura 3:

Aquests models donen uns resultats exitosos, millors que barrejant diferents models estadístics, però l'inconvenient més gran és que abans de l'entrenament s'ha de fixar una longitud fix del context, generalment només té en compte de 5 a 10 de les últimes paraules per predir la següent.

A partir de la idea que a través d'un model d'alimentació intentem obtenir un output, per resoldre el problema anterior, Mikolov *et al.* 2010 [1] va proposar una millora a través de RNN (recurrent natural network).

L'objectiu de les xarxes neuronals recurrents és retroalimentar el model d'alimentació en diferents moments o "timesteps". Això es fa repetint les operacions de la capa oculta de manera recurrent, de manera que les sortides de la capa oculta en un moment es converteixen en entrades per a la següent iteració. Aquesta retroalimentació permet a la xarxa neuronal retenir informació sobre les entrades passades i utilitzar-la per prendre decisions en el futur. És guarda la informació de les anteriors capes, aprofitant que aquest presenta un ordre temporal. D'aquesta manera es manté informació de les dades que van entrant i el seu ordre, donant més importància a les últimes entrades que a les primeres. 4.

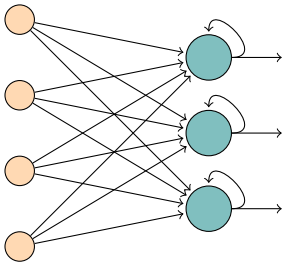

Figura 4: Esquema RNN.

En aquestes no es limita la mida del context, es fan servir connexions recurrents, que, per tant, la informació circula contínuament dins de les xarxes neuronals. Gràcies a això obtenim una predicció molt més fiable a la sortida de la capa oculta, és per això que es diu que és un model que aprèn a predir el resultat d'una capa.

El funcionament d'una RNN simple com la que s'ha comentat anteriorment és el següent. Tenim x que serà l'input, s que serà del context (capa oculta), i y que serà l'output. El

vector input x(t) es forma concatenant el vector w que representa la paraula actual, i l'output de les neurones al context s al temps t-1.. 1

$$x(t) = w(t) + s(t-1) \tag{1}$$

$$s_j(t) = f\left(\sum_i x_i(t) u_{ji}\right) \tag{2}$$

$$s_j(t) = f\left(\sum_i x_i(t) u_{ji}\right) \tag{3}$$

on $f(z)$ es una funció sigmoid d'activació 4

$$f(z) = \frac{1}{1+e^{-z}} \tag{4}$$

i $g(z)$ es la funció softmax 5

$$g(z_m) = \frac{e^{z_m}}{\sum_k e^{z_k}} \tag{5}$$

Per inicialitzar s(0) s'utilitza un vector de valors petits, com 0.1, i quan processi una gran quantitat de dades, la inicialització no tindrà un paper important, serà durant el procés d'entrenament que se li donarà un valor. La mesura del vector x és igual a mida del vocabulari x la mida de la capa de context.

Per entrenar la xarxa s'utilitza l'algoritme de retropropagació estàndard amb descens de gradient estocàstic. S'entrena en diferents etapes, en què totes les dades es presenten seqüencialment. Els pesos estan inicialitzats amb valors molt petits i a mesura que es va propagant usarem un algoritme de propagació amb descens de gradient estocàstic.

Després de cada etapa, la xarxa es testeja amb dades de validació. Si aquesta similitud augmenta, l'entrenament segueix a una nova etapa, si no hi ha millora significant, la taxa d'aprenentatge disminueix a la meitat. Si continua sense haver-hi millora, l'entrenament és pausa. La sortida es representarà com la distribució de probabilitat de la següent paraula donada una paraula anterior w(t) i un context s(t – 1).

Per optimitzar, el que es va proposar va ser ajuntar totes les paraules que apareixen més poc que un llindar, i processar les probabilitats com es veu a la Formúla 6,on Crare

és el nombre de paraules en el vocabulari que apareixem amb menys freqüència que el llindar.

$$P(w_i(t+1)|w(t), s(t-1)) = \begin{cases} \frac{y_{rare}(t)}{C_{rare}} & \text{if } w_i(t+1) \text{ is rare} \\ y_i(t) & \text{otherwise} \end{cases} \tag{6}$$

2.3 Encoder i Decoder

Més tard van aparèixer els encoders i decoders. El primer cop que van ser vistos va ser en el treball de Bahdanau *et al.* 2014 [2].

El seu objectiu va ser entrenar un model amb finalitat de traduir. Ajustar un model d'entrenament per maximitzar la probabilitat de dues frases paral · lela utilitzant un corpus d'entrenament. Un cop la distribució condicional és apresa pel model de traducció, donada una oració, la traducció corresponent es genera buscant la frase que maximitzi la probabilitat condicional.

Aquest model es basa en la concatenació de dos models diferents, l'encoder i el decoder.

- Encoder: El que fa l'encoder és llegir una frase com a input, de manera recurrent en cel · les (ja sigui GRU o LSTM), en què a cada iteració l'input serà un únic element de la frase. En una frase en el temps t, el que farà serà calcular l'estat ocult al temps t, h^t. Es va emmagatzemant tots els estats ocults en un vector c, generat per la seqüència d'estats ocults, ja que així ajudem al decoder a ser més precís.

$$h_t = f(x_t, ht-1) \tag{7}$$

$$c = q(\{h_1, ..., h_{T_x}\}) \tag{8}$$

  Com podeu veure en la fórmula 7 i 8 són funcions no lineals.

- Decoder: El que fa el decoder és predir l'output y^t en el timstep t. Donat el vector c de context i totes les prediccions anteriors y, es defineix una probabilitat sobre la traducció y descomponent la probabilitat conjunta en els condicionals ordenats.

$$p(y) = \prod_{t=1}^{T} p(y_t|\{y_1, ..., y_{t-1}\}, c) \tag{9}$$

$$p(y_t|\{y_1, ..., y_{t-1}\}, c) = g(y_{t-1}, s_t, c) \tag{10}$$

En resum, l'encoder i el decoder treballen conjuntament per transformar una entrada en una sortida en una tasca de processament de llenguatge natural. L'encoder serà qui agafi la seqüència d'entrada i la transformi en una representació d'aquesta, i el decoder utilitza aquesta representació per generar una seqüència de sortida, tal com es veu a la Figura 5.

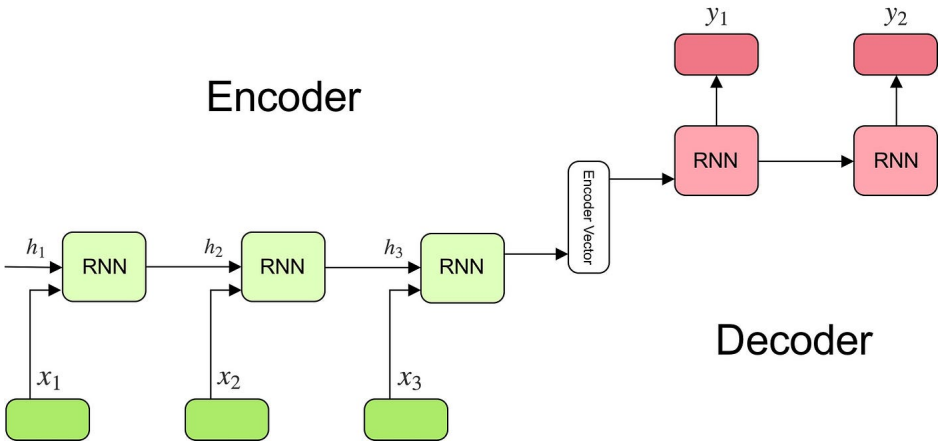

Figura 5: Esquema de la topologia, a nivell de blocs, d'un sistema format per un *encoder* i un *decoder*.

El problema de l'encoder que s'ha vist anteriorment és que llegeix paraules x en el temps t en una seqüència de x1 a xt, per tant, cada paraula té en compte el que té davant, però no el que té enrrere. Una nova implementació és un nou model que segueix l'arquitectura encoder decoder, però la diferència és que cada paraula no només resumeix les paraules anteriors, també les paraules posteriors. L'objectiu és llegir la seqüència de x1 a xt, i calcular la seqüència d'estats ocults, de h1 a ht, i fer el mateix en direcció contrària, llegir una seqüència a l'ordre revers, de xt a x1, i després calcular la cadena d'estats ocults. Per cada paraula, tindrem un estat ocult que va cap endavant i un estat ocult cap enrere. L'estructura que s'ha descrit anteriorment es pot visualitzar a la Figura 6.

Amb aquest model solucionem el problema del context de les paraules, que l'agafem tot, però, així i tot, ens trobem amb un altre problema, tot i implementar un model

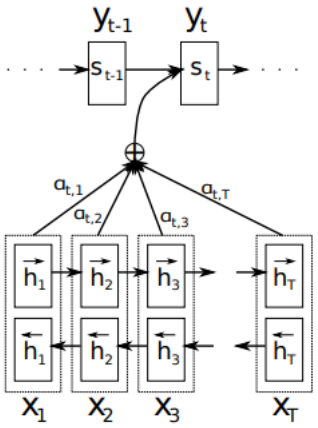

Figura 6: Esquema de la topologia, a nivell de blocs, d'un *encoder* bidireccional.

bidireccional, els models que hem vist no tenen memòria a llarg termini, ja que no té prou capacitat per retenir informació. Amb grans seqüències de dades apareix el problema que el gradient arriba un punt que és insignificant. L'altre problema és que les xarxes neuronals recurrents funcionen de manera seqüencial, per tant, el fet de processar les dades és un procés molt lent, pel fet que s'ha de fer seqüencialment.

2.4 Transformer

En el treball de Vaswani *et al*, 2017 [3] s'introdueix el concepte de transformer, que consisteix en la implementació de l'encoder i decoder. En els models presentats anteriorment sorgien dos grans problemes; en primer lloc, la memòria a llarg termini, i, en segon lloc, el de la naturalesa seqüencial que tenen les xarxes neuronals recurrents. Aquest últim no ens permet paral · lelitzar el model, per la qual cosa en tractar grans quantitats de dades resulta en un alentiment el procés.

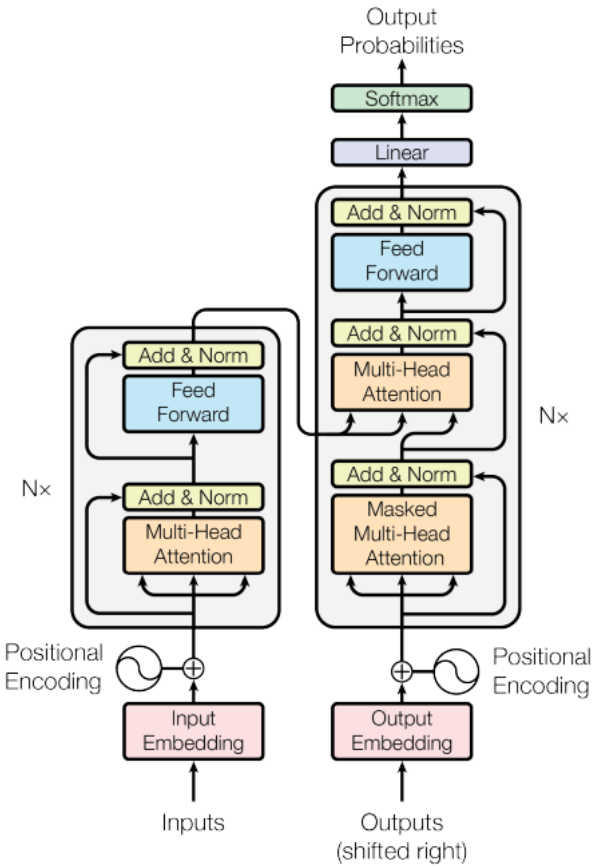

Figura 7: Caption

El transformer 7 és una xarxa neuronal que utilitza l'arquitectura d' encoder i decoder, amb l'objectiu de solucionar el problema de la velocitat i la memòria sense utilitzar xarxes neuronals recurrents. Es concatenen tots els inputs i es tracten alhora.

El procés comença amb un embbeding, una representació de les paraules prèvia als transformers. Consisteix a agrupar les paraules de forma automàtica en un espai multidimensional de n dimensions, on n és el nombre de paraules. Com més s'assemblen

els conceptes, més a prop estaran els vectors corresponents.

Un cop s'ha fet l'embedding, s'ha d'afegir d'alguna manera la posició en la qual es troba aquesta paraula. L'ordre de les paraules és important, no és el mateix.

Tanca la porta amb clau $\neq$ *La porta tanca amb clau*

Per solucionar aquest problema s'afegeix un valor addicional que ens permeti interpretar la posició en la qual es troba. En les frases anteriors, la paraula porta tindrà el mateix token, però amb un valor addicional que ens descrigui la posició en la qual es troba.

Un cop fet això, la nostre paraula ja podrà passar per l'encoder.

- ENCODER: Està format per 6 capes idèntiques, i cada capa està composta per dues parts. La primera serà un component d'atenció, i la segona serà una xarxa neuronal. L'output de cada subcapa, serà la normalització de la sortida, aplicant una connexió residual.
- DECODER: També està format per 6 capes idèntiques, molt semblant al decoder, però en lloc de tenir dues parts tindrà tres parts. La primera serà un mecanisme d'atenció en què l'input serà l'output en t-1, una posició a l'esquerra cada element. Amb això ens assegurem que l'element en la posició només sigui afectat pels elements anteriors a ell. Seguidament hi haurà un altre mecanisme d'atenció sobre l'output de l'encoder. Finalment l'última part serà una xarxa feedforward. Cadascuna de les parts, igual que a l'encoder, s'aplica una connexió residual i és normalitzem

La capa d'atenció 8 té com a objectiu relacionar les paraules d'una oració, amb la seva importància. Si tenim la frase

"El Carlos va de camí a casa amb uns pantalons blaus"

ha de saber que qui té uns pantalons blaus és el Joan i no la casa, per tant, ha d'entendre que la penúltima paraula afecte la segona.

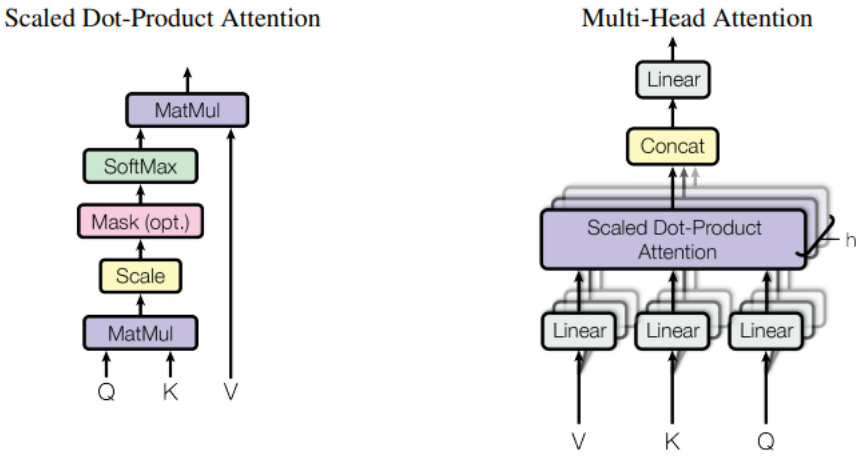

Figura 8: Esquema capa d'atenció

Primer s'extreu una query, un valor i una clau, que són transformacions de cada vector d'entrada. Les transformacions seran aquestes:

- Query = I x W(Q)
- Key = I x W(K)
- Value = I x W(V)

on I és el vector input, i W(Q), W(K) i W(V) són les matrius corresponents per transformar el vector. I en el vector Query, Key i Value. Un cop tenim aquests tres vectors aplicarem la fórmula 11

$$\text{Attention } (Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{11}$$

on l'output serà el vector d'atenció, que representa l'importància relativa de l'input.

2.5 BERT i GPT2

Actualment, els models de llenguatge més importants són el BERT com encoder i el GPT2 com a decoder.

Bert és un model de llenguatge encoder que va ser descrit l'any 2019 [4], introduït per Google. Aquest model és un codificador basant en l'arquitectura Transformer, explicada anteriorment.

Una de les característiques més importants d'aquest model és que permet fer diverses tasques de processament de llenguatge natural.

Aquest model es basa en dos passos, per una banda, es fa un pre-entrenament genèric amb dades no etiquetades sobre diferents tasques del pre-entrenamen, i un cop s'ha fet aquest, per obtenir una major precisió d'una tasca especifica, es realitza un fine tunning a aquest model preentrenat, utilitzant dades supervisades, com es pot veure en la Figura 9.

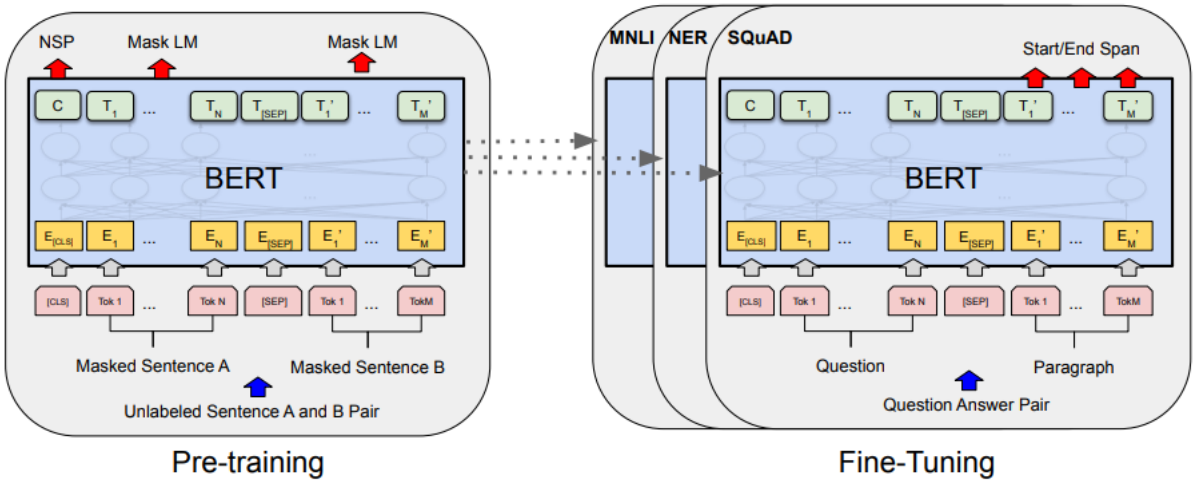

Figura 9: Esquema pasos BERT

Pel que fa a l'entrenament, es fa amb dues tasques principals, una és el MLM (Masqued Language Modeling) i l'altre és el NSP(Next sentence Prediction). EL MLM consisteix a trobar paraules aleatòries d'una oració, i el NSP en determinar si una oració segueix un altre.

En ser un encoder bidireccional té una gran capacitat per aprendre les representacions contextuals de les paraules, ja que, per una banda, agafa el context previ i per l'altra banda agafa el posterior.

GPT (Generative Pre-Training) en canvi, és un decoder basat en l'estructura Transformer, implementat per OpenAI en 2019.

Com s'ha vist anteriorment al BERT, aquest model també consta de dues etapes. La primera d'elles consisteix a entrenar aquest model amb un model lingüístic sobre un gran dataset.

Després d'aquest, es farà un *finne tunning* per ajustar-ho a una tasca específica. Un cop el model està entrenat, es pot utilitzar per a la generació de text automàticament. Donat un fragment, el model fa servir el seu coneixement per fer una continuació d'aquest de manera coherent.

Una de les característiques d'aquest model és la seva gran capacitat per generar text coherent, així i tot, aquest té les seves limitacions i no sempre té una precisió màxima.

3 Anàlisi del problema

Tenint en compte la descripció de manera abstracta i genèrica del problema a ser resolt en la secció 1.1, en aquesta secció seran descrites amb un major grau d'especificació les característiques de la solució que ha estat desenvolupada en el present treball.

Per una banda, definirem característiques tècniques de la solució que ha estat implementada, tant en l'àmbit global com un nivell més específic de cadascuna de les parts a partir de les quals ha estat construït el sistema, i, per altra banda, a escala de *software*, definirem les llibreries utilitzades i la implementació del codi.

En la data de realització del present treball no ha estat publicada a la UB cap eina d'intel · ligència artificial en la qual puguem obtenir, a partir del contingut d'una assignatura, algun tipus de model d'avaluació per aquesta assignatura. Els models d'avaluació poden consistir en preguntes de vertader o fals, o qüestions de selecció múltiple.

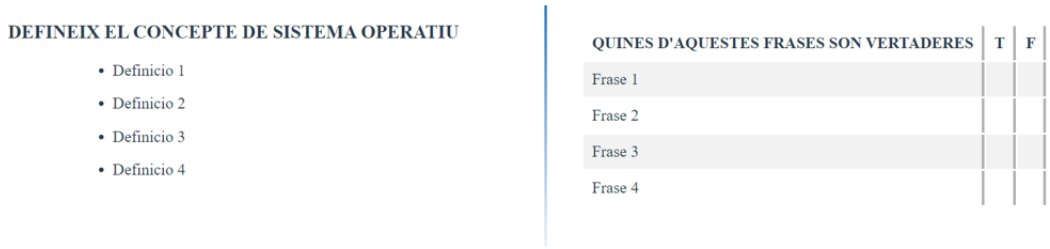

Figura 10: Exemple model de preguntes generades pel nostre model

Per tant, dins de l'absència d'eines de caràcter similars, en el present treball s'implementarà una eina mitjançant la qual puguin ser satisfets els requisits i es farà una valoració dels resultats obtinguts d'aquesta.

3.1 Descripció de l'arquitectura proposada

La proposta que fem es basa en una arquitectura global, composta per un conjunt d'eines d'intel · ligència artificial de Natural Language Processing, en la que utilitzem diferents models per diferents tasques específiques. Aquests models estaran preentrenats, amb això ens assegurem que es realitzi un entrenament amb un gran conjunt de dades que faran que millori el rendiment, i suposarà un gran estalvi de temps i recursos.

Per una banda, haurem d'extreure les paraules claus d'un text per agafar els conceptes claus que ens interessa que l'alumnat entengui.

Seguidament, haurem de relacionar aquestes paraules claus per plantejar un problema d'autoavaluació amb una certa dificultat, ja sigui per preguntes vertaderes o fals, o preguntes de selecció múltiple. El que ens interessarà és tenir definicions de conceptes relacionats entre si.

Finalment, haurem d'entrenar un model amb els continguts facilitats per poder definir els conceptes de l'assignatura.

3.1.1 Keyphrase Exctraction

La tècnica de keyphrase extraction o extracció de paraules clau, en NLP (Processament del Llenguatge Natural) consisteix a identificar les paraules o frases més importants d'un text.

Això es pot realitzar de diverses maneres, com ara l'anàlisi sintàctic i semàntic d'un text, l'aplicació de tècniques d'estadístiques de manera que puguem identificar les paraules amb més importància, o amb models de llenguatge preentrenats.

En el nostre cas utilitzarem un model preentrenat que hagi après la semàntica de les paraules. Per dur a terme aquesta tasca, ens interessarà un model que, a partir de l'embedding de les paraules, hagi estat entrenat per poder assignar-los una puntuació.

Ens interessarà que aquest model sigui un encoder, ja que, tindrem la capacitat de capturar les relacions entre les paraules d'un text. Això ens permet entendre el significat de la paraula tant en l'àmbit individual com en el conjunt del text, i suposa una millora en la precisió a l'hora d'extraure la paraula clau. Finalitzat aquest mòdul tindrem les paraules claus d'un text

3.1.2 Embedding Keyphrases

Tal com s'ha vist anteriorment, l'embedding consisteix a transformar dades en vectors numèrics, i que aquestes siguin descriptives. Es poden trobar variïs tipus de dades, com ara paraules, frases o documents, però en el nostre cas el format de dades seran frases, ja que l'objectiu és extraure un vector numèric dels conceptes definits al mòdul anterior.

Per a dur a terme l'embedding dels conceptes hi haurà un encoder, pels mateixos motius que hem comentat anteriorment. Aquest serà un model genèric d'encoder, com més entrenat millor, que sigui capaç de crear un vector de n dimensions.

El resultat d'aquest mòdul seran tots els termes claus representats en una matriu de m x n dimensions, on m serà el número de termes i n la seva representació numèrica.

Figura 11: Esquema funcionament embedding

3.1.3 Clustering

El clustering és una tasca d'anàlisis de dades que té com a objectiu agrupar en grups un conjunt d'elements amb característiques diferents però similars. Per dur a terme aquesta tasca no es coneix informació prèvia dels grups ni de les seves característiques. Aquests models són models no supervisats, ja que no hi ha classificacions correctes o incorrectes, el resultat es basarà en la tècnica de clustering que utilitzem per agrupar les dades.

Les tècniques calcularan la distància entre els registres i els clústers, minimitzant la distància entre els registres pertanyents al mateix clúster.

Per dur a terme aquest treball hem tingut en compte diferents mètodes d'agrupació:

- KMeans: L'algoritme Kmeans 12 defineix en l'inici el número de clústers k, i a continuació s'assignen k centroides a l'espai. A continuació anirem assignant a cada ítem al seu centroide més proper i de manera iterativa anirem actualitzant els valors dels centroides amb la posició mitja dels elements pertanyents al grup.
- Hierarchial clustering: L'algoritme de clustering jeràrquic 13 consisteix a enfocar el clúster de baix a dalt. Consisteix en agrupar cada element amb un únic grup i, anirem ajuntant els grups que tinguin la distància més curta per crear grups més grans.

3.1.4 Fine Tunning del model Seq2Seq

El fine tunning, és la tècnica que s'utilitza per millorar un model preentrenat. Consisteix en agafar un model entrenat amb un gran conjunt de dades, i entrenar-ho amb un conjunt de dades més petites.

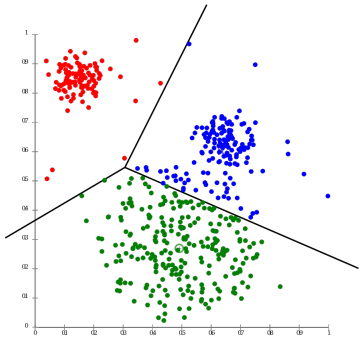

Figura 12: Clustering *k-means*.

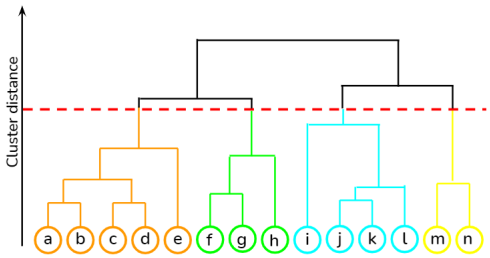

Figura 13: Clustering jeràrquic.

En el nostre cas ens servirà per adaptar un model de generació de text(seq2seq) preentrenat als nostres conceptes i que pugui acomplir la tasca de definicions de conceptes amb més precisió, i de la manera que vulgui ser explicat pel professorat. Això ho fem amb un model preentrenat perquè aquest tindrà un coneixement més ampli de tot, però el fine tunning ens aportarà més coneixement en un determinat domini.

3.1.5 Resultat

El resultat d'aquest model és el que veiem a la figura 14.

Seguint els diferents mòduls descrits anteriorment, començarem amb un text d'entrada i com a sortida tindrem grups de conceptes d'aquest text amb la seva definició corresponent.

A partir d'aquests outputs, jugarem amb els grups de paraules per generar preguntes de verdader o fals o d'escull la definició correcte.

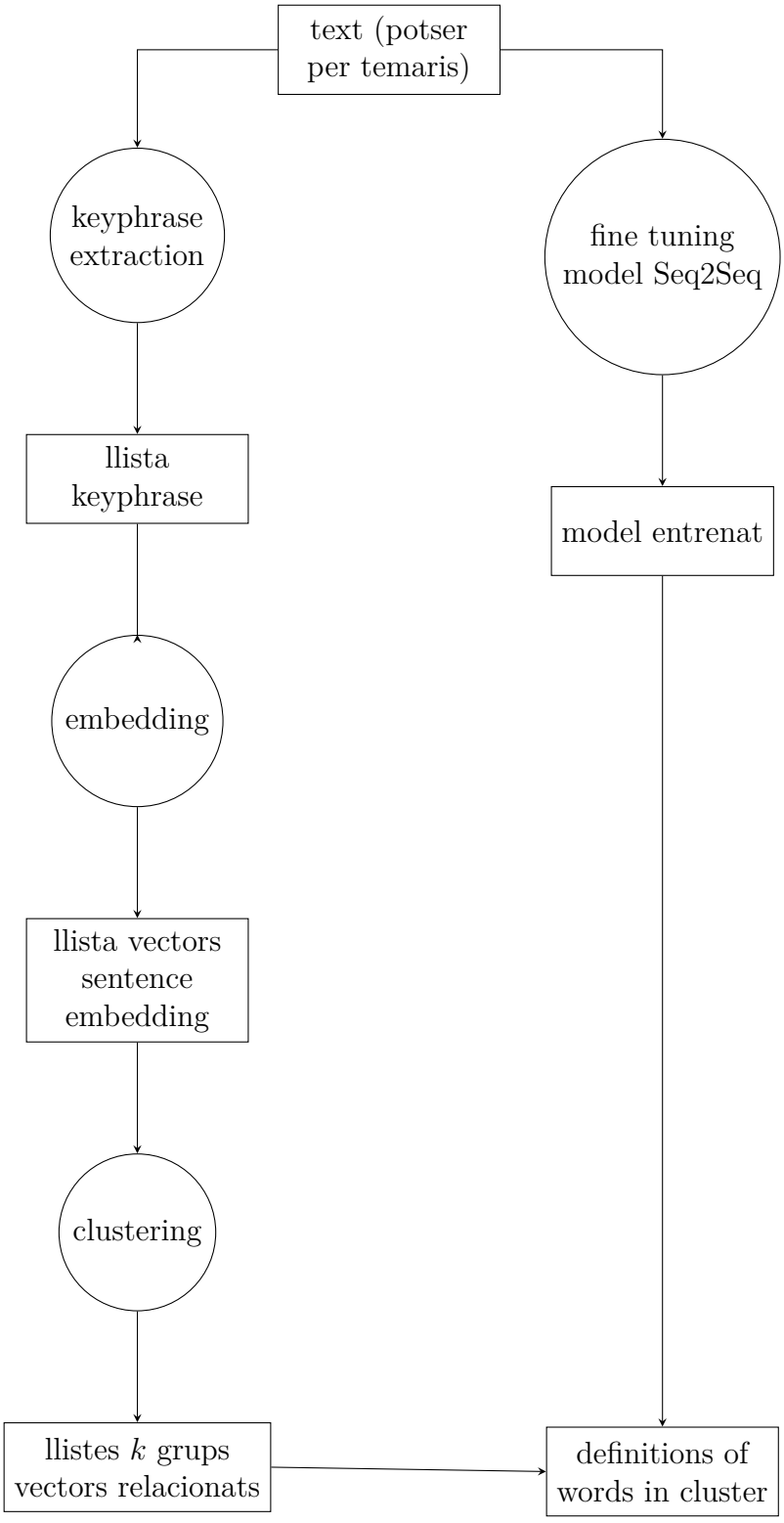

Figura 14: Esbós de l'estructura, respecte a blocs, del sistema on puguin ser obtingudes les definicions d'un concepte i múltiples conceptes claus relacionats.

3.2 Materials

Per les característiques del projecte que ha estat realitzat aquí, s'ha escollit el llenguatge de programació Python com el llenguatge de programació amb el qual s'ha escrit el codi de l'aplicació. L'elecció ha estat motivada, per una banda, per a la seva facilitat i flexibilitat, gràcies a la qual poden ser desenvolupats fragments de codi d'una elevada complexitat algorísmica en una quantitat substancialment baixa de línies de codi. I, per altra banda, en tenir una gran comunitat, té un ampli repertori de llibreries que ofereixen funcionalitats avançades per al processament i l'anàlisi del llenguatge natural.

A continuació seran exposades totes aquelles eines de software que han estat utilitzades durant la realització del present treball, juntament amb la justificació de la utilització d'aquestes respecte a unes altres de semblants.

3.2.1 HuggingFace

Hugging face: La llibreria Hugging Face és una llibreria que es fa servir per processar el llenguatge natural. És de codi obert i té com a objectiu principal oferir una interfície fàcil d'utilitzar per interactuar amb models de llenguatge preentrenats, així com per crear, entrenar i avaluar models personalitzats.

Aquesta llibreria té una gran quantitat de models de llenguatge pre-entrenats que duen a terme moltes tasques diferents, com ara la classificació de text, la generació de text, el reconeixement d'entitats i la traducció automàtica.

3.3 Implementació

A l'hora de plantejar la implementació de l'arquitectura proposada anteriorment s'ha decidit ubicar cadascun dels mòduls en directoris separats. L'estructura que ha seguit el projecte és la següent:

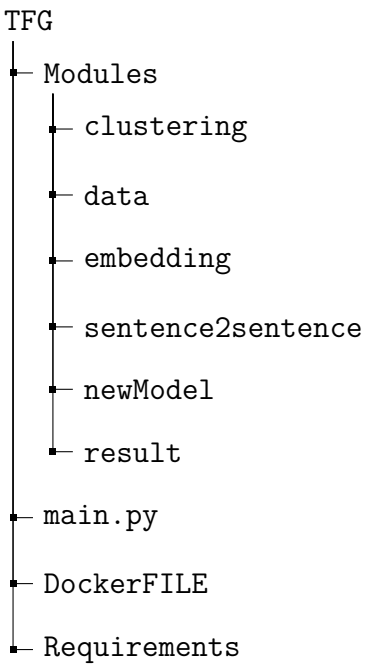

Figura 15: Esquema de directoris del projecte

Actualment, l'execució del projecte està de manera local en un entorn de python, però, en cas que es vulgui dockeritzar, es disposarà d'un DockerFile per a poder crear un contenidor amb el projecte.

En la imatge 15, es pot veure que a cadascun dels mòduls del projecte li correspon un directori. En cadascun d'aquest es proporcionarà les funcionalitats necessàries per a dur a terme totes les tasques del mòdul.

A part dels directoris citats anteriorment, també hi haurà tres directoris més. Per una banda, un directori anomenat data, qui serà el que contindrà el temari de l'assignatura, i d'on s'extrauran les dades per entrenar el nostre model. Aquest directori disposa d'una funcionalitat per convertir els fitxers de pdf a txt. Així i tot, en cas que es vulgui utilitzar aquesta, ha d'haver-hi una supervisió per comprovar que el text de la sortida sigui coherent. Per altra banda, estarà el directori de resultats, on es trobarà la sortida en format *txt* del nostre software. I finalment, estarà el directori model, on es guardarà el nostre model amb el finne tunning.

A l'hora d'executar el software s'obrirà un menú amb la tasca que es vol dur a terme, com es pot veure en la figura 16.

```
(venv) C:\Users\lroca\Documents\tfg_git\tfg>python main.py
1. Fine tunning
2. Keyphrase Extraction
3. Embedding and clustering
4. Get definitions
```

Figura 16: Captura de pantalla de l'execució del programa principal, on es mostra el menú de l'arquitectura prposada.

Si la tasca que es vol realitzar és el fine tunning, s'haurà d'introduir el text amb el qual es vol entrenar el model. El model de Seq2Seq que s'ha decidit utilitzar és el gpt2. La decisió d'escollir aquest model de la llibreria hugging face ha estat presa perquè aquest ha estat entrenat amb un gran nombre de dades i és un dels més recomanats per la comunitat, ja que el text generat té molta coherència i fluïdesa en comparació amb altres models.

La segona tasca és el keyphrase extraction. Les paraules claus seran extretes del mateix fitxer amb el qual es fa el finne tunning, i en aquest cas s'utilitzarà el model *Keyphrase Extraction Model: distilbert-inspec*. Aquest ha estat entrenat amb papers científics per trobar les paraules claus, per tant, en el nostre cas resulta molt útil.

Model Name	Performance Sentence Embeddings (14 Datasets)	Performance Semantic Search (6 Datasets)	Avg. Performance	Speed	Model Size
all-mpnet-base-v2	69.57	57.02	63.30	2800	420 MB
multi-qa-mpnet-base-dot-v1	66.76	57.60	62.18	2800	420 MB
all-distilroberta-v1	68.73	50.94	59.84	4000	290 MB
all-MiniLM-L12-v2	68.70	50.82	59.76	7500	120 MB
multi-qa-distilbert-cos-v1	65.98	52.83	59.41	4000	250 MB

Figura 17: Visualització dels diferents resultats de tots els models, a mode de comparativa entre tots ells.

A continuació, per la tasca d'embedding, s'ha fet una comparació de diferents models d'embedding 17, i s'ha decidit escollir en model *all-mpnet-base-v2*. La sortida de l'embedding serà una llista de vectors, i amb aquesta es farà el clustering amb una de les estratègies de clustering vistes en l'apartat 3.1.3 i s'extrauran els diferents grups de paraules.

Finalment, l'última tasca a fer serà extraure les definicions dels clusters. Un cop el model hagi estat entrenat, es seleccionarà un clúster de termes, i aquest model generarà una definició per cadascuna d'aquestes paraules. A l'hora de crear les preguntes es barrejaran els diferents termes amb les seves corresponents definicions.

Figura 18: Visualització dels diferents resultats de tots els models, a mode de comparativa entre tots ells.

Per altra banda, s'ha implementat una petita interfície gràfica per a poder visualitzar els resultats per millorar l'experiència de l'usuari, com podem veure en la Figura 18.

4 Resultats

En la part d'avaluació, en primer lloc, s'avaluarà de manera subjectiva la qualitat del text generat pel model, en segon lloc, farem una anàlisi dels vectors generats pel model d'embedding i el seu clustering, i, en tercer lloc, avaluarem les paraules extretes del model de keyphrase extraction.

Finalment, avaluarem la totalitat del pipeline, amb les limitacions d'aquests.

4.1 Generació de text

En la generació de text es tindrà en compte la coherència del text de sortida i la creativitat d'aquest, és a dir, si el text de sortida va més enllà d'agafar el text i copiar el seu contingut, o, en canvi, si es genera una resposta a partir d'aquest. També es tindrà en compte la rellevància de la resposta, és a dir, si entén la pregunta i respon adequadament a aquesta.

Per extraure uns resultats interessants del nostre model i veure les limitacions d'aquest s'han realitzat proves amb diferents datasets.

A continuació s'analitzaran els resultats de les proves que han estat realitzades de generació de text. Donat que l'entrenament no ha estat supervisat, ens centrarem en els resultats qualitatius.

Com a exemple es mostrarà diferents preguntes i respostes d'un dataset d'un text de xarxes neuronals recurrents, en el que parla de definicions de conceptes. En aquest text s'explica el funcionament de les xarxes neuronals. A continuació mostrarem un exemple d'un concepte que s'ha volgut definir.

En el següent text es pot veure el fragment de text del dataset en què s'explica la definició de xarxes neuronals feed forward:

A feedforward network, also known as a feedforward neural network or a multilayer perceptron (MLP), is a type of artificial neural network where information flows in a single direction, from the input layer to the output layer, without any cycles or loops. It is the most basic and widely used type of neural network architecture. In a feedforward network, the network is composed of multiple layers of neurons.

I introduint l'input preguntant per a la definició del concepte de feedforward network, la sortida és la seguent.

The feedforward network is a type of artificial neural network where information flows in a single direction, from the input layer to the output layer, without any cycles or loops. It is the most basic and widely used type

En l'extracció de resultats en diferents datasets, en generar respostes de definicions de diferents conceptes vistos en el text, s'ha pogut comprovar, que les respostes són bastant coherent, però, majoritàriament, es troben dos problemes. El primer és que, per la mida definida en l'inici, el model no sigui capaç de concloure el text de sortida. I el segon és quan la mida definida és massa extensa pel tema concret. En aquesta situació el que passarà serà que, la part final del text de sortida, no serà coherent amb el demanat.

També s'ha vist que quan el dataset té conceptes definits amb nomenclatura matemàtica, ja sigui fórmules, gràfiques, etc. el model no té la capacitat per poder contextualitzar aquestes i en aquests casos les respostes generades manquen de coherència.

Per altra banda, s'ha vist que la creativitat de la resposta no és gaire més extensa que el que redacta el text. En l'exemple anterior es pot veure que el text generat és gairebé idèntic al del dataset.

Finalment, s'ha pogut comprovar que la rellevància de la resposta és molt bona, tot i que si es troba en diferents fragments, a vegades li costa relacionar-los entre si.

És interessant veure com aquest exemple relaciona el concepte generant una definició en diferents fragments del text.

The input layer receives the input data, and each neuron in the input layer is connected to the neurons ...

Each connection between neurons has a weight associated with it, which determines the strength of the connection. The output of each neuron in a given layer is computed as a weighted sum of the outputs from the neurons in the previous layer, and an activation function is applied to this sum to produce

I preguntant sobre la definició del concepte d'*input layer*, el model respon el següent:

The input data is a list of weights, where each weight in the input layer is an element of the network, and each element in the input layer is an element of the network.

Com es pot comprovar en la generació de resposta s'aconsegueix relacionar el concepte en diferents mencions que fa el text

4.2 Sentence Embedding

En el model de sentence embedding, per una banda, s'avaluarà la qualitat del vector generat a partir de la relació semàntica que es captura entre els conceptes, i, per altra banda, s'avaluarà els diferents grups de clusters generats a partir dels vectors d'embedding.

En un inici es mostrarà un exemple d'un parell de resultats extrets en els diferents experiments fets, primer en l'embedding i després en el clustering, finalment es valorarà a trets generals els resultats dels experiments.

A causa de les grans dimensions de la taula que s'han generat amb l'escriptura de totes les classes, aquestes classes són referenciades amb un número i a la taula seran escrits únicament els números.

La taula d'assignacions dels números és la següent:

1. **Artificial Intelligence**
2. **Web Development**
3. **Natural Language Processing**
4. **Cybersecurity**
5. **Ecology**
6. **Global warming**
7. **Heavy metal**
8. **Jazz**

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	1.0000	0.2613	0.4837	0.3443	0.2490	0.2808	0.2969	0.1926
2	0.2613	1.0000	0.3585	0.3520	0.1387	0.0914	0.1912	0.0725
3	0.4837	0.3585	1.0000	0.3107	0.2188	0.1603	0.1757	0.1124
4	0.3443	0.3520	0.3107	1.0000	0.2978	0.3071	0.3237	0.1698
5	0.2490	0.1387	0.2188	0.2978	1.0000	0.4792	0.2442	0.2052
6	0.2808	0.0914	0.1603	0.3071	0.4792	1.0000	0.2790	0.1503
7	0.2969	0.1912	0.1757	0.3237	0.2442	0.2790	1.0000	0.4179
8	0.1926	0.0725	0.1124	0.1698	0.2052	0.1503	0.4179	1.0000

Taula 1: Taula recopilatòria dels graus de similitud entre els diferents conceptes.

De forma anàloga amb l'anterior taula, en aquesta taula també seran referenciades les paraules per els següents identificadors.

1. **Artificial Intelligence**
2. **Computer Vision**
3. **Natural Language Processing**
4. **Cybersecurity**
5. **Malware**
6. **Factory Method**
7. **Singleton**
8. **Continuous Integration**

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	1.0000	0.4729	0.4837	0.3443	0.3437	0.1554	0.0474	0.3295
2	0.4729	1.0000	0.3191	0.2408	0.2530	0.1161	-0.0060	0.2013
3	0.4837	0.3191	1.0000	0.3107	0.2978	0.1616	0.0133	0.2213
4	0.3443	0.2408	0.3107	1.0000	0.4626	0.0684	0.0590	0.2715
5	0.3437	0.2530	0.2978	0.4626	1.0000	0.1574	0.0374	0.1723
6	0.1554	0.1161	0.1616	0.0684	0.1574	1.0000	0.3025	0.1662
7	0.0474	-0.0060	0.0133	0.0590	0.0374	0.3025	1.0000	0.0715
8	0.3295	0.2013	0.2213	0.2715	0.1723	0.1662	0.0715	1.0000

Taula 2: Taula recopilatòria dels graus de similitud entre els diferents conceptes.

A l'hora d'extreure resultats amb el clustering s'ha vist que generalment la qualitat dels resultats que s'han obtingut s'adeqüen al nivell d'exigència esperat. A continuació es mostraran dos exemples que s'han considerat rellevants a l'hora de mostrar els resultats assolits.

En la taula 2 es mostra una taula, en la que es relaciona la similitud de l'embedding de cadascun dels parells de conceptes, i s'obté un score de similitud, on 1 és el màxim valor. Fent un embedding de conceptes més genèrics en què es tracten conceptes de diferents temàtiques. En aquest, el model és capaç de separar tots els conceptes d'una mateixa temàtica, i com es pot veure la puntuació de cada relació de paraules té relació amb la similitud d'aquestes.

A continuació, en l'exemple 3 es fa una comparació de conceptes més específics d'una temàtica en concreta, com és en aquest cas diferents temàtiques amb relació a la *computer science*. Com es pot veure a la puntuació, el clustering realitzat té un gran nivell de precisió.

A continuació mostrarem el clustering d'aquest segon exemple:

- Cluster 1 ['Cybersecurity', 'Malware']
- Cluster 2 ['Artificial Intelligence', 'Computer Vision', 'Natural Language Processing', 'Continuous Integration']
- Cluster 3 ['Factory Method', 'Singleton']

Aquesta sortida és generada per l'algoritme *aglomerative clustering*. S'ha decidit utilitzar aquest algoritme no supervisat perquè a l'hora d'agrupar els grups de clusters, el nombre de membres de cadascun dels grups, està definit per la distancia, per aquest motiu, els conceptes sempre sòn similars independentment dels grups de clusters.

En el clustering dels conceptes citats anteriorment, es pot observar que els grups que forma s'adapten a les puntuacions extretes anteriorment.

4.3 Keyphrase extraction

L'últim model a avaluar és el model de keyphrase extraction. S'ha fet una sèrie de proves amb els diferents textos i s'han identificat diversos problemes.

```
 [
'computer science', 'artificial intelligence', 'complex problems',
'search algorithms', 'binary search', 'sorted lists', 'middle element',
'logarithmic complexity', 'data sets', 'root node', "neighbors ' neighbors",
'queue structure', 'root', 'heuristic search algorithmuristic function',
'beam search', 'local search resources tools', 'fields', 'empower problems'
    ]
```

Figura 19: Sortida model de keyphrase extraction

L'exemple que es pot veure en la figura 19 es basa en un text del dataset que parla sobre algoritmes de cerca. En aquest es pot comprovar que gairebé totes les paraules que extrau són d'àmbit científic i són conceptes interessants.

Així i tot, s'ha vist que el seu output presenta una sèrie de limitacions que s'exposaran a continuació

- En primer lloc, aquest model és capaç d'extraure un gran nombre de conceptes que poden resultar interessants, però la gran majoria d'aquests no estan definits en el text. Com es pot veure en l'exemple, en algun moment es cita la intel · ligència artificial, però en cap moment es defineix el concepte.
- En segon lloc, a vegades es pot veure que no acaba d'agafar bé les paraules, com és en l'element de la llista *neighbors ' neighbor*. Per tant, l'acuraccity del model no és del cent per cent.
- Finalment, hi ha molts conceptes que no és capaç d'extraure, com seria en l'exemple del text sobre algoritmes de cerca, els conceptes de Depth-First Search o Breadth-First Search estan definits en el dataset però no són identificats.

En conclusió, es considera que aquesta part del pipeline, si es vol utilitzar, ha de ser supervisada, igual que el fine tunning, per així poder adaptar la tasca del model a la nostra tasca específica.

4.4 Pipeline

Finalment, s'avaluarà el conjunt del pipeline. En consideració amb els problemes citats anteriorment, es veu que aquest pipeline té una sèrie de limitacions.

Per una banda, s'ha vist que el mòdul de Keyphrase extraction necessita una supervisió per part de l'usuari, en primer lloc, per comprovar que les paraules claus no són errònies, en segon lloc, per comprovar que aquestes estan definides al text, i per acabar, per

introduir els conceptes que no apareixen en el text i que són interessants que estiguin en la llista.

No obstant això, s'ha vist que la part de clustering es fa de manera correcta, i la part de generació de la resposta, en cas que el concepte estigui definit en el text, majoritàriament la rellevància de la resposta és bastant bona, tot i que no sigui gaire creativa.

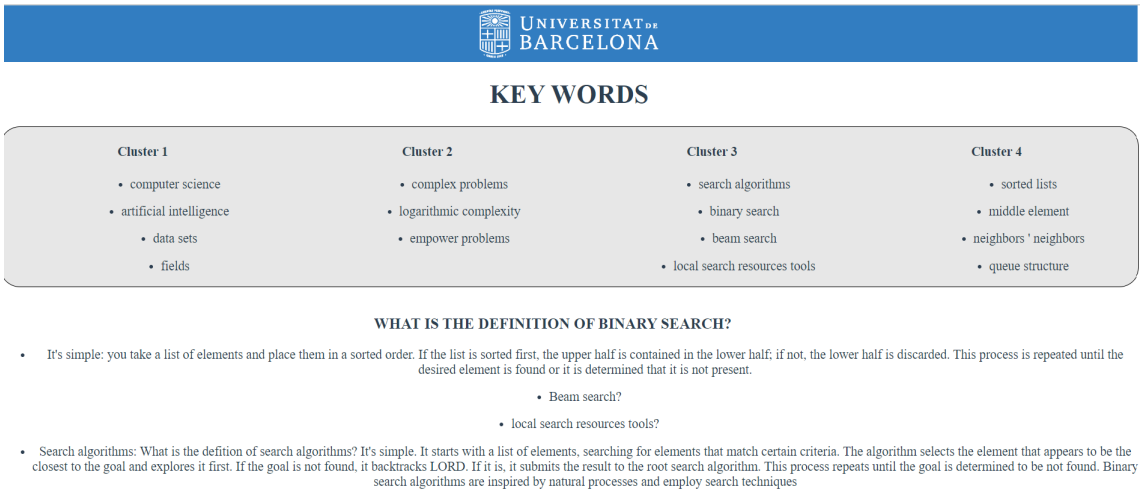

Figura 20: Visualització de resultats del pipeline.

En la imatge 20, a causa de la mala extracció de conceptes claus, l'error s'arrossega i en el moment de generar la pregunta *multichoice* , s'escull el cluster 3, que està format per les paraules *search algorithms, binary search, beam search i local search resources tools*. D'aquests conceptes, l'únic que està descrit és el de binary search, i una petita definició de search algorithms, però els altres dos conceptes no. És per aquest motiu que a l'hora de generar la pregunta jugant amb les definicions, la primera i la quarta són les úniques que es poden donar per vàlides.

Vista la problemàtica anterior, s'ha decidit fer proves del pipeline de manera que l'extracció de paraules claus sigui supervisada, amb l'opció d'eliminar les paraules que no estiguin definides en el text, i afegir les que no han sigut trobades pel model.

D'aquesta manera, justament amb l'obtenció dels clusters, ens permet la generació de preguntes com les vistes en l'exemple 3. Agafant un cluster vàlid, es juga amb les seves definicions i permet generar preguntes vàlides, tant de multiselecció, com de True o False.

Select the correct answer : the Depth-First Search (DFS)	
it combines the advantages of both BFS and DFS. IDDFS is designed to improve readability by reducing the order in which data is processed. It starts at a root node and explores each branch until the goal is found. It uses a logarithmic scale to reduce the size of the list. In summary, IDDFS is a powerful algorithm for finding the optimal solution to a puzzle	
it starts at a root node and explores all its neighbors. If it finds one, it compares its neighbors to find the root node. If it finds none, it compares its neighbors to reject the root node. If it finds a root node, it alternates neighbors by choosing an alternate neighbor and exploring the root node. While the process is relatively simple, it's worth mentioning that it takes a minute or so to complete.	
it starts at a root node and explores all its neighbors before moving on to the neighbors' neighbors. It uses a queue data structure to store the nodes to be explored, ensuring that nodes closer to the root are visited first.	

Definició de conceptes	**True**	**False**
the Depth-First Search (DFS) It combines the advantages of both BFS and DFS. IDDFS is designed to improve readability by reducing the order in which data is processed. It starts at a root node and explores each branch until the goal is found. It uses a logarithmic scale to reduce the size of the list. In summary, IDDFS is a powerful algorithm for finding the optimal solution to a puzzle		
The Breadth-First Search (BFS) It starts at a root node and explores all its neighbors. If it finds one, it compares its neighbors to find the root node. If it finds none, it compares its neighbors to reject the root node. If it finds a root node, it alternates neighbors by choosing an alternate neighbor and exploring the root node. While the process is relatively simple, it's worth mentioning that it takes a minute or so to complete.		
The Breadth-First Search (BFS) It starts at a root node and explores all its neighbors before moving on to the neighbors' neighbors. It uses a queue data structure to store the nodes to be explored, ensuring that nodes closer to the root are visited first.		

Taula 3: Taula recopilatòria de cadascun dels models de preguntes amb una pregunta generada pel model.

5 Conclusions

Després d'haver implementat tot el codi de l'aplicació, a més de realitzar tota la investigació prèvia, tant en l'àmbit de buscar treballs similars com en l'anàlisi de les eines i metodologies que hem utilitzat, s'han extret algunes conclusions i consideracions, a escala global del treball.

Aquestes conclusions engloben totes les possibles millores del treball que aquí s'ha fet, tant pel que fa a possibles millors maneres d'haver encarat algunes parts de la realització d'aquest, problemes trobats, com futures implementacions.

5.1 Problemes trobats

Tal com s'ha exposat abans, durant la realització d'aquest treball s'han trobat problemes estructurals més grans o problemes de la qualitat d'alguns dels mòduls que hem fet servir. A continuació es fa una enumeració del subconjunt més rellevant de tots aquests problemes.

5.1.1 Insuficiència de capacitat computacional

A conseqüència de la falta d'accés a algun tipus de màquina d'altes prestacions, com podria ser un servidor, hi ha hagut diverses parts del procés de realització del codi que no s'han pogut fer de la millor manera possible.

Més concretament, tot el desenvolupament del codi ha hagut de ser fet en un ordinador local, però, per culpa de la poca capacitat computacional d'aquest ordinador, l'execució del model no s'ha pogut realitzar en aquest ordinador.

Per aquest motiu, s'ha hagut de fer ús de màquines remotes especialitzades en l'entrenament de models, a les quals hem accedit mitjançant l'eina Google Colab.

5.1.2 Necessitat de supervisió

A conseqüència del fet que el model d'extracció de paraules té grans limitacions, es necessitaria una supervisió per part de l'usuari que fa l'entrenament per poder aconseguir preguntes que puguin ser vàlides.

Altrament, aquestes imperfeccions de les dades del dataset es poden propagar per tot l'entrenament del model, causant un deteriorament dels resultats d'aquest.

5.2 Possibles ampliacions futures

El resultat final d'aquest treball es pot millorar de diverses maneres, en aquest apartat explicarem cadascuna de les propostes per la seva optimització.

5.2.1 Unificació dels models

Una possible ampliació seria intentar utilitzar el mateix model gpt2 per fer l'embedding. Es va provar de fer experiments amb l'embedding del gpt2 a partir d'un embedding de l'última capa del model, però com no és un model específic per aquesta tasca s'hauria de mirar de modificar el model per a obtenir més bons resultats. Si el model fos capaç de dur a terme aquesta tasca, s'aconseguiria que, amb el mateix finne tunnning del gpt2, pogués servir per l' embedding d'aquest i només tindrem la necessitat de fer ús dos models en lloc de tres.

5.2.2 Ús model més potent

Durant la recerca, s'ha vist que l'API que procorciona *openAi*, que és de pagament, pot proporcionar un model com el gpt 3 o gpt 4, que tant en la generació de text com en la interpretació d'aquest és molt més òptim. Per tant, si en un futur es volgués donar un ús més real i aconseguir uns millors resultats, seria una opció molt interessant.

Un altre possible implementació podria ser agafar un major ventall de preguntes. Tal com està dissenyada la nostra arquitectura és capaç de generar dos models de preguntes bastant limitades, l'avantatge d'això, és que són de definicions de conceptes i no resulten gaire obertes, per tant, no requereixen gaire supervisió. Però en un futur, el mateix model que genera les respostes, podria generar preguntes del text, però això suposaria una major supervisió d'aquestes i es necessitaria un model més potent.

5.2.3 Entrenament supervisat

S'ha vist que els resultats, sobretot del keyphrase extraction, no són gaire òptims.

Una possible optimització seria fer un finne tunning amb diferents datasets supervisats, tant per al keyphrase extraction com pels altres models.

Referències

[1] Recurrent neural network based language model
https://www.fit.vutbr.cz/research/groups/speech/publi/2010/mikolov_interspeech2010_IS100722.pdf

[2] Neural Machine Translation by Jointly Learning to Align and Translate
https://arxiv.org/abs/1409.0473

[3] Attention Is All You Need
https://arxiv.org/pdf/1706.03762.pdf

[4] BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding
https://arxiv.org/abs/1810.04805

[5] Feed Forward Neural Network Definition | DeepAI
https://deepai.org/machine-learning-glossary-and-terms/feed-forward-neural-network

[6] FeedForward Neural Networks: What is Feed Forward | Built In
https://builtin.com/data-science/feedforward-neural-network-intro

[7] Two minutes NLP — Keyword and keyphrase extraction with KeyBERT
https://medium.com/nlplanet/two-minutes-nlp-keyword-and-keyphrase-extraction-with-keybert-a9994b06a83

[8] Keyword Extraction Methods from Documents in NLP
https://www.analyticsvidhya.com/blog/2022/03/keyword-extraction-methods-from-documents-in-nlp/

[9] EmbedRank: Simple Unsupervised Keyphrase Extraction using Sentence Embeddings
https://towardsdatascience.com/embedviz-simple-unsupervised-keyphrase-extraction-using-sentence-embeddings-97ed5e16ad00

[10] Como funcionan los transformers?
https://www.aprendemachinelearning.com/como-funcionan-los-transformers-espanol-nlp-gpt-bert/

[11] KeyPhrase extraction using sentence embeddings (Unsupervised Learning)
https://medium.com/analytics-vidhya/keyphrase-extraction-using-sentence-embeddings-unsupervised-learning-21cc5a296396

[12] Simple Unsupervised Keyphrase Extraction using Sentence Embeddings
https://aclanthology.org/K18-1022/

[13] sklearn.cluster.KMeans
https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

[14] Dockerfile reference | Docker Documentation
https://docs.docker.com/engine/reference/builder/

Annex

MANUAL D'USUARI

CREACIÓ CONTAINER

DOCKER EXECUTION

docker build -t getting-started docker buildx build ./Dockerfile

VENV EXECUTION

(CREAR EL VENV) python -m venv venv(et crea una carpeta venc al directori on es executada la commanda) (ACCEDER AL VENV) ./venv/Scripts/activate

Un cop al entorn virtual instalar tots els requeriments descrits el en fitxer requirements.txt.

EXECUCIÓ PROJECE

Per executar el projecte s'ha d'especificar el directori on es troba el arxiu amb el que es vol entrenar el projecte. Aquest sera especificat al *main*.

Tmbé s'haura de especificar el directori on volem que es guardin els resultats i el nou model entrenat.

PLANIFICACIÓ PROJECE