Generació Automàtica de Preguntes

Biel Castaño Segade

Resum– La generació automàtica de preguntes és un procés mitjançant el qual un model informàtic és capaç d'obtenir preguntes basades en una entrada -habitualment, un text-. Un model capaç d'executar aquesta tasca té nombroses aplicacions pràctiques, tals com contribuir a la creació d'un tutor intel·ligent o un model de conversació automàtica. Així mateix, la generació automàtica de preguntes també és rellevant de cara a la creació de bases de dades que permetin l'entrenament d'un model generador de respostes. En aquest article es presenten els fonaments teòrics i la metodologia seguida per a crear un model generador de preguntes a partir dels *transformers*, un tipus d'arquitectura de xarxes neuronals que és àmpliament utilitzada en l'estat de l'art del processament del llenguatge natural.

Paraules clau– preguntes, pytorch, transformer, $t5$, xarxa neural

Abstract– Automatic question generation is a process by which a computer model can obtain questions based on input - usually text. In addition to their relevance in the creation of databases that train automatic answer generation models, such models have other practical applications, such as contributing to the creation of intelligent tutor and conversational models. This article outlines the theoretical foundations and methodology behind the creation of a question generation model from a transformer, the kind of neural network architecture widely used in the state of the art of natural language processing.
Keywords– question, pytorch, transformer, $t5$, neural network

1 Introducció - Context del treball

El processament del llenguatge natural és una branca de la intel·ligència artificial i la informàtica que s'encarrega de resoldre els problemes relacionats amb les llengües humanes computacionalment. Això inclou tant tasques de baix nivell -anàlisis lèxica, semàntica, sintàctica, morfològica-, com aplicacions a alt nivell -generació de resums d'un text, traducció automàtica, reconeixement de la parla [1]-.

Aquest treball se centrarà en la tasca relativa a la generació automàtica de preguntes, és a dir, generar preguntes a partir d'una certa entrada, per exemple text, àudio o imatges. Un generador de preguntes té diverses aplicacions pràctiques, les més destacades són [2]:

- Resposta de preguntes. Un model capaç de trobar la resposta a les preguntes d'un text seria el problema invers a la generació de preguntes, de manera que un generador de preguntes pot ser utilitzat per a construir les bases de dades que entrenen un generador de respostes.
- Comprensió de textos. Un generador de preguntes es pot fer servir per a crear datasets prou grans que permetin entrenar un model de comprensió de textos.
- Conversació automàtica. Un generador de preguntes també pot ser útil per a crear un model capaç de mantenir un diàleg amb un humà.
- Tutor intel·ligent. Un generador de preguntes és necessari a l'hora de crear un tutor intel·ligent educacional, ja que caldrà que aquest sigui capaç de crear preguntes a partir del contingut educacional.

Més concretament, la generació de preguntes pot englobar diferents tipus de problemes, podent-los classificar segons tres característiques diferents [2]:

1. Tipus d'entrada:
    - Text amb context a nivell de document.
    - Text amb context a nivell de paràgraf.
    - Text amb context a nivell de frase.
    - Text amb context a nivell de paraules clau.

• E-mail de contacte: biel.castano@autonoma.cat
• Mención realitzada:Computació
• Treball tutoritzat per: Ernest Valveny Llobet (Dept. Ciències Computació)
• Curs 2021/2022

- Imatge. En aquest cas, si es tracta de la imatge d'un text, es pot fer servir un *OCR* per transformar el problema en un dels descrits als anteriors punts.
- Àudio. De nou, es podria usar un sistema reconeixedor de la parla per obtenir un problema dels descrits als punts anteriors.

2. Resposta desitjada:

   - Resposta coneguda i seqüencial. En aquest cas, la resposta desitjada és coneguda i està continguda seqüencialment en el text, de manera que el generador de preguntes ha de generar una resposta per a la pregunta donada.
   - Resposta coneguda i abstracta. Similar al cas anterior, però ara la resposta no està continguda seqüencialment al text.
   - Resposta desconeguda. En aquest cas el generador de preguntes haurà de ser capaç de generar respostes, i posteriorment, les preguntes corresponents.

3. Preguntes generades:

   - Preguntes aïllades. En aquest cas el generador de preguntes generarà preguntes independents i sense interacció.
   - Preguntes seqüencials. El generador de preguntes generarà una sèrie de preguntes interconnectades seqüencialment.
   - Preguntes d'opció múltiple. En aquest cas les preguntes tindran el format d'opció múltiple, és a dir, es proporcionaran múltiples respostes a la pregunta tancada, de les quals només una és correcte.

Aquest projecte se centrarà en els generadors de preguntes on el tipus d'entrada és text amb context a nivell de frase, la resposta desitjada és coneguda i seqüencial, i les preguntes generades són preguntes aïllades. En la resta del document, en referir-se a un generador de preguntes es farà referència a aquest tipus de model.

Amb tot, els objectius del projecte són:

- Servir d'introducció als mètodes i tecnologies més usades pel que fa al *Deep Learning* i el processament del llenguatge natural.
- Triar i estudiar el funcionament d'una base de dades per a l'entrenament d'un model generador de preguntes.
- La creació d'un primer model funcional utilitzant xarxes neuronals preentrenades que segueixin l'arquitectura dels *transformers*.
- La millora i modificació d'aquest model inicial fins a obtenir un model amb un rendiment prou bo.
- Usar el model utilitzat per a crear una pipeline capaç de generar preguntes sobre la imatge d'un text mitjançant un *OCR*.

2 ESTAT DE L'ART

En aquest apartat s'explicarà quina és la tendència actual a l'hora de definir models generadors de preguntes. També es presentaran les bases de dades més utilitzades en l'actualitat per a entrenar aquest tipus de models.

2.1 Models generadors de preguntes

Els mètodes clàssics per a dissenyar un generador de preguntes es basen en regles heurístiques que primer seleccionen un context i posteriorment construeixen una resposta. Pel que fa a la selecció de context, el model troba un tema pel que preguntar en el text a partir d'un parser semàntic o sintàctic, i posteriorment s'usa la representació del parser per a reformular la informació en forma de pregunta [2].

Es tracta d'una aproximació problemàtica pel fet que definir un model d'aquest tipus requereix una quantitat inviable de temps i un domini de la temàtica del text, de manera que és difícilment escalable. Per tant, darrerament s'ha optat per usar models de *Deep Learning* que siguin més generalitzables i escalables. Els tipus de models més utilitzats en aquest sentit són els models tradicionals Seq2Seq basats en xarxes neuronals recurrents o en xarxes neuronals convencionals. Aquests darrers anys, sobretot es fan servir *transformers* i models Seq2Seq preentrenats, com per exemple el $t5$ [3]. A la secció 4 es presenta una explicació més detallada d'aquests models.

2.2 Bases de dades

Alguns dels *datasets* més utilitzats per a entrenar el model que resol la tasca de generació de preguntes considerada són [2]:

- *sQuAD 1.1*. Es tracta d'un *dataset* de Standford amb cent mil parelles de preguntes i respostes proporcionades per usuaris de *Wikipedia* [4].
- *NewsQA*. Aquest *dataset* conté cent mil parelles de preguntes i respostes generades per humans basades en articles de la cadena de televisió CNN [5].
- *SearchQA*. Aquesta base de dades conté més de 140 mil parelles de preguntes i respostes. A més, cada parella té uns 50 snippets que permeten augmentar les parelles utilitzant text snippets recuperats de *Google* [6].

En aquest projecte, s'ha triat el *dataset* l'sQuAD pel fet de ser el *dataset* més utilitzat en l'estat de l'art, amb l'objectiu de tenir més bibliografia disponible.

3 METODOLOGIA

La metodologia que s'ha seguit durant el desenvolupament d'aquest projecte és la metodologia agile, ja que es tracta d'un tipus de metodologia que permet fer adaptacions en el projecte a mesura que aquest progressa, per tant, és idònia per al projecte plantejat, que compta amb objectius opcionals.

En conseqüència, es planteja un desenvolupament incremental i iteratiu, amb reunions setmanals amb el tutor del projecte per tal de plantejar el seguiment, modificacions,

planificacions i documentacions adequades en cada punt del projecte.

Cada iteració del cicle de vida del projecte inclou una planificació, una anàlisi de requisits, un disseny i implementació, proves, i l'elaboració de la documentació adequada. Aquesta metodologia ha permès que el projecte avancés més ràpid i que hagi sigut més fàcil identificar quins són els elements importants i els errors en cada moment del projecte. A més, també ha proporcionat una retroalimentació més ràpida, així com una gran flexibilitat i adaptabilitat.

En particular, les iteracions del projecte han estat:

- Revisió i estudi de la bibliografia existent.
- Estudi i pràctica dels mètodes i tècniques bàsiques del *Deep Learning*.
- Obtenció d'una primera versió funcional d'un model generador de preguntes utilitzant una xarxa neuronal preentrenada $t5$.
- Millorar el model inicial fins a obtenir un rendiment prou bo.
- Desenvolupar i valorar altres models per comparar-los amb l'inicial.
- Tancar el projecte i elaborar la documentació final.

4 FONAMENTS TEÒRICS

4.1 Introducció als *Transformers*

Els *Transformers* són un tipus d'arquitectura basada en *Deep Learning* que ha estat àmpliament utilitzada durant els darrers anys en l'àmbit del processament del llenguatge natural. Aquesta arquitectura va ser proposada per primer cop l'any 2017 a l'article *Attention is All You Need* [7] per part d'investigadors de la companyia *Google*. Aquest article buscava suggerir un model que donés una empenta als models de traducció de textos usats fins al moment, habitualment basats en models de xarxes neurals recurrents i *LSTM's*.

Més concretament, els models de traducció fets servir en aquell moment es basaven en una filosofia encoder-decoder estrictament seqüencial, on les posicions dels símbols dins d'una frase a processar avançaven seqüencialment a través dels diferents components del model. Aquesta manera de tractar les dades planteja una limitació important quant al cost temporal de l'entrenament, de manera que un dels objectius principals dels *Transformers* és proporcionar un model més paral·lelitzable [7].

Un concepte clau per aconseguir aquesta paral·lelització és l'ús de la *attention* [7, 10], un mecanisme consistent en un mètode capaç de capturar les relacions existents entre totes les parelles de paraules d'una frase independentment de la seva distància dins de la frase. A més a més, aquest concepte també permet l'obtenció de millors resultats en el processament del llenguatge, ja que millora el comportament dels models al tractar frases llargues.

Posteriorment, els *Transformers* s'han utilitzat en nombroses tasques del processament del llenguatge natural i dels models *seq2seq*, i fins i tot en altres àmbits tals com la visió per computador. En particular, s'han arribat a fer servir *Transformers* en el tema que afecta aquest informe, la generació de preguntes [3]. S'ha decidit que s'emprarà un tipus de model derivat dels *Transformers* durant l'elaboració d'aquest treball. Així, aquest apartat exposarà les bases teòriques dels *Transformers* i del model particular usat durant el projecte.

4.2 Arquitectura dels *Transformers*

L'arquitectura típica d'un *Transformer*, proposada a [7], es mostra a la figura 1. En aquesta figura s'observen diferents mòduls, que bàsicament conformen els mòduls de processament de l'entrada, mòduls de processament de l'output del model, mòduls d'encoder i mòduls de decoder. En el transformer original, es suggeria un conjunt de cinc mòduls encoder encadenats seqüencialment, i un conjunt de cinc mòduls decoder encadenats seqüencialment i connectats amb l'últim mòdul encoder[1]. A l'apèndix s'inclouen figures que mostren amb més detall quina és l'arquitectura dels transformers.

El comportament de l'arquitectura serà lleugerament diferent en l'entrenament i la inferència:

- **Entrenament:** En aquest cas, l'entrada de l'encoder serà una sentència del conjunt d'entrenament, que en el cas del *Question Generation*, serà el context de la qüestió juntament amb la resposta donada. Per altra banda, s'aplicarà *masked language modeling* sobre la seqüència d'entrada del decoder, que en aquest projecte és la pregunta desitjada.

  El *masked language modeling* consisteix a enmascarar la seqüència d'entrada del decoder per tal que el model, a l'hora de predir un token determinat, només tingui en compte els tokens que apareguin a la sentència en posicions anteriors.

  L'output serà un vector de probabilitats de la mida del vocabulari, indicant la probabilitat que la propera paraula predita sigui cada paraula del vocabulari. Es calcularà una funció de cost respecte a aquestes probabilitats i la paraula esperada, i es realitzarà *backpropagation* per tal d'entrenar els diferents mòduls del model.

- **Inferència** En el cas de la inferència, es recorrerà el model iterativament. Per tant, l'entrada serà el context i la resposta donada. Per altra banda, l'entrada del decoder serà inicialment el *token* $< start >$. S'executarà el model i es seleccionarà la paraula w_1 amb més probabilitat segons el seu output. Llavors, es tornarà a executar el model, però aquest cop l'entrada del decoder serà $< start > +w_1$, on $+$ indica la concatenació. Es tornarà a executar el model i s'obtindrà una nova paraula que es concatenarà a l'entrada del decoder. Se seguirà iterativament aquest procés fins a assolir el token $< stop >$ que indica el final de frase.

En el següent apartat s'explicarà en detall cada component de l'arquitectura.

[1] Realment, es podria triar qualsevol nombre d'encoders i decoders, de manera que es tracta d'un hiperparàmetre

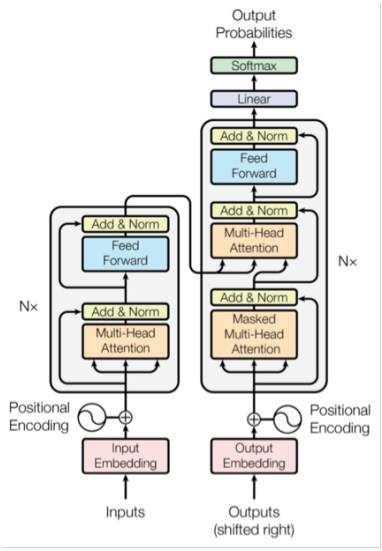

Fig. 1: Arquitectura d'un transformer bàsic [7].

4.3 Components

4.3.1 Word Embedding

El primer component habitual en els *Transformers*, així com en la pràctica totalitat dels models *seq2seq*, és el mòdul que s'encarrega de calcular els *Word Embeddings*. Es tracta d'un mecanisme consistent en, donada una paraula, assignar-li un *Word Embedding* [8], és a dir, un vector numèric que l'identifica, per tal que pugui servir d'entrada per a les xarxes neurals *Feed Forward* que conté el model, així com per facilitar una implementació del mecanisme de la *attention*.

Un exemple trivial de mecanisme de *Word Embedding* seria els *one-hot vectors*, consistents a treballar amb vectors de dimensió igual a la mida del vocabulari considerat, on cada coordenada del vector representa una paraula del vocabulari. D'aquesta manera, donada una paraula, totes les posicions del seu vector valdran zero excepte la corresponent a la coordenada de la paraula.

És evident que els *one-hot vectors* són un tipus de *Word Embedding* extremadament ineficient, ja que es tracta de vectors amb una mida intractable tenint en compte que el vocabulari acostuma a ser de l'ordre de desenes de milers de paraules. A més a més, és una representació de les paraules que no aconsegueix captar en absolut les possibles proximitats semàntiques d'aquestes. Per tant, s'acostumen a utilitzar altres tècniques més sofisticades, anomenades models *word2vec*, com per exemple, el continuous bag of words [8, 9].

En el continuous bag of words, s'entrena una xarxa neural *Feed Forward* amb una única capa oculta totalment connectada amb la capa d'entrada i la capa de sortida. El nombre de neurones a la capa oculta és un hiperparàmetre que donarà la dimensionalitat dels *Word Embeddings*, mentre que el nombre de neurones a la capa d'entrada i de sortida serà igual al nombre de paraules al vocabulari. Aquesta xarxa neural s'entrenarà per tal de, donades les paraules del context com a entrada, predir la probabilitat que aparegui cada paraula del vocabulari. Un cop entrenada i per cada paraula, els pesos assignats a la seva neurona corresponent en la xarxa d'entrada formaran els valors del seu *Word Embedding*.

Actualment, ja existeixen moltes llibreries que proporcionen models entrenats que donen la codificació en *Word Embeddings* de les paraules en anglès.

Tot i això, és important recalcar que aquesta codificació permet guardar numèricament les relacions de les paraules, ja que per la mateixa naturalesa del conjunt d'entrenament, paraules amb significats semblants o de la mateixa família de substantius, tindran posicions properes en l'espai vectorial creat, de manera que es poden arribar fins i tot a identificar clústers de significats o calcular operacions aritmètiques entre paraules.

4.3.2 Positional Embeddings

Tot i que els models *word2vec* aconsegueixin representar les paraules en un espai on queden implícites les semblances entre aquestes, cada paraula continua tenint sempre la mateixa representació independentment del context i la seva posició dins d'una frase.

Per aquest motiu, s'utilitzen els *Positional embeddings*, que aconsegueixen donar rellevància a la posició. Simplement, consisteix a sumar un vector que depèn únicament de la posició de la paraula, al *Word Embedding* de la paraula. En el cas de l'estructura bàsica dels transformers, aquests vectors estan definits per les fórmules [7]:

$$PE_{(pos,2i)} = sin(\frac{pos}{1000^{\frac{2i}{d_{model}}}})$$

$$PE_{(pos,2i+1)} = cos(\frac{pos}{1000^{\frac{2i}{d_{model}}}})$$

on *pos* és la posició de la paraula dins la frase, d_{model} és el nombre de *Word Embeddings* a la seqüència i i és cada coordenada vectorial.

4.3.3 Encoder Multihead Self-Attention

El component de *Self-attention* és el component clau dels *Transformers*, ja que proporciona un mecanisme que deixa en evidència les interaccions entre cada parell de paraules proporcionades com a entrada del model. Es pot apreciar que aquestes interaccions no s'han processat fins aquest punt, pel fet que els *Word Embeddings* assignen un vector a cada paraula independentment del context, mentre que els *Positional Embeddings* únicament tenen en compte la posició de la paraula en el context.

Aquest mòdul retorna com a output tants vectors com els que hagi rebut d'entrada. Es pot entendre com un mòdul que, donats tots els vectors resultants de sumar els *Word Embeddings* als *Positional Embeddings*, mitjançant una sèrie de càlculs vectorials [7, 10] els modifica per tal que les relacions entre les paraules quedin implícites internament[2]:

1. Es realitzen projeccions lineals[3] sobre cada *Word Embedding* W_i de l'entrada per obtenir tres vectors diferents: *key* (K_i), *query* (Q_i) i *value* (V_i). Les matrius de

[2] Tot i que es parli d'operacions vector a vector de l'entrada, en la implementació real tots els vectors es tractaran simultàniament mitjançant càlculs matricials per qüestions de rendiment.

[3] Una projecció és una operació algebraica consistent a projectar un vector de dimensió n a un espai vectorial amb una dimensió inferior m. En el cas lineal, això es pot aconseguir multiplicant el vector per una matriu $n \times m$ amb uns valors determinats.

projecció que permeten aconseguir cadascun d'aquests vectors tindran pesos diferents, que permetin extreure diferents característiques del *Word Embedding* original.

2. Per cada parella de *Word Embeddings* W_i, W_j, es calcula un vector $S_{i,j}$ a partir dels seus vectors *key* i *query*, que deixa implícita la relació entre aquestes dues paraules a la sentència, utilitzant la funció $softmax$:

$$S_{i,j} = softmax(\frac{Q_i \cdot K_j^T}{\sqrt{d_{model}}})$$

on d_{model} és el nombre de *Word Embeddings* a la seqüència i $\cdot$ indica el producte escalar.

3. Per cada *Word Embedding* W_i, s'obté l'output final del self-attention O_i sumant tots els valors $S_{i,j}$ calculats en el pas anterior, i fent el producte escalar del resultat pel seu vector *value*:

$$O_i = (\sum_{1 \leq j \leq d_{model}, j \neq i} S_{i,j}) \cdot V_i$$

Tot i això, cal introduir el concepte de *multihead attention*, que consisteix a partir els vectors en diferents *heads* [7]. Posteriorment, es realitzen els càlculs descrits anteriorment per cada conjunt de *heads* per separat, utilitzant diferents pesos en les matrius de projecció en cada grup de *heads*, per finalment concatenar els resultats. Això permet que cada grup de *heads* es puguin enfocar en diferents tipus de dependències entre les paraules, donades pels pesos fets servir.

4.3.4 Decoder Multihead Self-Attention

Aquest component és similar a l'explicat a l'apartat anterior, però en aquest cas, tenint en compte que la sentència arriba seqüencialment al decoder, només es consideraran les posicions iguals o anteriors a la sentència en els càlculs, per tal de preservar la propietat autoregressiva[4] del model [7]. Això s'aconsegueix aplicant una màscara a les posicions futures, és a dir, fixant el seu valor a infinit abans d'aplicar el *softmax*.

4.3.5 Encoder-Decoder Multihead Attention

Aquest tipus de mòdul només es troba a les capes del decoder. Es tracta d'un mecanisme d'atenció similar al descrit a l'apartat *Encoder self-attention*. Però en aquest cas, les queries s'obtenen de la capa de decoder *multihead self-attention*, mentre que els *values* i *keys* s'obtenen de l'output de l'últim encoder [7].

4.3.6 *Feed Forward Neural Network*

Es tracta d'una xarxa neural *Feed Forward* convencional [11], que serà entrenada en la fase d'entrenament del *transformer*. Més concretament, en cada back propagation del *Transformer* es calcularà el back propagation de cada xarxa *Feed Forward*. Els seus hiperparàmetres són hiperparàmetres del propi *transformer*, però tenint en compte que es restringeix que totes les xarxes *Feed Forward* del model tinguin la mateixa estructura.

4.3.7 Residual Connection

Els *Transformers* utilitzen *Residual Connection* [12] entre l'entrada i la sortida de cada component d'*Attention* i *Feed Forward Neural Network*. La residual connection consisteix a sumar els vectors d'entrada amb els de sortida, per tal d'evitar que els vectors interns del model acabin tendint a valors infinits o a zero.

4.3.8 Normalize

Aquest mòdul realitza una normalització típica dels vectors, a partir de la seva mitjana i desviació típica [7].

4.3.9 Linear Layer i Softmax Layer

Aquestes dues capes finals són, respectivament, una xarxa neural *Feed Forward* [11] que transforma l'output del decoder en un vector de la dimensió del vocabulari, i un mòdul que aplica $softmax$ [13] a cada component d'aquest vector per tal de donar les probabilitats que cada paraula del vocabulari sigui la següent en la seqüència de sortida.

4.4 Models derivats

4.4.1 t5

El $t5$ és un model basat en *Transformers*, presentat a l'article *Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer* [14]. Principalment, es diferència dels transformers bàsics en tres aspectes:

- ***Position embeddings* relatius:** En comptes de calcular els *Posisiont Embeddings* amb les funcions sinusoidals presentades a la subsecció 4.3.2, ho fa mitjançant el que es coneix com a *Position Embeddings* relatius, que en comptes de sumar directament un valor fix a cada *word embedding*, afegeixen directament la informació de la posició als valors de les *keys* i les *queries* durant el càlcul del mecanisme d'atenció.
- **Normalització:** En el t5, la normalització s'aplica al principi de cada mòdul d'encoder i decoder, i al final de l'últim mòdul. A més, aquesta normalització únicament realitza un redimensionat, sense aplicar un biaix additiu.
- **Rendiment:** El t5 planteja diversos canvis menors respecte al transformer original per tractar de millorar el rendiment.

Addicionalment, es poden trobar fàcilment models preentrenats en la llengua anglesa convenients per a la tasca proposada.

4.4.2 Bart

Es tracta d'un model estàndard basat en *transformers* creat per part de $facebook$ [20]. Essencialment, és un model de transformer bàsic, tot i que utilitza una funció d'activació diferent en les *Feed Forward Neural Network*. El model ha estat preentrenat a partir de reconstruir text que ha estat corromput mitjançant una funció de soroll.

[4] Propietat referida a què la variable de sortida del model depengui de les sortides anteriors.

5 IMPLEMENTACIÓ

5.1 Entorn i llibreries utilitzades

La implementació d'aquest treball s'ha realitzat en un *Notebook* basat en el llenguatge *Python*. Més concretament, la implementació s'ha realitzat en l'entorn de *Google Colab Pro+*. El notebook és accessible en el següent repositori.

Pel que fa a les llibreries, principalment s'han utilitzat quatre llibreries de *Python* relacionades amb l'aprenentatge automàtic i el processament del llenguatge natural:

- **Pytorch:** És una de les llibreries d'aprenentatge automàtic més conegudes i importants, molt enfocada a l'aprenentatge profund d'alt rendiment. D'aquesta llibreria s'han utilitzat funcions i classes relacionades amb els *datasets*.
- **pytorch_lighting:** Es tracta d'una llibreria d'intel·ligència artificial d'alt nivell que fa de wrapper de la llibreria *Pytorch*, per tal de simplificar-ne l'ús sense perdre el total control i flexibilitat dels seus models.
- **transformers:** D'aquesta llibreria s'ha fet ús de la funció *AutoModelForSeq2SeqLM(model)*, que crea un model preentrenat que és una instància de l'arquitectura passada com a paràmetre. La llibreria baixa automàticament aquest model preentrenat de la web *Huggingface*. També s'ha fet servir la funció *AutoTokenizer(model)*, que crea la classe que prepara les entrades per al model seleccionat.
- **nlgeval:** S'ha usat aquesta llibreria per a calcular les mètriques del processament del llenguatge natural esmentades a la secció anterior, ja que proporciona funcions que implementen directament aquesta funcionalitat.

5.2 Base de dades

El *dataset sQuad* ja està dividit en conjunt d'entrenament, validació i test, i es troba en format *json*, que s'ha reestructurat per tal de simplificar l'estructura de les mostres. En particular, s'han eliminat les mostres sense resposta definida, i després de la reestructuració cada mostra del *dataset* conté únicament tres camps:

- **'answers':** Contindrà dos subcamps:
  - **'answer start':** Posició de la paraula que conté la resposta dins el context.
  - **'text':** Text de la resposta.
- **'context':** Context de la pregunta, és a dir, el paràgraf d'on s'extraurà.
- **'question':** Pregunta per a la resposta donada[5].

Per exemple, la primera mostra del *dataset* és:

- **'answers':**
  - **'answer_start':** 236
  - **'text':** 'cremated'
- **'context':** 'The Bagmati River which flows through Kathmandu is considered a holy river both by Hindus and Buddhists, and many Hindu temples are located on the banks of this river. The importance of the Bagmati also lies in the fact that Hindus are cremated on its banks, and Kirants are buried in the hills by its side. .'
- **'question'** 'What is done with Hindus after they die?'

El preprocessament de la base de dades, a més a més de reestructurar-ne la informació, realitza dues accions:

- Ressalta la resposta dins el context mitjançant els *Highlight tokens*..
- Tokenitza el context amb la pregunta ressaltada per tal de tenir directament preparada l'entrada de l'encoder.

5.3 Definició i entrenament del model

Les llibreries *transformers* i *pytorch_lightning* proporcionen classes i funcions que defineixen i entrenen el model automàticament. Per tant, l'únic que cal fer és la definició i implementació de classes derivades de les interfícies *LightningModule* i *LightningDataModule*, necessàries per a l'entrenament.

Pel que fa a la interfície *LightningModule*, s'ha definit la classe derivada *SQuADModel*, que proporciona els mètodes *training_step, forward, test_step* i *configure_optimizers*.

Per altra banda, s'ha definit la classe *SQuADDataModule* que deriva de la interfície *LightningDataModule*. Aquesta classe conté tres objectes de la classe *SQuADDataLoader*, un per cada divisió de la base de dades. La classe *SQuADDataLoader* és la que conté els mètodes pertinents per accedir als elements d'una base de dades.

6 RESULTATS

6.1 Mètode d'avaluació dels models

En aquest apartat es presenta de manera teòrica com s'ha dut a terme l'avaluació dels models. En particular, l'avaluació dels models s'ha fonamentat en diverses mètriques típiques del processament del llenguatge natural. Es tracta de mètriques que són capaces de proporcionar un valor que mesura com és de semblant una sentència a un conjunt de sentències de referència. Més concretament, aquestes mètriques s'utilitzaran per comparar numèricament com són de semblants les prediccions obtingudes amb el *ground truth*.

Per tant, per a mesurar la qualitat d'un model entrenat a partir dels conjunts d'entrenament i de validació, s'usarà el conjunt de test de la següent manera:

1. Per cada entrada del conjunt de test, es calcula la predicció del model, és a dir, el resultat de fer inferència sobre l'entrada.
2. Per cada entrada del conjunt de test, es calcula el *ground truth*, és a dir, el conjunt de preguntes que es poden considerar vàlides per a l'entrada donada.

   És important notar que en cada entrada del conjunt de test hi apareixen un context, una pregunta i un conjunt

[5] Com s'explicarà en la següent secció, en el cas del conjunt de test es considerarà una llista de preguntes, i no una de sola.

de respostes. En canvi, el que es necessita per a l'avaluació és un conjunt de preguntes. Per il·lustrar aquesta necessitat, si se suposa que l'entrada del conjunt de test és:

- **context**: *Gaius Julius Caesar was a Roman general and statesman. A member of the First Triumvirate, Caesar led the Roman armies in the Gallic Wars before defeating his political rival Pompey in a civil war, and subsequently became dictator of Rome from 49 BC until his assassination in 44 BC.*
- **answer**: *Gaius Julius Caesar*.
- **question**: *Who led the Roman armies in the Gallic Wars?*.

Però la pregunta predita pel model entrenat ha estat:

- **output_question**: *Who became dictator of Rome in the year 49 BC?*

Es pot veure que el *ground truth* i la pregunta predita tenen un nivell de semblança mínim i, per tant, qualsevol mètrica que les compari donarà un valor baix a la predicció. Però qualitativament, és clar que la pregunta predita és del tot correcte.

Així, amb l'objectiu de pal·liar aquest tipus de fet, el que s'ha fet és treballar amb un conjunt de preguntes com a *ground truth*. El mètode per obtenir aquest conjunt de preguntes per a una entrada donada del conjunt de test, consisteix a recórrer tot el *dataset* buscant altres entrades que tinguin la mateixa resposta i el mateix context, i en cas de trobar-ne, la seva pregunta s'afegeix al *ground truth*.

3. Un cop calculada una predicció i un *ground truth*, es calcula el valor de la mètrica entre elles.

4. Es realitza aquest procés per cada entrada del conjunt de test, i mitjançant la mitjana aritmètica de totes les mètriques obtingudes, s'obté la mesura de qualitat del model.

Les següents subseccions presenten de manera teòrica les diferents mètriques utilitzades que permeten mesurar el nivell de semblança entre una sentència i un conjunt de sentències de referència. Per evitar soroll i diferències innecessàries entre les sentències, es treballarà amb aquestes en minúscules, sense signes de puntuació i sense articles. Totes les mètriques prenen valors a l'interval real $[0, 1]$, on el valor 1 indica una predicció perfecta, mentre que el valor 0 indica una predicció completament errònia.

6.1.1 Mètrica 1: Bleu score

La *BLEU Score* [16] és una mètrica basada en el concepte de $n-grams$. Els $n-grams$ són subconjunts de n paraules seguides dins d'una frase.

Exactament, el que fa aquesta mètrica és, donada una n, compta quants cops es repeteix cada $n-gram$ a la frase candidata. Sigui k el nombre de $n-grams$ diferents que té la frase candidata, siguin $count_1, count_2, \ldots, count_k$ el nombre de vegades que es repeteix cada $n-gram$ de la frase candidata, i siguin $ref_{1,j}, ref_{2,j}, \ldots, ref_{k,j}$ el nombre de vegades que es repeteix cada $n-gram$ de la frase candidata a la j-èssima frase de referència, i sigui m el nombre de sentències de referència. Llavors, es calcula la proporció p_n de vegades que els $n-grams$ de la frase candidata apareixen a la resta de frases, de la següent manera:

$$p_n = \frac{\sum_{i=1}^{k} min(count_i, max_{j \in \{1,\ldots,m\}} ref_{j,i})}{\sum_{i=1}^{k} count_i}$$

On el mínim del numerador s'aplica per tal d'evitar que un $n-gram$ no pugui comptar més cops a la proporció que el màxim de cops que apareix en alguna frase de referència.

Aquest procediment es repeteix per cada n desitjada individualment, i posteriorment s'aplica una mitjana geomètrica dels resultats obtinguts per cadascuna.

Per altra banda, s'aplica una penalització per perjudicar a les sentències massa breus. Sigui c la longitud en paraules de la sentència candidata i sigui r la longitud en paraules de la sentència de referència més curta, llavors es defineix la penalització BP com:

$$BP = \begin{cases} 1 & if\ c > r \\ e^{1-\frac{r}{c}} & if\ c \leq r \end{cases} .$$

Finalment, si es consideren els $n-grams$ per a $n \in \{1, 2, \ldots, N\}$, i si $w_1, w_2, \ldots, w_N$ són els pesos sobre la mètrica per a cada n, la mètrica es calcula com:

$$BLEU = BP \cdot e^{\sum_{n=1}^{N} w_n log(p_n)}$$

6.1.2 Mètrica 2: METEOR

En el cas de la mètrica METEOR [17], a diferència del que es feia en la mètrica anterior, la frase candidata es compara individualment amb cada sentència de referència. Un cop fetes totes les comparacions, el valor final de la mètrica serà el màxim de tots els valors obtinguts.

Aquesta mètrica compara la frase candidata amb una frase de referència mitjançant un mapeig de paraules coincidents. Per a realitzar aquest mapeig, primer busca paraules que siguin exactament iguals a les dues sentències, després busca paraules morfològicament relacionades, i finalment busca sinònims.

Un cop mapejades totes les paraules coincidents d'una frase amb l'altra, calcula el valor de la F_1-score. Tot i que aquest valor només ha tingut en compte el mapeig entre paraules de les dues sentències, i és independent de l'ordre en què aquestes apareguin. Per tant, per a reflectir la importància de l'ordre de les paraules en el mapeig realitzat, s'aplica una penalització Pen, donada per

$$Pen = 0.9 \cdot frag^3$$

On els valors numèrics van ser determinats per experimentació [17] i $frag$ és la proporció del nombre de cadenes de paraules mapejades que són adjacents a les dues sentències entre el nombre total de paraules mapejades. Amb tot, el valor de la mètrica és definit per

$$METEOR = (1 - Pen) \cdot F_1$$

Model	Bleu 1	Bleu 2	Bleu 3	Bleu 4	METEOR	ROUGE_L	exact match
t5-small	58.98	46.48	35.32	26.78	37.66	61.06	10.32
t5-base	60.66	48.69	37.84	29.09	39.98	62.85	11.63
t5-efficient-base-n14	58.03	45.39	34.19	25.73	36.59	60.32	9.63
bart	47.72	36.38	27.01	19.36	28.28	51.87	0.23

Taula 1: Mètres obtingudes per a cada model entrenat.

6.1.3 Mètrica 3: ROUGE_L

Com en la mètrica anterior, la mètrica ROUGE_L [18], compara la sentència candidata amb cada sentència de referència individualment, i després tria el màxim valor obtingut com a valor final de la mètrica.

Rouge són les sigles de *Recall-Oriented Understudy for Gisting Evaluaion*, i es tracta d'un conjunt de mètriques diferents. En el cas de la ROUGE_L, la L fa referència a *Longest Common Subsequence*, ja que el que fa és buscar la subseqüència més llarga comuna entre la sentència candidata i la sentència de referència. A partir d'aquesta subseqüència, es calcula el valor de la mètrica com la proporció entra la longitud de la subseqüència trobada i la longitud de la frase de referència.

6.1.4 Mètrica 4: Exact match

Aquesta mètrica val 1 si la sentència donada és exactament igual a alguna de les sentències del conjunt de referència, i en cas contrari, val 0.

6.2 Anàlisi dels resultats obtinguts

Un cop realitzada la implementació del projecte, gràcies a l'abstracció donada per la classe *Trainer* de *pytorch_lightning*, es pot canviar l'arquitectura de *transformer* utilitzada només amb la modificació del paràmetre que determina quin model preentrenat utilitzar.

Així, s'ha utilitzat el projecte per a entrenar i avaluar quatre models preentrenats diferents:

- **t5-small:** Es tracta de la versió reduïda del $t5$, que té aproximadament 60 milions de paràmetres.
- **t5-base:** És la versió bàsica del $t5$, que té aproximadament 220 milions de paràmetres.
- **t5-efficient-base-nl4:** És una versió modificada del $t5$, que té aproximadament 90 milions de paràmetres. La modificació es troba en l'arquitectura de les xarxes *feed forward* del model, que en aquesta versió s'han definit seguint una estratègia *Deep-Narrow*, és a dir, xarxes amb poques capes, però on cada capa tingui una gran quantitat de nodes [19].
- **facebook/bart-base:** Es tracta de la versió bàsica del $bart$, que té aproximadament 139 milions de paràmetres.

La taula 1 mostra les mètriques obtingudes per cadascun dels models en forma de percentatge. Les mètriques *Bleu 1*, *Bleu 2*, *Bleu 3* i *Bleu 4* fan referència a la *Bleu-score* tenint en compte fins a 1, 2, 3 i $4 - grams$ respectivament.

6.2.1 Anàlisi quantitatiu dels resultats

Quantitativament, es pot dir que tots els models han obtingut resultats raonables i equiparables a altres projectes de l'estat de l'art entrenats a partir del mateix *dataset*, com per exemple els obtinguts a [3]. De fet, en aquest projecte les mètriques assolides han donat valors més alts que en les de la bibliografia, tot i que això probablement es deu al fet que en altres projectes s'acostuma a utilitzar el *dataset SQuAD* directament, amb una única pregunta de referència per entrada, i això provoca que les mètriques donin valors més baixos als aconseguits amb una avaluació amb més referències.

Per altra banda, s'observa que en general, el model $bart$ assoleix resultats significativament pitjors que totes les variants considerades del $t5$, per a totes les mètriques. A més, com era d'esperar per la diferència en el nombre de paràmetres, el model $t5 - base$ obté millors mètriques que les altres dues variants del model $t5$.

Finalment, els models $t5 - small$ i *t5-efficient-base-n14* obtenen mètriques molt properes, especialment pel que fa a la *Bleu-score*, tot i que el $t5 - small$ obté mètriques lleugerament superiors. Tenint en compte que a més a més té menys paràmetres, el model $t5 - small$ és una millor tria.

La *Bleu-score* i la *METEOR* s'acostumen a utilitzar per avaluar models traductors, mentre que la *ROUGE_L* s'acostuma a utilitzar per a models generadors de resums. Per tant, tot i que és adient tenir en consideració la combinació de totes les mètriques, es pot pensar que la *ROUGE_L* és la més adequada per al tipus de model considerat, ja que una pregunta no deixa de ser un tipus de resum.

Per tot això, és clar que el model final triat ha de ser el $t5 - base$. Tot i que té l'inconvenient de la seva gran mida, que provoca un entrenament de desenes d'hores i una ocupació de memòria prou gran. Per tant, el model $t5 - small$ seria una alternativa adient.

6.2.2 Anàlisi qualitatiu dels resultats

Un cop triat el model $t5 - base$ com a model final, s'ha fet inferència sobre aquest per tal de realitzar un petit nombre d'experiments que permetin una anàlisi qualitativa dels resultats.

La taula 2 presenta un conjunt d'exemples de prediccions obtingudes per inferència que s'han considerat interessants per a l'anàlisi qualitativa. Aquests exemples permeten extreure diverses conclusions:

- Els exemples 1, 2, 3 i 4 mostren que el model és capaç de generar preguntes amb sentit i completament correctes per a una entrada senzilla. De fet, l'exemple 3 mostra que el model és fins i tot capaç de generar preguntes amb una resposta implícita, és a dir, una resposta que no apareix al context directament.

Exemple	Entrada		Output
	Context	Resposta	
1	After ten years of training, the french athlete won the 2018 world championships.	the french athlete	Who won the 2018 world championships?
2	After ten years of training, the french athlete won the 2018 world championships.	ten years	How many years of work did the athlete have to do?
3	After ten years of training, the french athlete won the 2018 world championships.	Because he trained for ten years	Why did he win the 2018 world championships?
4	After ten years of training, the french athlete won the 2018 world championships.	ten years of training	How long did it take for the French athlete to win the 2018 world championships?
5	The athlete had been training for a long period. During this time, he suffered from hungry. This period lasted ten years.	ten years of training	How long had he been training?
6	The older sister is 38 years old, while the younger sister is 25.	25	How old is the younger sister?
7	The older sister is 38 years old, while the younger one is 25.	25	How old is the older sister?
8	Werner Karl Heisenberg was a scientist.	Werner Karl Heisenberg	Who was a scientist?
9	Albert Einstein was a scientist.	Albert Einstein	What scientist was a scientist?
10	The UK prime minister has spent the afternoon talking to his ministers.	The UK prime minister	Who has spent the afternoon talking to his colleagues?
11	This fella has spent the noon talking with his buddies.	This fella	Who has spent the afternoon talking with his buddies?
12	Lad chit chat noon with the lads.	Lad	What do the lads do?

Taula 2: Exemples de prediccions obtingudes per inferència.

A més, comparant els exemples 2 i 4, es pot veure que el model és capaç de generar preguntes amb un cert grau d'exactitud, ja que per a dues respostes molt semblants en el mateix context, el model proporciona dues preguntes diferents amb un bon nivell de detall.

- L'exemple 5 reflexa que el model és capaç de predir una pregunta qualitativament correcte quan la resposta es troba repartida en sentències diferents del context, quan a més a més hi ha sentències sense cap rellevància intercalades.

- Els exemples 6 i 7 reflecteixen que en algunes situacions el model no identifica bé quin és el subjecte relacionat amb la resposta quan n'hi ha més d'un. D'aquesta manera, s'observa que a l'exemples 6 el model no té cap problema per diferenciar quan la resposta es refereix a la *younger sister*. En canvi, en l'exemple 7, on els subjectes són més difícils d'identificar, ja que ara es parla de la *younger one*, el model no ha sigut capaç d'identificar a quina germana es refereix la resposta.

- Els exemples 8 i 9 presenten un cas especial en què el model no genera una pregunta adequada. Els dos contexts són idèntics, però variant el subjecte de la frase. En aquest cas, s'utilitza el nom de dos científics àmpliament coneguts.

  En el cas de *Heisenberg*, es genera una pregunta qualitativament correcte. En canvi, en el cas d'*Albert Einstein*, es genera una pregunta sense gaire sentit i que s'autorespon, pel fet que el model ha identificat que quan la resposta a una pregunta està formada per les paraules *Albert Einstein*, la pregunta corresponent estarà preguntant per un científic, i no per una persona. Això es deu al fet que el conjunt d'entrenament conté més de 150 entrades relacionades amb *Albert Einstein*. Per tant, el model està massa especialitzat en aquest cas i es pot considerar un error d'*overfitting*.

- Els exemples 10, 11 i 12 permeten veure com es comporta el model per a diferents registres de formalitat. En el cas de l'exemple 10, escrit en un registre prou formal, el model genera una pregunta qualitativament correcte. El mateix passa en la pregunta 11, escrita en un registre més informal, però guardant una estructura gramatical estàndard.

  En canvi, el model no és capaç de generar una pregunta amb sentit per a l'entrada de l'exemple 12, que està escrit en un llenguatge extremadament informal. Això té sentit, ja que el model s'ha entrenat a partir d'una base de dades extreta de *Wikipedia*, que conté textos que acostumen a estar en un llenguatge formal.

7 CONCLUSIONS

El projecte elaborat ha assolit amb èxit els objectius proposats, excepte l'objectiu de crear una pipeline amb un *OCR* que sigui capaç de generar preguntes sobre la imatge d'un text. A més, el projecte ha proporcionat les següents aportacions:

- S'han creat i avaluat diferents models de *Deep Learning* a partir de models preentrenats de l'estat de l'art.

- S'ha obtingut un model fiable capaç de generar preguntes a partir d'un context i una resposta donada.

- S'han presentat diferents mètriques típiques del processament del llenguatge natural, i s'ha exposat la importància de tenir diferents referències per a cada entrada del conjunt d'entrenament.
- El projecte ha servit d'introducció a l'autor en la disciplina de l'aprenentatge profund i el processament del llenguatge natural.

7.1 Possibles ampliacions

Com a possibles ampliacions per a aquest projecte, es proposen:

- Utilitzar diferents *datasets* o models amb els meteixos requisits de còmput, i veure quin efecte tenen sobre els resultats obtinguts. Per exemple, en relació amb el registre de les preguntes generades.
- Entrenar i avaluar models amb més paràmetres en computadores amb més potència de còmput.
- Entrenar i avaluar un model per a altres llengües. Per exemple, utilitzant les versions espanyoles del *t5* i l'*SQuAD dataset*.
- Construir un pipeline que permeti generar preguntes a partir d'imatges d'un text, mitjançant un *OCR*.
- Combinar el model amb un model resumidor de textos per tal de generar automàticament respostes i preguntes a partir d'un text.

AGRAÏMENTS

Agraeixo al meu tutor acadèmic, Ernest Valveny Llobet, pel seu seguiment i guia al llarg de tot el desenvolupament del projecte. També agraeixo a la Claudia Uranga Alonso i en Marc Gonzalez Motlló per la seva revisió i consells sobre la redacció de l'article.

REFERÈNCIES

[1] P. Johri, S. K. Khatri, A. T. Al-Taani, et al. "Natural Language Processing: History, Evolution, Application, and Future Work ". Proceedings of 3rd International Conference on Computing Informatics and Networks, pp. 365-375, 2021.

[2] R. Zhang, J. Guo, L. Chen, Y. Fan, X. Cheng. "A Review onQuestion Generation from Natural Language Text". ACM Trans. Inf. Syst. 40, 1, Article 14, 2021.

[3] K. Kriangchaivech, A. Wangperawong. "Question Generation by Transformers". arXiv:abs/1909.05017, 2019.

[4] P. Rajpurkar, J. Zhang, K. Lopyrev, et al. "SQuAD: 100, 000+ questions for machine comprehension of text". Proceedings of the 2016 Conference on Empirical Methods in NLP, pp. 2383-2392, 2016 .

[5] Adam Trischler, T. Wang, Xingdi Yuan, et al. "NewsQA: A machine comprehension dataset". Proceedings of the 2nd Workshop on Representation Learning for NLP, pp. 191-200, 2017.

[6] M. Dunn, Levent Sagun, M. Higgins, et al. "SearchQA: A new QA dataset augmented with context from a search engine ". arXiv:abs/1704.05179, 2017.

[7] A. Vaswani, N. Shazeer, N. Parmar, et al. "Attention is All you Need ". Advances in Neural Information Processing Systems, pp. 5998–6008, 2017.

[8] T. Mikolov, K. Chen, G. Corrado, J. Dean. "Efficient estimation of word representations in vector space". arXiv preprint arXiv:abs/1301.3781, 2013.

[9] Q. Wang; J. Xu; H. Chen; B. He. "Two improved continuous bag-of-word models". International Joint Conference on Neural Networks (IJCNN), pp. 2851-2856, 2017.

[10] M. Luong, H. Pham , C. D. Manning. "Effective Approaches to Attention-based Neural Machine Translation". Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pp. 1412-1421, 2015.

[11] M. Sazli. "A brief review of feed-forward neural networks". Communications Faculty of Sciences Univer sity of Ankara Series A2-A3 Physical Sciences and Engineering 50, pp. 11-17, 2006.

[12] K. He, X. Zhang, S. Ren, J. Sun. "Deep residual learning for im- age recognition". In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778, 2016.

[13] C. M. Bishop. "Pattern Recognition and Machine Learning". Springer. ISBN 0-387-31073-8, 2006.

[14] C. Raffel, N. Shazeer, A. Roberts, et al. "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer". arXiv:abs/1910.10683, 2019.

[15] D. P. Kingma, J. Ba. "Adam: A Method for Stochastic Optimization". 3rd International Conference on Learning Representations, ICLR 2015, 2015.

[16] K. Papineni, S. Roukos, T. Ward. "BLEU: a Method for Automatic Evaluation of Machine Translation". Proceedings of the 40th Annual Meeting of the ACL, pp. 311-318, 2002.

[17] A. Lavie, A. Agarwal. "METEOR: An automatic metric for MT evaluation with high levels of correlation with human judgments". Proceedings of the Second Workshop on Statistical Machine Translation. pp. 228-231. 2007.

[18] C.Y. Lin. "ROUGE: A Package for Automatic Evaluation of Summaries". Text Summarization Branches Out, pp. 74–81. 2004.

[19] Y. Tay, M. Dehghani, J. Rao, et al. "Scale Efficiently: Insights from Pre-training and Fine-tuning Transformers". arXiv:abs/2109.10686, 2021.

[20] M.Lewis, Y. Liu, N. Goyal, et al. "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension". ACL 2020, pp. 7871-7880, 2020.

APÈNDIX

A.1 Figures il·lustratives dels *transformers*

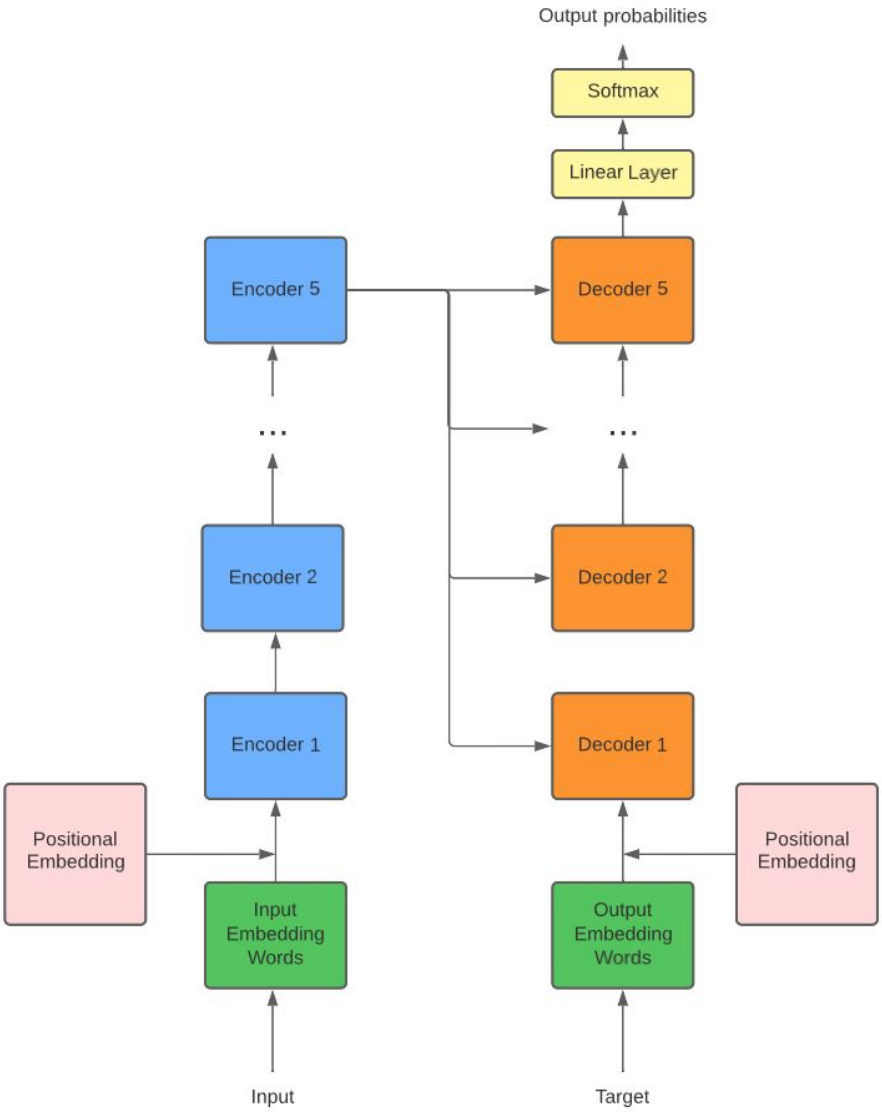

Fig. 2: Arquitectura típica d'un *transformer*.

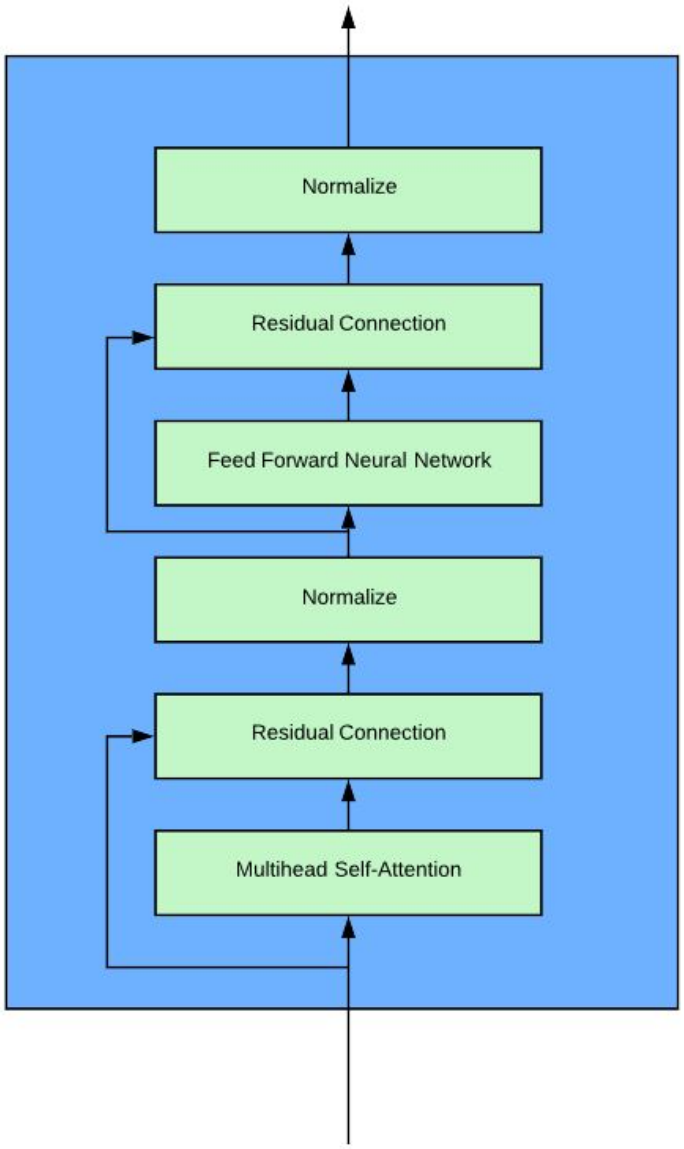

Fig. 3: Arquitectura típica d'una capa de l'encoder.

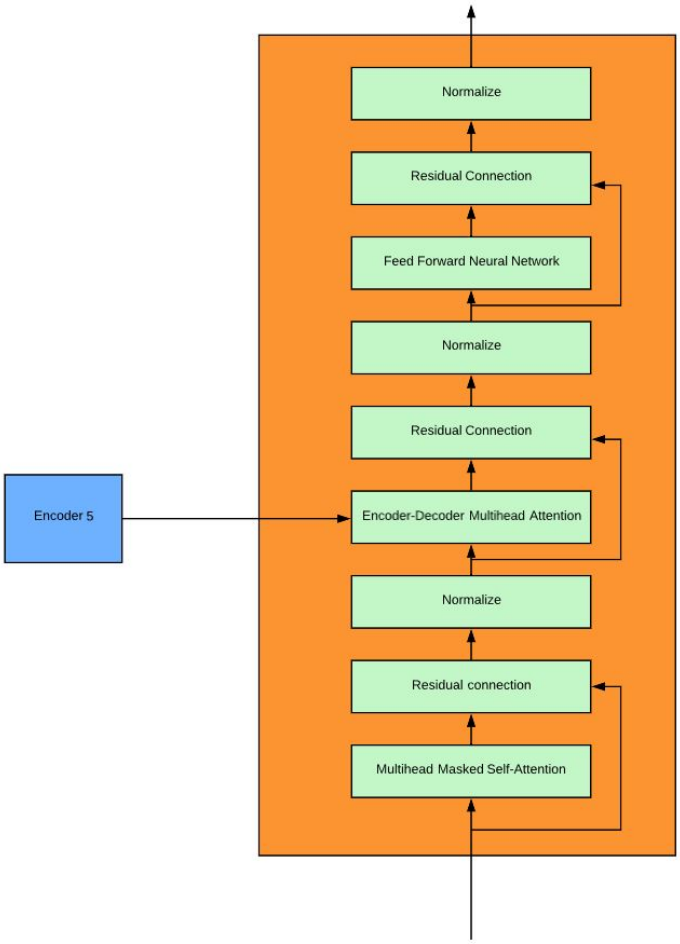

Fig. 4: Arquitectura típica d'una capa del decoder.