UNIVERSITAT DE BARCELONA

Treball final de grau

GRAU DE MATEMÀTIQUES

Facultat de Matemàtiques i Informàtica
Universitat de Barcelona

Alineació de paraules i mecanismes d'atenció en sistemes de traducció automàtica neuronal

---

Autor: Pol Safont Gascón

Director: Dr. Daniel Ortiz Martínez
Realitzat a: Departament de matemàtiques i informàtica

Barcelona, 24 de gener de 2022

Abstract

Deep Neural Networks have become the state of the art in many complex computational tasks. While they achieve great improvements over several benchmarking tasks year after year, they seem to operate as black boxes, making it hard for both data scientist and end users to assess their inner decision mechanisms and trust their results.

While statistical and interpretable methods are widely used to analyze them, they don't fully grasp their internal mechanisms and are prone to misleading results, leading to a need for better tools. As a result, self-explaining methods embedded inside the architecture of the neural networks have become a possible alternative, with attention mechanisms as one of the main new technics.

The project main focus is the word alignment task, finding the most relevant translation relationships between source and target words in a pair of parallel sentences in different languages. This is a complex task of the *Natural Language Processing* and machine translation field, and we analyze the use of the novel attention mechanisms embedded in different encoder-decoder neural networks in order to extract the word to word alignments between source and target translations as a byproduct of the translation task.

In the first part we analyze the background of the machine translation field: the main traditional statistical methods, the neural machine translation approach to the sequence to sequence problem and finally the word align task and the attention mechanism. In the second part, we implement a machine translation deep neural networks model: a recurrent neural network with an encoder-decoder architecture with attention. And we propose an alignment generation mechanism using the attention layer in order to extract and predict source to target word to word alignments. Finally, we train the neural networks with an English and French bilingual parallel sentence corpus and analyze the experimental results of the model for the translation and align word to word tasks, using a variety of metrics and suggest improvements and alternatives.

Resumen

Las redes neuronales profundas se han convertido en el estado del arte en muchas tareas computacionales complejas. Mientras año tras año consiguen grandes mejoras respecto a muchas tareas de referencia, parecen funcionar como cajas negras, lo que dificulta tanto a los científicos de datos como a los usuarios finales evaluar sus mecanismos de decisión internos y confiar en sus resultados.

Aunque métodos estadísticos e interpretables son usados ampliamente para analizarlas, no logran abarcar completamente sus mecanismos internos y son propensos a resultados engañosos, lo que lleva a la necesidad de desarrollar mejores herramientas. Por ello, los métodos autoexplicativos incorporados a la arquitectura de las redes neuronales se han convertido en una posible alternativa, siendo los mecanismos de atención una de las principales novedades técnicas.

El proyecto se centra en el problema de alineación de palabras, que consiste en encontrar las relaciones de traducción más relevantes entre las palabras de origen y las de destino en un par de frases paralelas en diferentes idiomas. Se trata de una tarea compleja del campo del procesamiento del lenguaje natural y la traducción automática, y analizamos el uso de los novedosos mecanismos de atención integrados en diferentes redes neuronales codificador-decodificador con el fin de extraer las alineaciones palabra a palabra entre las traducciones de origen y destino como subproducto de la tarea de traducción.

En la primera parte analizamos los antecedentes en el campo de la traducción automática: los principales métodos estadísticos tradicionales, el enfoque de la traducción automática neuronal al problema de secuencia a secuencia y, finalmente, la tarea de alineación de palabras y el mecanismo de atención. En la segunda parte, implementamos un modelo de red neuronal profunda de traducción automática: una red neuronal recurrente con una arquitectura codificador-decodificador con atención. Y proponemos un mecanismo de generación de alineaciones utilizando la capa de atención para extraer y predecir las alineaciones entre palabras de origen y destino. Finalmente entrenamos las redes neuronales con un corpus de oraciones paralelas bilingües en inglés y francés y analizamos los resultados experimentales del modelo para las tareas de traducción y alineación palabra a palabra, empleando una variedad de métricas y sugerimos mejoras y alternativas.

Resum

Las xarxes neuronals profundes s'han convertit en l'estat de l'art en moltes tasques computacionals complexes. Encara que any rere aconsegueixen grans millores respecte moltes tasques de referencia, semblen operar com caixes negres, dificulten tant als científics de dades com als usuaris finals avaluar els seus mecanismes de decisió interna i confiar en els seus resultats.

Tot i utilitzar mètodes estadístics i interpretables per analitzar-les, aquests no aconsegueixen capçar completament els seus mecanismes interns i solen ser propensos a resultats equívocs, el que porta a la necessitat de desenvolupar millors eines. En aquest sentit, els mètodes auto-explicatius incorporats a la arquitectura de les xarxes neuronals s'han convertit en una alternativa de futur, amb els mecanismes d'atenció com una de les principals novetats tècniques.

Aquest projecte es centra en el problema d'alineament de paraules, que consisteix en trobar les relacions de traducció mes rellevants entre les paraules d'origen i las de destí en un parell de frases paral·leles en diferents idiomes. Es tracta d'una tasca complexa del camp del processament del llenguatge natural i la traducció automàtica, i analitzem l'ús del innovadors mecanismes d'atenció integrats en diferents xarxes neuronals codificador-decodificador amb la finalitat d'extreure les alineacions paraula a paraula entre les traduccions d'origen i destí com un subproducte de la tasca de traducció.

En la primera part del projecte analitzem els antecedents en en el camp de la traducció automàtica: els principals mètodes estadístics tradicionals, l'enfocament de la traducció automàtica neuronal al problema de seqüencia a seqüencia i finalment la tasca d'alineació de paraules i el mecanisme d'atenció. En la segona part, implementem un model de xarxa neuronal profunda de traducció automàtica: una xarxa recurrent amb una arquitectura codificar-decodificador amb atenció. I proposem un mecanisme de generació d'alineacions entre paraules d'origen i destí. Finalment entrenem les xarxes neuronals amb un corpus d'oracions paral·leles bilingües en anglès i francès i analitzem els resultats experimentals del model per les tasques de traducció i alineació paraula a paraula emprant una varietat de mètriques i suggerim millores i alternatives.

Agraïments

Vull agrair a la meva parella i a la meva família pel seu suport.

I a la Universitat de Barcelona i al Dr.Daniel Ortiz per la seva paciència.

Índex

1 Introducció

La intel·ligència artificial cada cop es més present en el dia a dia de la nostra societat. Les noves tècniques d'aprenentatge automàtic en conjunt amb les xarxes neuronals artificials, l'abundància de dades i els avenços en computació paralel·la han impulsat una revolució en múltiples àmbits tan diversos com el reconeixement facial, la conducció automàtica, la generació de continguts o la predicció meteorològica.

En el camp del processament del llenguatge natural (*Natural Language Processing*, NLP) i concretament en la tasca de la traducció automàtica (Machine Translation, MT) l'impacte de les xarxes neuronals artificial (Artificial Neural Networks, ANN) i en concret les xarxes neuronals profundes (*Deep Neural Networks*) organitzades en múltiples capes conjuntament amb les tècniques de d'aprenentatge profund (Deep Learning, DL) que utilitzen han significat un canvi de paradigma respecte els models tradicionals de traducció automàtica estadística (Stadistical Machine Translation, SMT).

Actualment la investigació capdavantera en l'àmbit de la traducció automàtica s'ha enfocat envers els models de traducció automàtica neuronal (Neural Machine Translation, NMT)[18]. La primera ANN en sobrepassar els models SMT va ser les xarxes neuronals recurrents (Recurrent Neural Networks, RNN) enfocades al processament de dades seqüencia a seqüencia proposada per primer cop en l'article 'Sequence to Sequence Learning with Neural Networks' del 2014. [34].

En aquest context el mateix any es va publicar un mecanisme novel en les ANN anomenat atenció en l'article 'Generating Sequences With Recurrent Neural Networks' [13] i popularitzat en 'Neural Machine Translation by Jointly Learning to Align and Translate' [3] del mateix any.

Al 2017 l'arquitectura *transformer* proposada a l'article 'Attention is all you need' [36] va tornar a revolucionar el camp. Basada totalment en els mecanismes d'atenció i abandonant l'arquitectura RNN que havia dominat fins el moment el NMT.

Al mateix any en el article 'Six challenges for Neural Machine Translations' [18] es fa un review dels principals reptes que afronta aquest nou paradigma envers el SMT: baixa qualitat de traducció quan es troben fora del domini on han sigut entrenats els models, necessiten grans quantitat de dades a canvi de petites millores, dificultats en al traducció correcte de paraules rares, traduccions de pitjor qualitat lligades a frases llargues, alineaments de paraules no sempre correctes i les limitacions del beam search per millorar les traduccions.

Finalment es fa una septima observació: els models de NMT son molt menys interpretables que els SMT. El perquè un model selecciona una paraula especifica a l'hora de fer una traducció esta amagat entre grans matrius de nombres reals. Existeix la necessitat de crear millor mètodes analítics per el camp de NMT.

1.1 Motivació

Actualment els models d'aprenentatge profund son entesos principalment com a caixes negres on es difícil si no impossible analitzar degudament la lògica interna de les xarxes neuronals. Els models emprats en multitud de tasques son opacs, estan des-regulats i són difícils de rebatre tant per part dels usuaris com per els investigadors [25]. Per altre banda el creixement exponencial de dades d'entrenament i paràmetres interns dels models compliquen encara mes el seu anàlisis.

Davant aquesta problemàtica els esforços per interpretar el funcionament i els resultats obtinguts s'han centrat en mètodes estadístics aixi com interpretacions a posteriori de inferències concretes de models ja entrenats. Davant la necessitat de crear mecanismes explícits, fiables i estables [2] els mètodes estadístics son difícilment entenedors per usuaris sense formació matemàtica especialitzada. Per altre banda les explicacions interpretatives a posteriori, centrades en l'aproximació de les relacions entre les entrades i les sortides amb regles senzilles, encara que aconsegueixen aproximar els resultats per casos concrets no capturen correctament els mecanismes d'aprenentatge i decisió interns de les xarxes neuronals profundes. La evidencia porta a pensar que les ANN entrenades amb DL aprenen interpolant suaument entre punts de dades en comptes de extraure regles d'alt nivell dels conjunts de dades.

Davant d'aquest problema sorgeix el concepte de intel·ligències artificials auto-explicables o self-explaining AI [11], que retorna prediccions acompanyades d'una explicació interna, per tal d'incrementar la confiança tant en els resultats del model com per millorar la supervisió humana del mateix davant de possibles errors.

En el camp de la traducció automàtica això es pot portar a terme en gran mesura gracies als mecanismes d'atenció interns de les xarxes [3]. Aquestes aprenen no només a traduir una seqüencia d'entrada a una seqüencia de sortida de forma supervisada, si no també a prestar diferent atenció als diversos elements de la seqüencia d'entrada al mateix temps. Aixo permet aprendre a realitzar la tasca d'alineament de forma no supervisada a la vegada que la traducció. Aquestes alineacions permeten una auto-explicació amb certa confiança que acompanya els resultats de la traducció i permet un anàlisis tant estadístic com purament lingüístic de la lògica interna del model i de la relació entre les seqüencies d'entrada i de sortida.

1.2 Objectius

El objectiu principal d'aquest treball es iniciar-se en el camp de la traducció automàtica i el processament del llenguatge natural des de una perspectiva centrada en les xarxes neuronals.

En segon terme esta la investigació i el desenvolupament de models basats en les tècniques capdavanteres en el camp, obtenir les traduccions inferides aixi com els alineaments paraula a paraula com a resultat de mecanismes auto-explicatius generats internament. En aquest sentit desenvoluparem una xarxa neuronal recurrent amb una capa d'atenció.

Un cop decidida la arquitectura interna passarem a la fase d'entrenament dels models i analitzar i avaluar els resultats obtinguts. Per dur a terme l'entrenament caldrà escollir en primer lloc com a conjunt de dades d'entrenament un corpus bitext alineat a nivell de frases i en segon lloc un conjunt de dades de test amb alineament tant a nivell de sentencia com a nivell de paraules. Un cop entrenades escollirem diverses mètriques adients als dos problemes, traducció i alineament, Amb els resultats obtinguts analitzarem els resultats de les xarxes neuronals, els compararem amb els resultats dels sistemes de SMT i NMT capdavanters i extraurem conclusions sobre la utilitat i precisió d'aquests sistemes.

1.3 Planificació

Per tal d'aconseguir els objectius marcats, caldrà ampliar els coneixement adquirits en les diverses assignatures relacionades amb NLP i aprenentatge automàtic, iniciar l'estudi del camp de traducció automàtica i aprofundir en el camp de la traducció automàtica neuronal.

Per altre banda, caldrà familiaritzar-nos amb el framework de desenvolupament de deep learning escollit PyTorch i amb eines auxiliars del camp del NLP.

Finalment tenim el problema d'accés a una sistemes computacionals en paral·lel per tal de portar a terme l'entrenament de les xarxes neuronals. Mitjançant l'ús de les plataformes de computació en el nuvol de google Colab i Kaggle tindrem accés limitat a GPUs (Graphical Processing Unit) i TPUs (Tensor Processing Unit) d'última generació per entrenar de forma intensiva els models amb grans volums de dades.

Planificació temporal del treball

Octubre:

1. Familiaritzar-mos amb l'entorn de treball escollit pel desenvolupament: el framework d'aprenentatge automàtic i deep learning Pytorch pel desenvolupament i els entorns de execució virtuals de Google Colab i Kaggle basats en Jupyter Notebook.

2. Lectura fonaments del deep learning, traducció automàtica i el problema d'alineament de paraules.

3. Lectura sobre Neural Machine Translation, especialment sobre xarxes neuronals recursives i mecanismes d'atenció, i *transformers*. Recopilació d'articles,llibres, fonts d'informació, tutorials i exemples.

4. Cerca de conjunt de dades adients al problema i de mesures de qualitat de la traducció i del problema de alineament paraula a paraula.

Novembre:

5. Implementació dels models de RNNs + Attention i *Transformers*.

6. Depuració dels models, generació de traduccions i implementació de force decoding.

Desembre:

7. Implementació de la funcionalitat de generació de heatmaps a partir de les atencions internes dels models.

8. Implementació de la funcionalitat per obtenir matrius d'alineament a nivell de paraula a partir dels heatmaps.

9. Implementació d'eines de visualització de les matrius d'atenció, dels heatmaps i les matrius d'alineament.

10. Comparació de la qualitat de les traduccions i dels alineaments generats amb RNNs + Attention i *transformers*.

Gener:

11. Escriptura de la memòria.

1.4 Recursos

En aquest projecte seguirem com a llibre de referencia 'Deep Learning' [12] de Ian Goodfellow, Yoshua Bengio i Aaron Courville , tant per la notació i la terminologia com per la teoria relativa a les bases teòriques del deep learning i en particular de les xarxes neuronals recursives.

Per la part de la documentació referent a les eines de treball, treballarem sobre la documentació oficial de Pytorch, els entorns virtuals de Google Colab i Kaggle, i la documentació oficial de jupyter notebook.

Com a eines de suport per NLP farem servir Pandas, Sklearn, Torchtext i Spacy. Per la visualització de les dades emprarem matplotlib.

Per la part d'implementació de les xarxes recurrents amb atenció i el model *transformer* seguirem diversos articles i tutorials [27] [35] [33] que ens permetran entendre en mes profunditat el funcionament intern d'aquests models i tots els petits detalls que els conformen a l'hora d'implementar-los.

Com a guies per la correcta implementació dels mateixos seguir els models plantejats en publicacions amb un gran impacte en la comunitat de machine learning com els proposats en els papers "Sequence to sequence learning with neural networks"[34], "Jointly learning to align and translate"[3] i "Attention is all you need"[36].

Per la tasca d'alineament utilitzarem una tasca compartida proposada en el NAACL-HLT (*North American Chapter of the Association for Computational Linguistics - Human Language Tecnology Conference*) del any 2003.

Per fer-ho ens centrarem en la proposta d'alineació de paraules organitzada per Rada Mihalcea i Ted Pedersen, i el seu article posterior on s'avaluen els resultats *"An Evaluation Exercise for Word Alignment"*. [19] Les dades utilitzades son el corpus bilingües Anglès-Francès dels Hansards o transcripcions del 36é Parlament de Canada, alineats a nivell de frase i un petit subconjunt alineat a nivell de paraules proporcionades per l'investigador Germann Ulrich de la Universitat de Toronto.

En aquesta tasca compartida trobarem els datasets d'entrenament aixi com els datasets de test, eines de validació de dades i de calcul de mètriques. També tindrem un benchmark per compara el funcionament dels nostres models basats en NMT comparats amb els models basats en SMT del 2003

2 Background

2.1 Processament de Llenguatge Natural

El *Natural Language Processing* es un un camp multidisciplinar que estudia, tracta i modela el llenguatge humà mitjançant la informàtica. Sorgeix a mitjans de segle XX, amb la intenció d'aplicar les noves capacitats de càlcul al problema de la lingüística i en concret en el processament i anàlisis de grans quantitats de text. L'objectiu es aconseguir que els ordinadors entenguin els continguts del llenguatge natural i siguin capaços de treballar en diverses tasques relacionades, com reconeixement de veu, generació de llenguatge natural o comprensió del llenguatge.

Un dels principals problemes per part del NLP a l'hora de treballar amb llenguatges humans es l'ambigüitat, que es presenta de diverses maneres:

- A nivell referencial, la resolució d'anàfores o l'ambigüitat de a quina entitat anomenada prèviament es referencia mitjançant pronoms o oracions subordinades.
- A nivell estructural, quan una mateixa frase pot tenir dos arbres d'anàlisi sintàctica diferents.
- A nivell pragmàtic, ja que moltes vegades una frase depenen de la intenció del autor o del context general on es produeix. Metàfores, ironies, textos de dominis lingüístics diversos, afecten el significat i la interpretació del discurs.

Per poder treballar amb llenguatges naturals, el NLP ha de poder resoldre totes aquestes ambigüitats, normalment amb mètodes d'anàlisis estadístic i probabilístic.

2.2 Machine Translation

La traducció automàtica es una de les tasques mes complexes del NLP i te l'objectiu de produir de forma automatitzada traduccions d'alta qualitat entre diferents idiomes, traduint una seqüencia de text d'un idioma d'origen a un idioma de destí amb el resultat mes similar possible a la traducció manual professional. Aquest procés es pot simplificar en dos tasques: primer es decodifica el significat del text d'origen i posteriorment es recodifica en el idioma de destí. En el cas humà, el proces de traducció es una operació cognitiva complexa que requereix un coneixement profund dels idiomes d'entrada i sortida a molts nivells. El repte en la traducció automàtica es com aconseguir un nivell de comprensió similar al del traductor humà del text en l'idioma d'origen i com generar un nou text en l'idioma objectiu que sembli una traducció fidedigna del original.

Els sistemes de traducció automàtica es poden classificar en dos grans grups: els basats en regles lingüístiques i els que utilitzen corpus textuals.

Traducció automàtica basada en regles

Originalment els primers models funcionals de MT van sorgir a la dècada de 1970 basats en el paradigma de la traducció basada en regles. Aquesta consisteix en modi-

ficar la seqüencia de text original a traduir seguint un conjunt de regles lingüístiques i substituint mitjançant diccionaris bilingües les paraules fins arribar a l'estat final de la traducció. La traducció basada en regles inclou els mètodes de traducció per transferència, llenguatge intermedi o interlingua, i basats en diccionaris. Aquests mètodes es diferencien en l'ús de representacions intermèdies internes a l'hora de fer la traducció. Els mètodes basats en diccionaris produeixen traduccions directes paraula a paraula, els basats en transferència creen representacions internes parcials entre els dos idiomes, i els interlingua, que creen representacions internes completes independents dels idiomes d'origen i destí en el proces de traducció.

Entre els avantatges d'aquests sistemes es troba que no requereixen corpus bilingues de les dos llengües a traduir i es pot supervisar fàcilment el funcionament intern i corregir mitjançant noves regles tenint un control total sobre el procés de traducció.

Com a desavantatges tenim la mancança de diccionaris de bona qualitat, la gran complexitat en la creació dels conjunts de regles i les seves interaccions a mesura que creixen els models, i dificultats per treballar amb l'ambigüitat inherent dels idiomes.

Traducció automàtica basada en corpus

La traducció automàtica a partir d'analogies amb un corpus lingüístic es basa en l'anàlisi de mostres reals amb les seves respectives traduccions. Entre els mecanismes que utilitzen un corpus bilingüe s'inclouen la traducció basada mètodes estadístics i els basats en xarxes neuronals. Aquest tipus de models depenen en gran mesura de la mida dels corpus dels que es disposin, de la seva qualitat i correcte alineament frase a frase.

Traducció automàtica estadística

Els paradigma de SMT es basa en la generació de traduccions a partir de models estadístics basats en grans corpus de textos bilingües paral·lels alineats a nivell de paraula.

Proposats per primer cop per Warren Weaver al 1949 el concepte va ser reintroduït al 1990 a l'article 'A Statistical Approach To Machine Translation' [5] i va significar una revolució en el camp del MT, dominar fins llavors pels models basats en regles.

El paradigma de SMT parteix del concepte que totes les sentencies en un idioma son una possible traducció de qualsevol sentencia en un altre i d'una idea d'arquitectura modular composta de tres parts: el model de llenguatge d'origen, el model de traducció i el decodificador. L'objectiu es donada una frase f en l'idioma d'origen trobar una frase $\hat{e}$ en l'idioma de destí que maximitzi la probabilitat $P(e|f)$ de que f sigui una traducció a la par d'un traductor professional.

$$\hat{e} = \underset{e \in e*}{\operatorname{argmax}} P(e|f)$$

El primer element modular es el model de llenguatge de destí que s'encarrega de

calcular la probabilitat $P(e)$, es a dir la freqüència d'aparició d'una sentencia e en el llenguatge de destí. Per a seqüencies molt llargues que poden no aparèixer en el corpus, s'utilitza un enfoc probabilístic de n-grams, on la probabilitat total de la seqüencia es el producte de les probabilitats condicionals de cada una de les paraules donades les n anteriors a elles.

El model de traducció calcula donada una traducció e en l'idioma objectiu la probabilitat $P(f|e)$ de que f sigui una possible origen de la traducció. Aquesta probabilitat serà molt baixa per la gran majoria de de parelles però mes alta quan millor sigui la traducció. El model de traducció treballa sobre un corpus bilingüe alineat a nivell de paraula per tal de crear les regles per generar les probabilitats associades entre les traduccions i el les frases d'origen mitjançant diversos paràmetres per cada paraula centrats en la probabilitat de la traducció de paraula per un altre, el nombre de paraules necessàries en el llenguatge de destí, la posició de destí de la paraula dintre de la frase i la probabilitat de generar paraules noves des de zero.

Finalment el decodificador s'encarrega de generar, calcular les probabilitats i cercar la millor traducció $\hat{e}$ entre totes les possibles a partir del model de traducció.

Aplicant el teorema de Bayes, obtenim:

$$P(e|f) = \frac{P(e)P(f|e)}{P(f)}$$

i com volem maximitzar per e podem ignorar el denominador obtenint:

$$\hat{e} = \underset{e \in e*}{\operatorname{argmax}} P(e|f) = \underset{e \in e*}{\operatorname{argmax}} P(e)P(f|e)$$

El principal inconvenient de SMT es que l'espai de cerca de traduccions creix exponencialment amb el nombre de paraules. Mitjançant les regles probabilístiques del model de traducció el decodificador genera un arbre d'hipòtesis a partir de les paraules de la frase d'origen f fins trobar la frase e que maximitza el criteri de cerca. Per tal d'evitar una cerca exhaustiva i reduir l'espai de cerca s'empren heurístiques i metodologies com *greedy search* i *beam search*. Un problema afegit es la gran complexitat que presenten aquests model degut a la seva arquitectura modular. Actualment ja no es pot parlar d'estat de l'art per part dels models estadístics en el camp del MT, i els models recents mes importants son els basats en frases o *phrase based models*, que amplien el model basat en la traducció paraula a paraula a la traducció frase a frase. [**?**]

2.3 Neural Machine Translation

El paradigma de NMT proposa una solució al problema de traducció de text entre llenguatges emprant únicament xarxes neuronals artificials. Aquestes xarxes inspirades en la biologia de les neurones intenten emular el funcionament de sistemes neuronals biològics tant a nivell estructural com d'aprenentatge per tal de resoldre tasques.

Per entendre les dificultats que ha comportat l'aplicació d'aquestes xarxes al camp del MT farem una breu introducció a les bases teòriques de les xarxes neuronals profundes, la arquitectura bàsica dels models i els principals algoritmes d'entrenament.

2.3.1 Xarxes neuronals profundes

L'arquitectura general de les ANN es una xarxa o graf dirigit d'unitats interconnectades anomenades neurones. Aquestes unitats estan basades en el model original del perceptró [22] que a la vegada esta inspirat en el funcionament de les neurones, en un intent d'emular el funcionament de sistemes biològics com el cervell.

2.3.2 Neurones

L'arquitectura interna del perceptró com es pot veure a la figura 1 esta composta per un total de n entrades representades pel vector $X = x_i$,.., x_n amb els seus pesos $W = w_i, ..., w_n$ corresponents, el biaix intern b, una funció de suma ponderada $\sum_i x_i w_i$, la funció d'activació σ i una única sortida y.

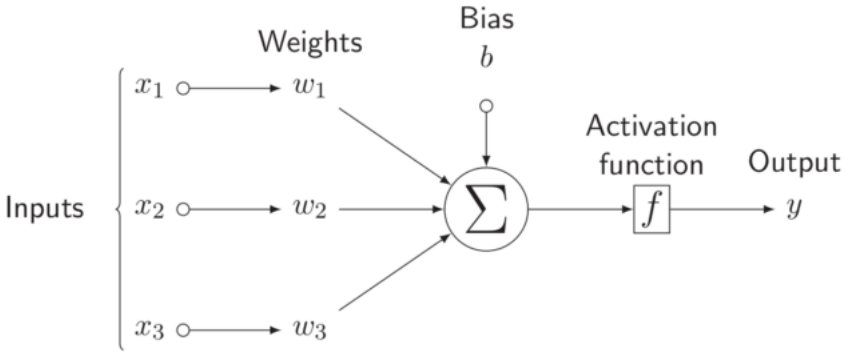

Figura 1: Estructura bàsica d'un perceptró

El funcionament intern es basa en el calcul del valor de sortida a partir de les entrades, els pesos interns i el biaix mitjançant combinació lineal. Primer es realitza la suma ponderada i li sumem b. A continuació es calcula amb la funció d'activació el valor y de sortida. El propòsit de la funció d'activació es introduir un comportament no lineal al funcionament intern del perceptró. Els pesos de les connexions d'entrada representen la importància relativa de la connexió i el biaix serveix com a variable independent de la funció d'activació permeten decidir un llindar d'activació diferent per cada neurona.

La primera funció d'activació proposada va ser la funció esglaó de Heaviside:

$$f(x) = \begin{cases} 1 \text{ si } \sum_i x_i w_i > b \\ 0 \text{ si } \sum_i x_i w_i \leq b \end{cases}$$

Aquesta funció converteix al perceptro en un classificador binari, com es pot veure a figura [**?**] on depenen del vector d'entrada X i dels valors dels paràmetres interns W i θ retornem una resposta binaria.

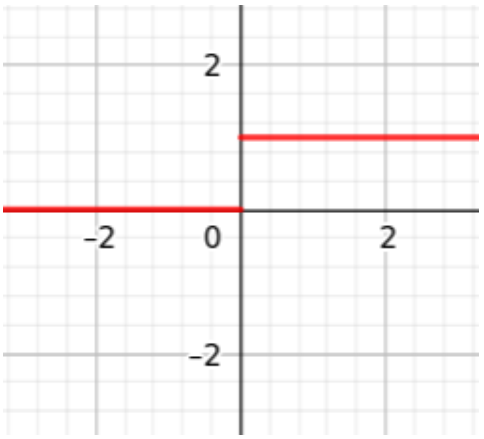

Figura 2: Funció d'activació Heaviside.

A mesura que les xarxes neuronals s'han desenvolupat han sorgit una gran diversitat de funcions d'activació. Les principals funcions d'activació emprades son:

Lineal: la funció d'activació mes simple. Figura 3. No esta acotada i tampoc introdueix cap comportament no lineal a la xarxa. S'utilitza amb la finalitat d'augmentar o disminuir el nombre de sortides.

$$f(x) = wx$$

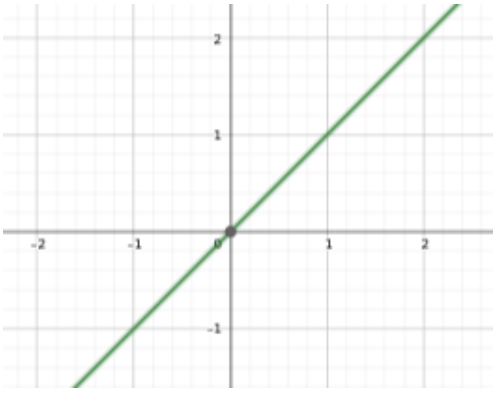

Figura 3: Funció d'activació lineal.

ReLU: Rectified Lineal Unit. Transforma el valor d'entrada a 0 si es negatiu i deixa passar els valors positius com es pot veure en la figura 4. Nomes esta acotada per la part inferior. Popular per la seva senzillesa i bon comportament en reds convolucionals i deep learning.

$$f(x) = max(0, x)$$

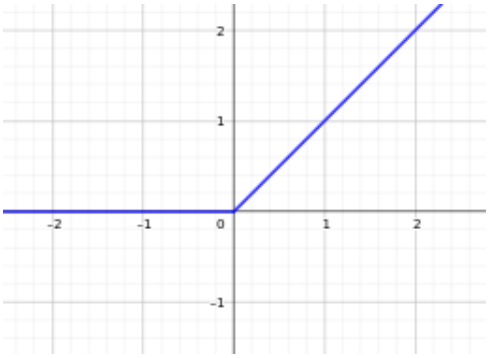

Figura 4: Funció d'activació ReLU.

Sigmoide o logística: transforma els valors al rang (0,1). En la figura 5 es pot apreciar el comportament asimptòtic de la funció.

$$f(x) = \frac{1}{1 - e^{-x}}$$

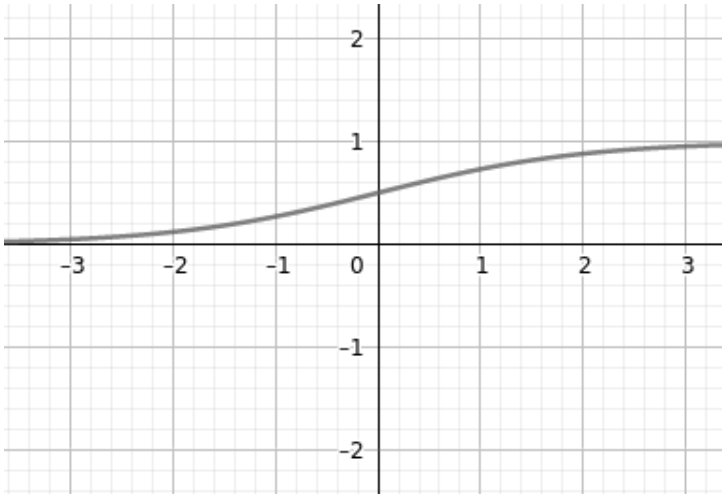

Figura 5: Funció d'activació logística.

Tanh: la funció tangent hiperbòlica (figura 6 converteix el valor d'entrada al rang (-1,1). Com la funció logística te un comportament asimptòtic en el infinit.

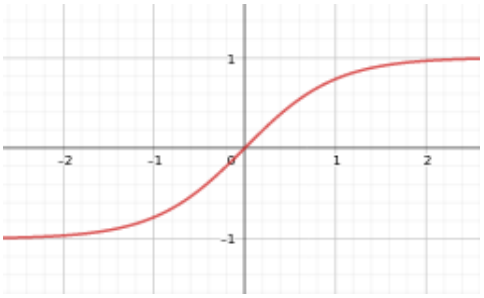

Figura 6: Funció d'activació Tanh. Imatge original

SoftMax: la funció softmax es la generalització de la funció logística a múltiples dimensions. En comptes d'acceptar un únic valor, softmax accepta un vector com entrada i retorna un vector d'igual dimensió.

$$f_i(\vec{x}) = \frac{e^{x_i}}{\sum_{j=1}^{J} e^{x_j}} \text{ per } i = 1, ...J$$

Aquesta funció aplica la funció exponencial a tots els elements del vector i retorna un vector normalitzat. Per fer-ho divideix cada exponencial entre la suma de totes les exponencials. SoftMax es pot interpretar com la distribució de probabilitats dels valors del vector d'entrada i normalment s'utilitza en la capa de sortida de les xarxes neuronals per generar una distribució de probabilitat en problemes de classificació.

La funció d'activació es l'encarregada de modificar i decidir com es propagara la informació rebuda des de la capa d'entrada per cada neurona a les neurones de les següents capes de forma seqüencial en un fenomen conegut com forward propagation fins arribar la informació a la capa de sortida. L'aprenentatge o modificació del comportament de la xarxa es realitza mitjançant la modificació amb diversos algoritmes dels paràmetres interns de les neurones: els pesos W i els biaixos b.

2.3.3 Arquitectura

La organització interna de la xarxa es produeix amb connexions neurona a neurona. Les neurones estan connectades entre elles, utilitzant la sortida d'una com l'entrada d'una altre, resultant en un graf dirigit. Cada neurona pot tenir múltiples connexions d'entrada i de sortida. Els conjunts de neurones s'organitzen normalment en capes o layers per simplificar les arquitectures.

Una capa es el conjunt de neurones operant de forma simultània a una certa profunditat dintre de la xarxa neural. En una arquitectura amb N capes les entrades de les neurones de les capa n estan connectades amb les sortides o outputs de les neurones de la capa $n-1$, i les sortides de la capa n estan connectades amb les entrades de la capa $n+1$.

Aquestes capes es poden dividir en 3 grups:

- Capa d'entrada: composta per les neurones que tenen alguna entrada connectada a l'exterior de la xarxa. Aquestes neurones son les encarregades de processar la informació rebuda per la ANN.
- Capa oculta: son les capes formades per neurones sense cap connexió directe amb l'entrada o la sortida. Aquestes capes son l'origen de les arquitectures d'aprenentatge profund.
- Capa de sortida: formada les neurones connectades mitjançant la seva sortida amb el exterior de la xarxa. Aquestes neurones retornen els resultats de la ANN.

Quan la connexió entre les capes es produeix de forma que totes les entrades de la capa n reben inputs de totes les sortides de cada una de les neurones de la capa $n-1$ diem que n es una capa densa i que aquestes dos capes estan totalment connectades o fully connected. Podem veure un exemple en la figura 7. Quan no es compleix aquest requisit parlem de capes escassament connectades o sparsely connected.

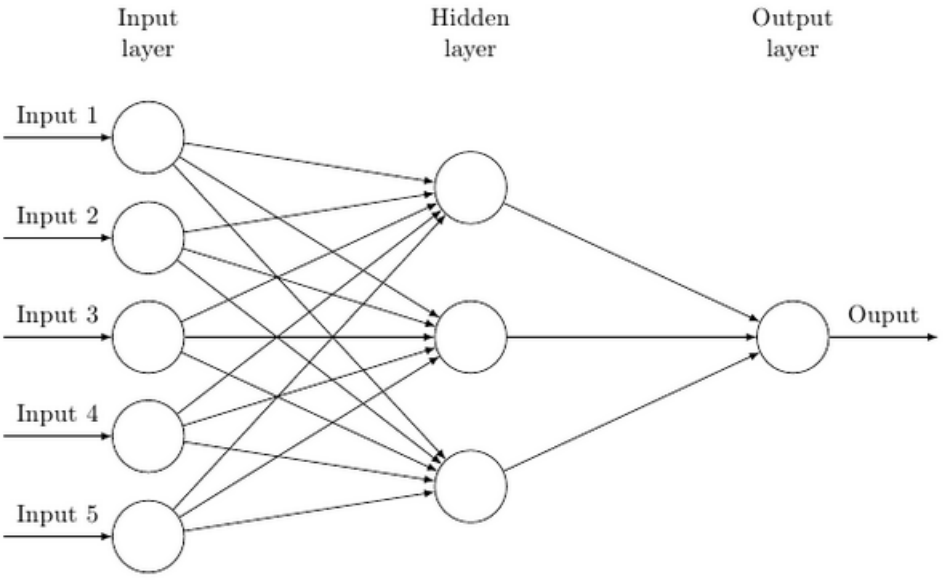

Figura 7: Estructura bàsica d'una ANN.

2.3.4 Xarxes neuronals directes

Quan totes les neurones estan connectades formant un graf dirigit acíclic parlem de xarxes neuronals directes o feedforward networks. Reben aquest nom per que la informació flueix des de la capa d'entrada fins la capa de sortida a través de les neurones [12].

El model mes comú per aquest tipus de xarxa es pot expressar com una composició de funcions. Una xarxa amb tres capes es pot expressar com: $f(x) = f^3(f^2(f^1(x)))$ on les funcions representen les diferents capes de la xarxa, sent f^1 la capa d'entrada, f^2 la segona capa, en aquest cas oculta i f^3 la capa de sortida. Aquesta arquitectura

de composició de funcions en profunditat es el que dona el nom al aprenentatge profund.

El propòsit d'aquestes xarxes es aproximar-se a una funció objectiu $\hat{f}$. En el model d'aprenentatge supervisat les dades de les que disposem son parelles de les entrades i sortides de la xarxa.

La xarxa neuronal defineix una funció $y = f(x, \theta)$ i aprèn els valors dels paràmetres interns de les neurones θ compost pels pesos i els biaixos que millor aproximen la funció f a $\hat{f}$.

2.4 Aprenentatge supervisat

L'aprenentatge de les xarxes neuronals es basa en l'algoritme de gradient descendent [28]. Aquest es un mètode iteratiu d'optimització de primer ordre per a trobar el mínim d'una funció derivable. la idea es partint d'un punt donat, buscar el mínim de la funció prenent passos definits per una valor d'aprenentatge γ en la direcció oposada al gradient o pendent de la funció i re-calcular aquest gradient abans de prosseguir. Per tant per poder aplicar aquest algoritme es necessari definir una funció derivable anomenada funció de cost que aproximi el error de la sortida inferida del model envers la sortida real.

2.4.1 Inicialització del model

Per poder dur a terme l'aprenentatge, es a dir, aproximar de forma incremental la sortida de la xarxa a la funció objectiu el primer pas es inicialitzar els paràmetres interns del model. En aquest pas es solen emprar estratègies basades en diverses distribucions aleatòries o estadístiques, per exemple una distribució normal o uniforme.

2.4.2 Processament de dades

A continuació cal preparar les dades per que puguin ser processades pel model. A part del processament necessari per assegurar la qualitat, uniformitat i validesa de les dades, les ANN només poden processar valors numèrics com a dades d'entrada. Com en el cas del NMT on les dades tant d'entrada com de sortida tenen un format de cadenes de caràcters, el primer pas es transformar-les en un format que la xarxa neuronal accepti com a entrada. Aquest procés sol dur-se a terme en dos passos diferenciats: la tokenització i el embedding.

Tokenització

La tokenització es el procés de transformar una cadena de llenguatge natural, per exemple una frase, en una llista de subcadenes. Cada una d'aquestes subcadenes s'anomena token i són la unitat mínima de text amb la que treballarem. Aquest procés normalment es pot dur a terme a nivell de paraules, subparaules o símbols. Depenen del llenguatge existeixen estratègies mes optimes, ja que podem trobar grans diferencies morfològiques entre diferents idiomes.

Embedding

El embedding es el proces de transformar els tokens en valors numèrics. En aquesta tasca tenim dos estratègies principals: one hot encoding vector i word embedding.

El primer pas es crear dos vocabularis, un pel corpus d'entrada i un altre pel corpus de sortida. Un vocabulari esta format per tots els tokens únics que formen part del corpus acompanyats per un index. Com el nombre de tokens pot variar enormement des de pocs milers fins a milions depenen de les dades i els idiomes amb els que treballem, cal fer un cribratge dels tokens mitjançant diferents criteris com per exemple de freqüència mínima o mida màxima de vocabulari.

Com a tokens especials que s'afegeixen al vocabulari tenim 4: <SOS>, <EOS>, <UNK> i <PAD>.

- <SOS>: Start of sequence. Aquest token es utilitzat per indicar al model el inici d'una seqüencia d'entrada.
- <EOS>: End of sequence. Indica el fi d'una seqüencia d'entrada o de sortida.
- <UNK>: Unknown. Serveix per substituir tots aquells tokens que no es trobin al vocabulari i que per tant siguin desconeguts pel model.
- <PAD>: Padding. S'utilitza per reomplir la capa d'entrada a partir del token de final de seqüencia.

El motiu de no poder utilitzar directament els index del vocabulari com entrades per la xarxa es deu a que interpretara aquests valors com informació rellevant dels tokens. Per exemple, a l'hora de calcular la distancia o error entre dos paraules, aquesta dependre dels valors dels seus index. Amb valors molt diferents el model interpretara l'error com mes gran que si tenen valors contigus en el vocabulari.

Un cop tenim els vocabularis, procedim a crear el one hot encoding. Aquest es un vector per cada token de la mida del vocabulari, on tots els valors son 0 excepte per la posició corresponent al index del token que es 1. Així tenim un vector únic per cada token, on les distancies geomètriques entre tots son iguals, permeten al model utilitzar aquesta informació pels seu funcionament intern.

Una estratègia més elaborada es el word embedding on l'objectiu es reduir la dimensionalitat dels vectors de one hot encoding transformant-los en vectors de nombres reals de menor longitud que a la vegada codifica informació semàntica rellevant. Per reduir la dimensionalitat i a la vegada introduir en la codificació del embedding s'utilitzen tècniques estadístiques com Anàlisis de Components Principals (Principal Component Analisis, PCA), models probabilístics i xarxes neuronals. [21]. Dintre de l'enfoc neuronal al word embedding sobresurt la tècnica word2vec que empra grans corpus de text per entrenar xarxes neuronals per codificar les representacions de cada token. Els vectors es trien de manera que la operació de similitud de cosinus entre dos vectors A i B retorni la similitud semàntica entre les paraules representades pels dos vectors.[20]

$$\text{Similitud de cosinus} = \cos\alpha = \frac{AB}{|A||B|}$$

Com a resultat, vectors que representen paraules properes semànticament o sinònims es troben propers en el espai vectorial i emprant la magnitud de similitud de cosinus podem trobar com de relacionades estan les dos paraules, indicat amb els valors de l'interval $[1, -1]$.

Validació creuada

La validació creuada o cross validation en permet entrenar i validar el model mitjançant un únic conjunt de dades. Un cop tenim les dades tokenitzades les dividim en un subconjunt d'entrenament i un subconjunt de validació. El conjunt d'entrenament servira per alimentar la xarxa neuronal amb parelles de dades d'entrada i sortida que pertanyen a la funció objectiu $\hat{f}$ que volem replicar i el conjunt de validació servira per calcular com s'aproxima realment la funció entrenada a un conjunt de dades no vist. Aquesta divisió serveix per evitar el fenomen conegut com overfitting, on els paràmetres de la xarxa minimitzen el error calculat amb la funció de cost pel conjunt de dades d'entrenament a costa d'augmentar l'error en el conjunt de dades de validació, sent necessari aturar l'entrenament abans d'acabar el nombre d'entrenaments previst realitzant un early stopping quan la funció de cost del model deixa de convergir.

Finalment aquest mètode requereix un tercer conjunt de dades anomenat de test que mai s'ha de fer servir per l'entrenament o la validació ja que serveix com a referencia objectiva per avaluar i comparar el rendiment de la xarxa neuronal, sent el mes similar possible al funcionament real del model fora del conjunt de dades d'entrenament.

2.4.3 Funció de cost

La funció de cost o loss function $C(y, \hat{y})$ calcula a partir de la diferencia entre la sortida esperada y i la obtinguda pel model $\hat{y}$ l'error produït en la predicció mitjançant un nombre real que ens indica la magnitud. Un cop calculada la funció de cost l'objectiu es minimitzar el seu valor.

Funció de cost quadràtic

La funció de cost quadràtic mig s'utilitza per problemes de regressió i permet obtenir bons resultats donats dos vector de longitud n on y es el total de sortides esperades i $\hat{y}$ conte les sortides inferides pel model. Aquest normalitza l'error tant si es positiu com negatiu i calcula la mitja de tots els exemples. Aixi doncs per millor el model només cal minimitzar la funció de cost.

$$C(y, \hat{y}) = \frac{1}{n}\sum_{i=0}^{i=n}(y_i - \hat{y}_i)^2$$

Funció de cost d'entropia creuada

La funció de cross entropy loss mesura l'error entre dos conjunts de dades a través del concepte d'entropia de la teoria de la informació.

La funció mesura el nombre de bits que necessitem de mitjana per identificar un esdeveniment d'un conjunt mitjançant una distribució de probabilitats estimada, es a dir $\hat{y}$ en comptes de la distribució verdadera y. Com el nombre de bits necessaris sempre serà major per qualsevol estimació excepte en el cas on $y = \hat{y}$ on la entropia creuada i l'entropia son iguals aquesta funció ens es útil per calcular l'error. Per tant obtenim una funció de cost que en la que caldrà buscar un mínim global.

$$C(y, \hat{y}) = \sum_{i=1}^{n} \ln \frac{1}{\hat{y}_i} = -\sum_{i=1}^{n} y_i \ln \hat{y}_i$$

Pel cas particular de NMT es pot generalitzar la funció de cross entropy loss per tal de mesurar la entropia creuada total entre les probabilitats inferides per cada token del vocabulari de destí respecte de cada token de la seqüencia objectiu. Es calcula el sumatori per cada token de la seqüencia objectiu del logaritme negatiu de la probabilitat calculada pel token correcte.

$$C(y, \hat{y}) = -\sum_{s=1}^{S} \sum_{v=1}^{V} y_{s,v} \log \hat{y}_{s,v}$$

On S la longitud de la seqüencia de sortida,V es la del vocabulari de sortida, $\hat{y}_{s,v}$ la predicció de probabilitat de cada token del vocabulari respecte el token de la seqüencia de sortida correcte i $y_{s,v} = 1$ quan el token predit es correcte i $y_{s,v} = 0$ quan es incorrecte.

Per tant la funció de cost augmenta exponencialment com mes baixa es la probabilitat inferida per la xarxa pel token correcte i disminueix a 0 quan la probabilitat calculada es 1, permeten aplicar el algoritmes de gradient descendent per l'aprenentatge de la tasca de traducció automàtica.

2.4.4 Descens de gradient

Un cop tenim la funció de cost l'objectiu ja podem minimitzar el seu valor modificant els valors dels paràmetres interns de la xarxa θ i b i idealment aconseguir $C(y, \hat{y}) = 0$. Com resoldre analíticament el mínim de la funció es difícil ja que parlem d'espais d'alta dimensionalitat amb un elevat nombre de paràmetres la solució es el algorisme del gradient descendent de cerca de mínims en funcions derivables. Aquest algoritme ens permet minimitzar la funció de cost de forma incremental aproximant en petits passos el mínim de la funció de cost. Per fer-ho el algoritme aproxima una seqüencia de punts $y_1, y_2, ..., y_k$ tals que $C(\hat{y}) - C(y_k) = 0$ quan $k \to \infty$ com es pot veure en la figura 8.

El primer pas es calcular el gradient de la funció de cost $\bigtriangledown C(y_n, \hat{y})$ i mitjançant passos iteratius calcular les solucions $y_{n+1} = y_n - \gamma \bigtriangledown C(y_n, \hat{y})$ aproximant-nos al mínim global de la funció de forma proporcional al ritme d'aprenentatge γ i al valor del gradient en el punt y_n de per minimitzar l'error. Les principals funcions de cost utilitzades son la de cost quadràtic i la cross entropy loss.

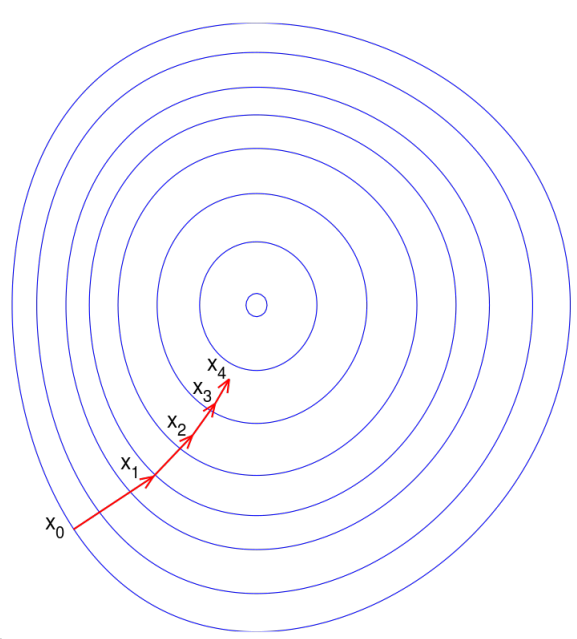

Figura 8: Exemple de descens de gradient

Normalment degut al cost computacional i a que obtindríem un ritme d'aprenentatge molt lent no es calcula el verdader gradient per tot el conjunt de dades d'entrenament, i en comptes s'aplica la variant de descens de gradient aleatori o Stochastic gradient descent (SGD). En aquest cas s'utilitza una mostra d'entre 1 i k elements, anomenada batch, per calcular un gradient pel subconjunt i aplicar l'algoritme descrit per calcular el gradient parcial.

Aquest proces es repeteix per tot el conjunt de dades d'entrenament i per cada conjunt total de dades d'entrenament que propaguem per la xarxa diem que ha passat una època. Normalment, a mesura que passant les èpoques els models convergeixen envers un valor mínim de la funció de cost. Un cop obtenim una època amb els millors valors de la funció de cost pel conjunt de validació, ja tenim entrenada la nostra xarxa neuronal i nomes cal testejar i calcular la funció de cost en el conjunt de dades de test i comprovar els valors definitius. Ara bé, per tal d'aplicar el descens de gradient a una xarxa neuronal ens cal saber com aplicar aquest descens de gradient per cada pas a tots els pesos i biaixos de la xarxa neuronal. Per fer-ho s'empra l'algoritme de backpropagation o retropropagació.

2.4.5 Retropropagació

L'algoritme de backpropagation [29] es un algoritme per modificar els pesos i biaixos interns de les neurones d'una xarxa neuronal per mitjançant múltiples iteracions basades en el algoritme del gradient descendent minimitzar l'error de la funció de cost fins a trobar un mínim.

Aquest algoritme s'inicia alimentant el model amb les entrades d'entrenament o d'un batch en el cas de SGD fins la capa de sortida. Per cada forward propagation calculem un error total E_T obtingut per la funció de cost. Aquest error es la suma dels errors de tots els outputs de les neurones de la capa de sortida $E_T = \sum_{i=1}^{n} E_i$. Un cop tenim aquest l'error per cada element del batch la idea es propagar-lo cap enrere en el camí invers, calculant la contribució parcial de cada paràmetre de la xarxa i modificant en conseqüència el seus valors.

Per poder aplicar el gradient descendent cal calcular l'error parcial contribuït per cada pes i biaix de la capa de sortida mitjançant la derivada parcial del error respecte cada pes:

$$\frac{\partial E_T}{\partial w_i}$$

i cada biaix:

$$\frac{\partial E_T}{\partial b_i}$$

Un cop obtenim la derivada parcial en el pes o biaix els procedim a actualitzar una funció d'optimització amb el valor de la derivada parcial, el valor previ del pes i del ritme d'aprenentatge:

$$w_i = w_i - \gamma \frac{\partial E_T}{\partial w_i}$$

$$b_i = b_i - \gamma \frac{\partial E_T}{\partial b_i}$$

Un cop tenim els nous valors per la capa de sortida procedim a repetir l'algoritme per les capes anterior fins arribar a la capa d'entrada, tenint en compte les contribucions relatives dels pesos i biaixos cada neurona a l'error total. Mitjançant la regla de la cadena i les derivades parcials en cada pes, simplifiquem el calcul de les derivades parcials de les funcions d'activació de les neurones i es simplifica l'algoritme de backpropagation. Un dels factors a tenir mes en compte en aquest algoritme es el ritme d'aprenentatge aixi com la funció d'optimització.

Explosió de gradient

Un dels problemes del backpropagation es l'explosió o desaparició de gradient que es dona en el calcul dels gradients parcials dels pesos i biaixos de les neurones. El calcul del gradient en el backpropagation es realitza mitjançant multiplicacions de matrius i donat la profunditat de les xarxes, els nombres poden créixer cap a infinit o disminuir molt ràpidament a 0, retornant errors NaN per part de la xarxa.

La principal estratègia per evitar el exploding gradient es el cliping o retallar els gradients que superen cert llindars numèrics per evitar el seu creixement excessiu o desaparició, aixi com una tria correcte de les funcions d'activació de les neurones ja que el gradient dependrà en gran mesura de les seves derivades.

Regularització

Un bon model no només ha de minimitzar les funcions de cost, si no que ha de ser capaç de generalitzar. Quan un model generalitza malament, es a dir, que obté pitjors resultats pels conjunts de dades de validació que pel conjunt de dades d'entrenament diem que s'ha produït overfitting. Per evitar que els models es centrin únicament a minimitzar en les dades d'entrenament i que aprenguin també a generalitzar la funció per minimitzar aixi l'error en les dades de validació tenim dos estratègies principals anomenades de regularització.

En el primer cas es tracta d'evitar que aquest fenomen es pugui donar. Mitjançant conjunts de dades d'entrenament mes grans que el nombre de paràmetres interns de les xarxes evitem que aquesta pugui inferir exactament la distribució de punts i l'obliguem a extrapolar. Això però no sempre es possible tant per limitacions en les dades com de memòria disponible per guardar i entrenar el model.

Per altre banda tenim la tècnica de dropout. [15] Aquesta tècnica de regularització consisteix a eliminar de forma aleatòria un percentatge del total de connexions entre neurones de les diferents capes on s'apliqui per cada proces de feedforward/backpropagation. Aquesta tècnica es molt eficaç a l'hora d'evitar overfitting en tot tipus de xarxes neuronals. [32] L'ús del dropout s'ha convertit en l'estàndard mes emprat gracies a la seva senzillesa i eficàcia.

2.4.6 Feed Forward Networks i Neural Machine Translation

Ara bé, sabent que les xarxes neuronals directes son perfectes per l'entrenament supervisat, perquè no s'utilitzen per les tasques de NMT donada la disponibilitat de corpus bilingües alineats? Encara que teòricament les xarxes neuronals directes son aplicables al problema de la traducció automàtica, no son populars degut a tres problemes propis de la seva arquitectura.

El principal es que les DNN tenen una capa d'entrada de longitud fixa. Davant la necessitat de processar seqüencies de dades de longitud variable, la única solució passa per o be limitar l'operabilitat de la xarxa a entrades de fins a certa longitud màxima o bé dividir les seqüencies en subseqüencies menors, fragmentant la traducció .

Per altre banda el ordre de les seqüencies de paraules en les frases d'entrada i les relacions entre elles tenen una gran importància per la correcta traducció però les DNN tenen problemes a l'hora de codificar l'ordre intern de les entrades d'una forma significativa i també d'establir relacions diferenciades entre elles ja que estan enfocades al processament de totes les entrades de forma paral·lela.

I finalment, tal i com hem vist amb els models de SMT, la xarxa neuronal per tal de realitzar la traducció ha d'aprendre ha realitzar 2 tasques: codificar el llenguatge

d'entrada a una representació interna i posteriorment decodificar aquesta representació interna en el llenguatge de sortida.

Això ha portat a l'utilització de xarxes neuronals alternatives al model DNN enfocades al processament de dades de forma seqüencials: les xarxes neuronals recurrents. L'ús d'una arquitectura amb 2 parts diferenciades connectades en serie on la primera realitza la funció de codificar l'entrada a un estat intern i la segona decodifica aquest estat al llenguatge de sortida, anomenada encoder - decoder. I finalment el desenvolupament de capes de neurones amb la capacitat d'extreure les relacions internes entre els diferents elements de les seqüencies: els mecanismes d'atenció.

2.5 Xarxes neuronals recurrents

Una xarxa neuronal recurrent és una classe de ANN on les connexions entre les diferents neurones formen un graf dirigit amb cicles. Les connexions de realimentació que defineixen aquestes xarxes es poden classificar en 3 classes com es veu en la figura 9):

- Realimentació directa : la sortida de la neurona s'aplica com a entrada en la mateixa neurona.
- Realimentació indirecta : la sortida de la neurona s'aplica com a entrada en la neurona d'una capa anterior.
- Realimentació lateral : la sortida de la neurona s'aplica com a entrada en una altra neurona de la mateixa capa.

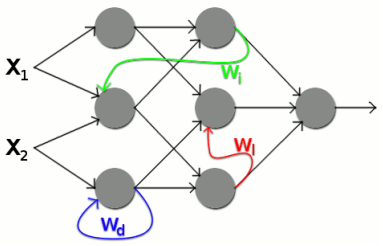

Figura 9: Tipus de RNN segons connexions : Directa (línia blava), Indirecta (línia verda) i Lateral (línia vermella).

La tipologia de connexió mes usada ja que simplifica tant el disseny com l'operació de els RNN es la realimentació directe. En el disseny mes bàsic com en la figura 10, es creen capes de neurones recursives on la neurones passen la seva sortida no només a la següent capa del model si no també a si mateixes.

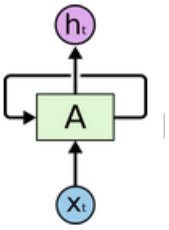

Figura 10: Esquema de xarxa recurrent bàsica. Imatge extreta de [24]

2.5.1 Arquitectura

Per poder entendre com es mou la informació dins d'una RNN és útil pensar en elles com múltiples copies de la mateixa xarxa en diferents estats de temps o passos, cada una rebent o passant informació a la següent en diferents moments temporals.

Al desenvolupar aquesta relació en diferents passos $t, t+1, t+2, ..., \tau$ podem veure com es desenvolupa el graf seqüencial de la xarxa en la figura 11 .

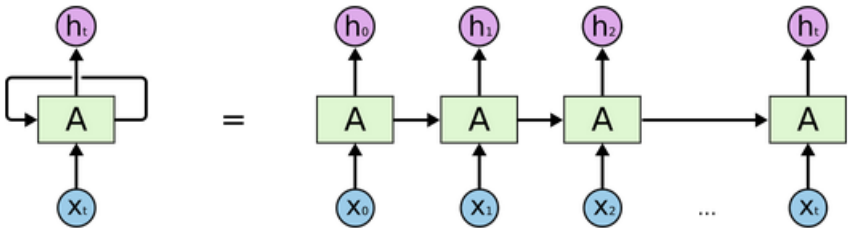

Figura 11: Desplegament d'una xarxa neuronal recurrent en múltiples estats temporals. Imatge extreta de capítol 10.1 de [12].

Per simplificar a nivell matemàtic, ens podem referirem a les RNN com funcions que operen sobre una seqüencia de vectors d'entrada $x(t)$ amb la variable t en el rang temporal de 1 a τ.

Mitjançant aquesta organització a nivell de connexions entre les pròpies neurones, les xarxes neuronals passen a tenir memòria interna al permetre que la informació generada en una entrada prèvia en una neurona persisteixi en el temps i sigui utilitzada en múltiples forward propagations. Aquesta informació es pot entendre com paràmetres interns compartits a través dels temps per les connexions recurrents, amb els seus propis pesos i propagats a través de funcions d'activació com es pot veure en l'exemple de la figura 12.

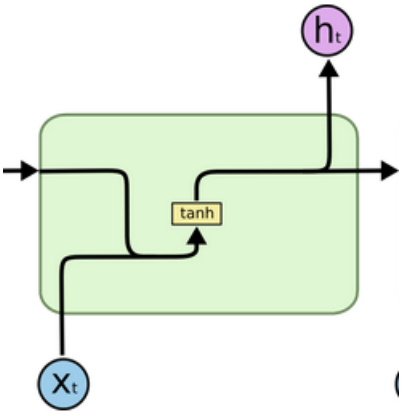

Figura 12: Exemple de connexió recurrent bàsica amb funció d'activació tanh. Imatge extreta de [24]

2.5.2 Aprenentatge supervisat aplicat a RNN

A l'hora d'aplicar els algoritmes d'aprenentatge automàtic de xarxes neuronals profundes a les xarxes recurrents cal tenir en compte aquesta nova dimensió temporal.

En la propagació endavant cal tenir en compte a més dels estats transmesos a la xarxa en l'estat de temps t per calcular les funcions d'activació, els estats transmesos per la pròpia xarxa en moments anteriors a t. Aquests estats s'anomenen estats ocults

o hidden states i els representem com $s(t)$. Son els paràmetres compartits a través del temps en els que es basa la memòria de la xarxa.

Per part de la funció de cost, cal tenir en compte la mitjana entre les sortides de tots els intervals de temps anteriors en el desplegament de la xarxa recurrent, ja que volem calcular la funció de cost de tots els estats des de 0 fins a τ de la RNN.

$$C(y, \hat{y}, \tau) = \frac{1}{\tau} \sum_{t=1}^{\tau} C(y_t, \hat{y}_t)$$

per calcular funcions de cost parcials podem aplicar la mateixa funció però reduint el desplegament de la xarxa fins el instant de temps t que volem analitzar.

A l'hora d'entrenar les RNN cal tenir en els estats ocults compartits i per tant el gradient de la funció de cost depèn dels paràmetres i funcions d'activació en el instant de temps t i de tot els anteriors. per tant l'algorisme de retropropagació s'ha de generalitzar a través del temps.

Conegut com Backpropagation through time [37] al igual que Backpropagation utilitza el calcul de gradient a partir de la funció de cost. Un cop la xarxa a processat un batch o una època de dades, es realitza un desplegament o unfolding de la RNN en els seus diferents estats temporals des de $t = 1$ fins a τ. Amb la xarxa desplegada apliquem backpropagation des de la capa de sortida propagant la funció de cost parcial per cada pes i biaix fins arribar a la capa d'entrada. Un cop hem propagat l'error repleguem la xarxa i actualitzem els paràmetres amb els gradients d'error parcials calculats en tots els moments temporals.

Aprenentatge forçat

Una tècnica per accelerar l'aprenentatge de les DNN i molt útil per les RNN es el aprenentatge forçat o *forced teaching*. Aquesta tècnica d'aprenentatge supervisat es basa en ignorar en funció d'un paràmetre probabilístic entre 0 i 1 les sortides produïdes pel decoder de la xarxa a l'hora de reutilitzar-les com el següent input. En comptes de tenir en compte la sortida erroni per la següent iteració, propagant el possible error comes, es força l'ús del token correcte en la següent iteració. Això permet per seqüencies llargues que la xarxa entreni sobre seqüencies mes similars a les objectiu i que l'aprenentatge s'acceleri.

Dependències a llarg termini

Una de les grans avantatges de les RNN es la introducció del concepte de memòria. Aquestes xarxes son capaces d'utilitzar informació prèvia per tal de millorar el seu procés d'inferència. Tal i com succeeix en la comprensió del llenguatge, les paraules anteriors en una frase tenen un gran impacte en el sentit o el significat de les paraules posteriors. Tot això porta a pensar que les RNN en principi solucionen definitivament el problema del processament d'informació seqüencial interrelacionada.

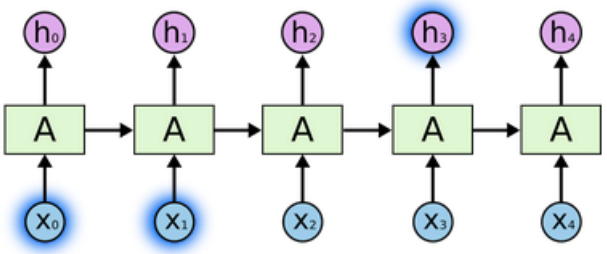

Figura 13: Exemple de dependència a curt termini entre les entrades x_0 i x_1 amb la sortida h_3. Imatge extreta de [24]

Encara que les RNN mes bàsiques com la vista en la figura 12 son capaces de guardar informació considerada important per la xarxa d'un estat fins el següent com en el exemple de la figura 13 i teòricament podria aprendre a crear relacions a llarg termini, aquestes unitats de memòria o loops interns es degraden ràpidament.

Aquest fenomen es causat per la funció d'activació interna, els pesos específics de la connexió i l'aprenentatge de la pròpia xarxa, que estan subjectes als fenòmens d'explosió o desaparició de gradient, tornant molt volàtil la informació emmagatzemada i portant a avaluar de forma mes positiva les relacions a curt termini que les de llarg termini independentment de la seva rellevància relativa. A mesura que augmenten les distancies entre les entrades i les sortides rellevants (Figura 14) les RNN es tornen cada cop mes incapaces de relacionar la informació de forma significativa.

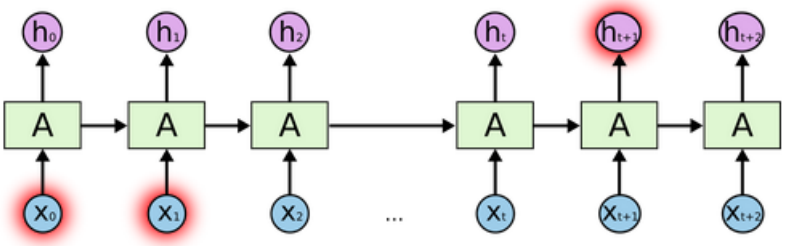

Figura 14: Exemple de dependència a llarg termini entre les entrades x_0 i x_1 amb la sortida h_{t+1}. Imatge extreta de [24]

La idea de incloure informació mes persistent en el temps en els estats interns de les neurones va portar al desenvolupament de dos neurones basades en la realimentació directe: Long Short Term Memory o LSTM i les Gated Recurrent Units o GRU. Aquestes neurones han portat a la creació de les Gated Recurrent Neural Networks.

Actualment aquestes dos son els dissenys de neurones recurrents mes utilitzats per implementar recursivitat i el que va permetre generalitzar l'ús de les xarxes neuronal en el problema del NMT ja que solucionen fins a cert punt el problema de la dependència a llarg termini.

Long Short Term Memory

Les LSTM van ser proposades per primer cop al 1997. [16] Aquestes neurones presenten una gran complexitat en el seu disseny.

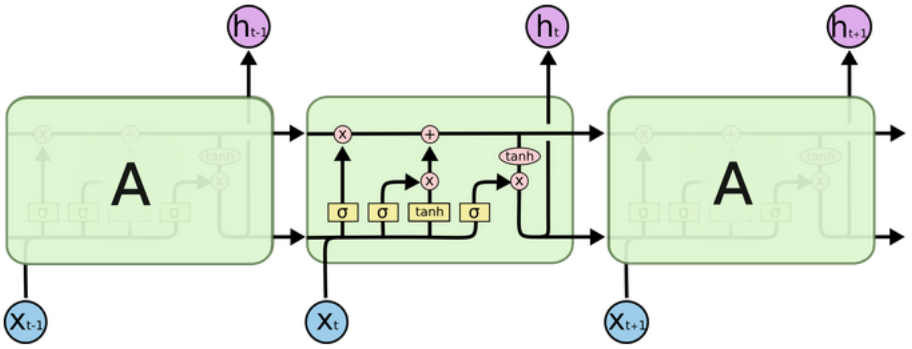

Figura 15: Esquema intern d'una LSTM on es pot apreciar l'alta complexitat que presenta. x_t es el vector d'entrada, h_t el vector de sortida o hidden state, els rectangles representen la capa neuronal interna i els cercles operacions element a element com el producte de Hadamard. Imatge extreta de [24]

La idea central del LSTM es la inclusió d'un estat intern anomenat cell state c_{t-1} i una capa neuronal interna amb tres portes lògiques o gates que realitzen diferents funcions. Com es pot veure en la figura 15 el cell state es representat amb la línia horitzontal superior i que es propaga des de l'anterior iteració de la RNN, es modificat i segueix fins la següent iteració com c_t, i una part inferior on tenim la sortida anterior h_{t-1} i l'entrada x_t. Cada porta o gate te els seus propis pesos w i biaixos interns b.

L'estat intern c_{t-1} només rep dos interaccions lineals, una operació de producte element a element o de Hadamard i una operació d'addició. Mitjançant el producte la LSTM es capaç de modificar el valor del cell state oblidant valors i amb l'operació d'addició pot afegir informació, tot en funció de la sortida anterior h_{t-1} i l'entrada x_t.

La primera operació que serveix per oblidar el estat anterior es du a terme amb la forget gate. Aquesta porta empra una funció d'activació sigmoide amb les entrades h_{t-1} i x_t i amb la sortida realitza un producte de Hadamard amb c_{t-1} per decidir quant oblidar conservar o oblidar del cell state anterior.

L'operació de la input gate consisteix en decidir quanta informació nova s'afegira al c_{t-1}. En aquesta porta, s'utilitzen dos funcions d'activació. La primera, es una sigmoide de forma anàloga a la forget gate decideix quins valors afegir a c_{t-1} per recordar. La segona es una funció d'activació tanh que crea un vector de valors candidats. Mitjançant un producte element a element obtenim els candidats escollits i amb una operació d'addició els afegim a c_{t-1}.

Un cop hem oblidat i afegit valors a c_{t-1} ja tenim el nou cell state c_t.

Finalment la tercera porta es la output gate que decideix la sortida h_t de la LSTM i implementa la funció de recordar. En aquesta porta s'utilitzen els valors del cell state c_t per modificar la sortida. Primer utilitzem la funció d'activació sigmoide sobre h_{t-1} i x_t per triar quines parts utilitzarem en la sortida. A continuació amb una operació tanh sobre c_t deixem els valors en l'interval $(-1, 1)$ i procedim a realitzar un producte element a element amb la sortida de la porta sigmoide per obtenir h_t.

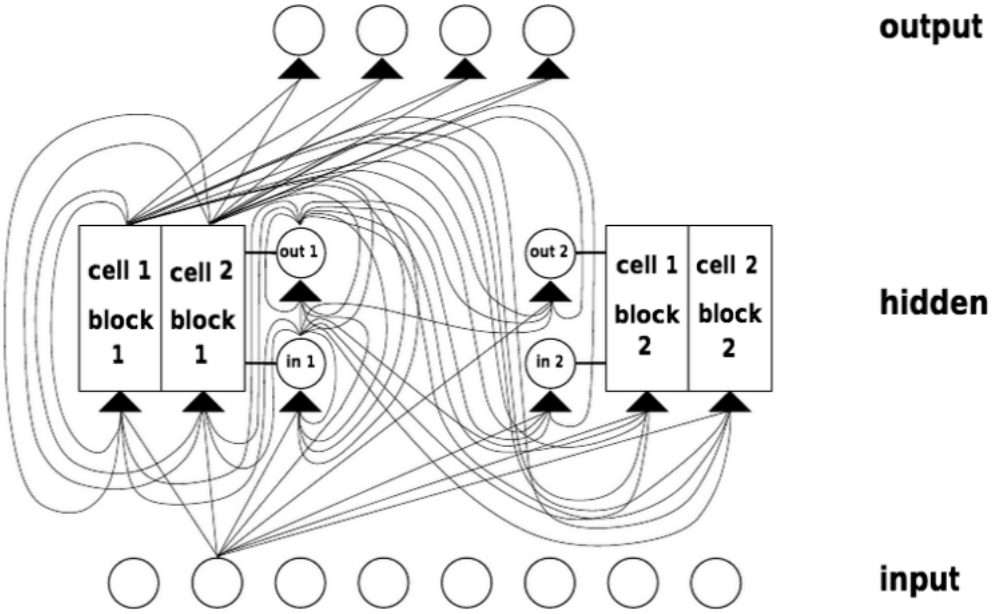

Figura 16: Primera proposta de xarxa neuronal directe amb LSTM. Imatge extreta de [16]

El principal problema de el disseny LSTM es la seva complexitat interna. Com es pot veure en la figura 16 el primer disseny d'una xarxa neuronal amb aquests components no deixa de ser una obra d'art d'alta complexitat.

L'arquitectura de RNN basada en LSTM es va proposar per primer cop al 2014 [13] per processar i generar posteriorment seqüencies complexes inferint les seqüencies element a element, passant la sortida generada com entrada del següent pas de forward propagation i utilitzant capes ocultes de LSTM per emmagatzemar i transmetre la informació rellevant entre passos a llarg termini. Aquesta primera implementació va estar dirigida al processament de text i la generació de seqüencies de text amb escriptura cursiva.

Gated Recurrent Units

La Gated Recurrent Unit o GRU va ser proposada per primer cop al 2014. [7] Aquesta es una versió simplificada de la LSTM on es combinen la forget i input gate en una única update gate, i es simplifica el cell state fusionant-lo amb el hidden state o sortida.

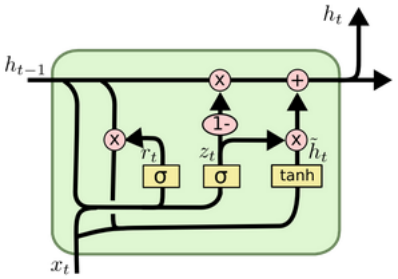

Figura 17: Esquema intern d'una GRU. x_t es el vector d'entrada, h_{t-1} el vector de sortida o hidden state, els rectangles representen la capa neuronal interna i els cercles operacions element a element com el producte de Hadamard. Imatge extreta de [24]

En el model GRU, la update gate esta regida per la funció d'activació sigmoide z_t, encarregada d'oblidar i afegir informació al hidden state. La funció d'activació encarregada de recuperar informació es la sigmoide r_t i finalment la output gate esta controlada per la funció d'activació tanh h_t que rep inputs de la funció d'activació r_t, del hidden state h_{t-1} i actualitza el hidden state h_t en funció de la update gate z_t.

Les GRU han demostrat un rendiment superior en quant a complexitat computacional respecte les LSTM mantenint uns nivells comparables de rendiment en molts problemes de ML enfocats al processament de seqüencies utilitzant RNN.[8] Gracies a aixo les Gated Recurrent Units s'han convertit en una alternativa popular a els LSTM.

2.5.3 Encoder decoder

En el mateix article on es va introduir la GRU es va fer la primera implementació d'una RNN amb capes GRU amb una arquitectura novel anomenada encoder - decoder enfocada l'any 2014. [7] Aquest nou model aplica el model modular de SMT al NMT dividint l'arquitectura en dos xarxes neuronals recurrents diferenciades com es pot veure en la figura 18. Per una banda tenim el encoder, que s'encarrega de processar les dades d'entrada i donar com a sortida un vector anomenat de context. Per altre banda tenim el decoder, que s'encarrega de partint del vector de context com a entrada generar un a un els elements de la sortida.

En la implementació del mateix any en l'article 'Sequence to sequence learning with Neural Networks' [34] un model basat en encoder-decoder i LSTM aconsegueix superar per primer cop els models de SMT en la tasca de la traducció automàtica.

En aquest article es va popularitzar l'ús de l'estructura encoder i decoder per la gran majoria de models enfocats a processar i retornar seqüencies, coneguts com sequence to sequence, seq2seq. La funció d'aquesta estructura es dividir la tasca seq2seq en dos parts mitjançant dos xarxes neuronals diferenciades.

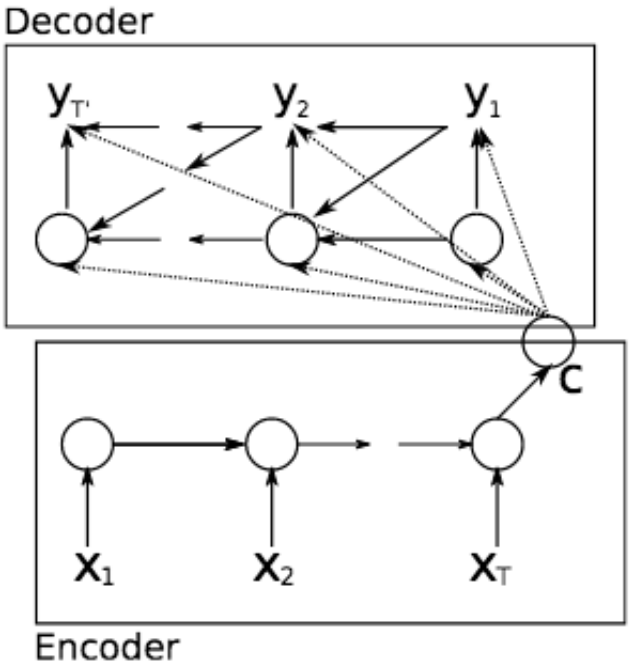

Figura 18: Arquitectura xarxa neuronal *encoder-decoder* amb vector de context c. Imatge extreta de [7]

Encoder

El encoder es una xarxa recurrent usualment basada en LSTM o GRU que processa un a un els elements de la seqüencia d'entrada començant per un token de inici de seqüencia. En cada iteració t el encoder utilitza el element d'entrada x_t i el hidden state h_{t-1} per generar el següent h_t.

$$h_t = \text{Encoder}(x_t, h_{t-1})$$

Cada hidden state h_t pot ser interpretat com un vector que codifica parcialment la seqüencia fins el moment t. Un cop el encoder rep el element final de seqüencia amb $t = \tau$, obtenim el vector de context $c = h_\tau$.

Vector de context

El vector de context $c = h_\tau$ extret per l'encoder sol ser un vector de longitud fixa que es pot entendre com una representació interna abstracte de tota la seqüencia d'entrada. Aquest vector es la representació interna abstracte de la seqüencia d'entrada per part de la xarxa recurrent, de forma similar a com els models interlingua de SMT creaven representacions completes. Aquest vector codifica la informació que el encoder ha decidit mes rellevant i serà la base a partir de la que el decoder reconstruirà la seqüencia en l'idioma de destí.

Decoder

El decoder també es una xarxa recurrent basada en LSTM o GRU però que genera en cada pas un a un els elements de la seqüencia de sortida. La principal diferencia amb el encoder es que pren com a hidden state inicial el vector de context $h_0 = c$ i com a primera entrada y_0 un token d'inici de seqüencia. A partir d'aquest estat inicial el decoder utilitza cada sortida y_t i el hidden state h_t que genera com a entrada per generar la següent sortida y_{t+1}. La generació acaba quan el decoder tria com a sortida el token de final de seqüencia.

$$y_t = \text{Decoder}(y_{t-1}, h_{t-1})$$

El model de decoder es pot millorar fàcilment si en comptes de utilitzar el vector de context únicament com a primera entrada del hidden state, es reutilitza com a entrada per tots els passos.[7]

$$y_t = \text{Decoder}(y_{t-1}, h_{t-1}, c)$$

Mitjançant aquest petit canvi de arquitectura, evitem el principal problema del vector de context: la necessitat de comprimir molta informació en relativament poc espai. Alimentant cada pas del decoder amb el vector de context permet a la xarxa aprendre a codificar i extreure informació rellevant en cada pas d'aquest vector.

Dependències a llarg termini

L'estructura encoder-decoder juntament amb les RNN implementades amb LSTM o GRU van millorar tant la qualitat dels resultats com la capacitat d'afrontar una gran varietat de tasques de seq2seq, convertint-se en el seu moment en l'estat de l'art en molts àmbits. Tot i aixi, la problemàtica de la dependència a llarg termini seguia sent un problema per les xarxes recurrents. [6]

Com es pot veure en la figura 19 la relació entre qualitat de les traduccions mesurada mitjançant la mètrica BLEU, i la longitud de les frases d'entrada, sortida i la seva mitja, te el seu millor resultat en l'interval de 10 a 20 tokens. A partir d'aquí els resultats de la xarxa empitjoren de forma inversament proporcional al increment de la longitud i decauen dramàticament a partir dels 30 tokens. Encara que per l'interval inicial sembla que la xarxa no dona bons resultats, això es pot interpretar mes aviat com una conseqüència de la mètrica utilitza aplicada a seqüencies curtes, mes que un problema del encoder-decoder.

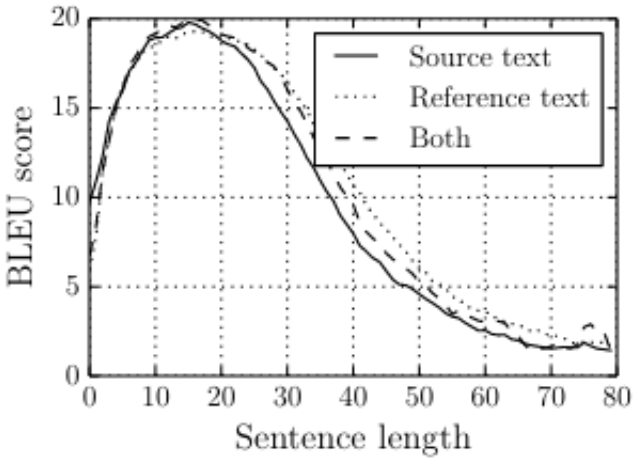

Figura 19: Relació entre BLEU score que indica la qualitat de la traducció i la longitud de les seqüencies d'entrada en les RNN. Imatge extreta de [6].

2.5.4 Mecanisme d'atenció

Davant la problemàtica recurrent del problema de dependències a llarg termini es proposa una arquitectura RNN amb un nou mecanisme per relacionar elements de les seqüencies: l'atenció. [3]

Aquesta arquitectura incorpora una capa RNN bidireccional com encoder, un decoder que emula la cerca d'elements rellevants a traves d'una frase durant la decodificació d'una traducció mitjançant vectors d'atenció i una capa neuronal que genera aquests vectors d'atenció a partir de les sortides del encoder i del vector de context. [6]

El mecanisme d'atenció funciona creant mitjançant una o mes capes neuronals un vector d'atenció a que evita el problema de la codificació i compressió del vector de context c. Aquest vector te la longitud de la seqüencia d'entrada i compleix dos propietats: tots els valors es troben entre 0 i 1, i esta normalitzat, es a dir, sumen 1. Per generar-lo s'utilitza normalment com a capa de sortida una funció Softmax.

Amb aquest vector d'atenció calculem un vector de suma ponderada w amb els hidden states h_t del codificador.

$$w = \sum_{i=1}^{\tau} a_i h_i$$

Per cada pas de descodificació utilitzem el vector de suma ponderada w com entrada pel decoder i per la capa superficial (tanh) per fer una predicció.

Encoder

El encoder esta format per una capa LSTM o GRU bidireccional. Aquesta consisteix en dos RNN, on cada una processa en un ordre diferent la seqüencia d'entrades.

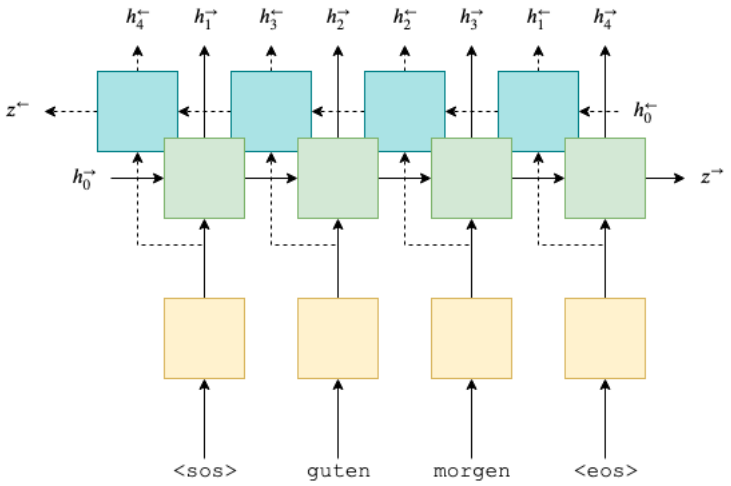

Figura 20: Estructura bàsica d'una capa LSTM bidireccional. Extret de [35]

Aquest processament proporciona dos vectors de context amb informació complementaria i el doble de hidden stats que una capa LSTM com es pot veure a la figura 20. Això millora el context disponible pel decoder, ja que es codifica tant els tokens precedents com posteriors a cada posició.

Capa d'atenció

Una vegada el encoder ha processat la seqüencia la capa d'atenció utilitza tots els hidden states generats pel encoder $H = h_1, ..., h_\tau, h'_1, ..., h'_\tau$, i el hidden state previ del decoder s_{t-1} per generar el vector d'atenció com es pot veure en la figura 21. Pel primer pas del decoder aquest es el vector de context $z = s_{t-1}$.

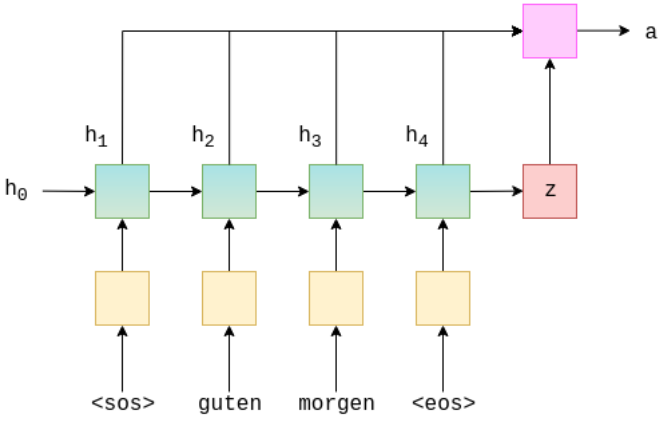

Figura 21: Estructura bàsica del model encoder decoder amb atenció de Jointly learning to align and translate. En aquest pas de la seqüencia es codifica el vector d'atenció a en el rosa mitjançant el vector de context i els hidden states del encoder. Extret de [35]

El primer pas es calcular l'energia entre cada hidden state de H i s_{t-1} aplicant primer

la capa lineal de atenció i una funció d'activació tanh. El concepte d'energia pot ser interpretat com la similitud entre els diversos hidden states i el vector de context, amb valors en l'interval $(1, -1)$.

$$E_t = \tanh(\text{attn}(s_{t-1}, H))$$

Per aconseguir un vector d'atenció de la mida de la seqüencia d'entrada cal reduir la dimensió de la matriu E_t. Per aixo la multipliquem per un tensor v.

$$\hat{a}_t = vE_t$$

I finalment normalitzem el vector d'atenció amb la funció softmax obtenint la atenció $\hat{a}_t$ sobre la seqüencia d'entrada.

$$a_t = \text{softmax}(\hat{a}_t)$$

Decoder

Un cop tenim l'atenció a_t calculem un vector de pesos w a partir de la suma ponderada de l'atenció i els hidden states del encoder. com podem observar en la figura 22

$$w_t = a_t H$$

En cada pas passem al decoder la concatenació de la paraula d' entrada $d(y_t$, la suma ponderada w_t i el hidden state previ $s_{t-1})$.

$$s_t = \text{Decoder}(d(y_t), w_t, s_{t-1})$$

Finalment passem a la capa linear $d(y_t$, la suma ponderada w_t i el hidden state actual s_t per obtenir la predicció $\hat{y}_{t+1}$.

$$\hat{y}_{t+1} = f(d(y_t), w_t, s_t)$$

2.5.5 Dependències a llarg termini

Els models basats en atenció milloren substancialment en el repte de traduir frases mes llargues com es veu en la figura 23. Els mecanismes d'atenció aconsegueixen solucionar en gran mesura la dependència a llarg termini, millorant el model al mateix temps però tot i aixi s'estanquen per seqüencies molt llargues de mes de 100 tokens. [18]

s

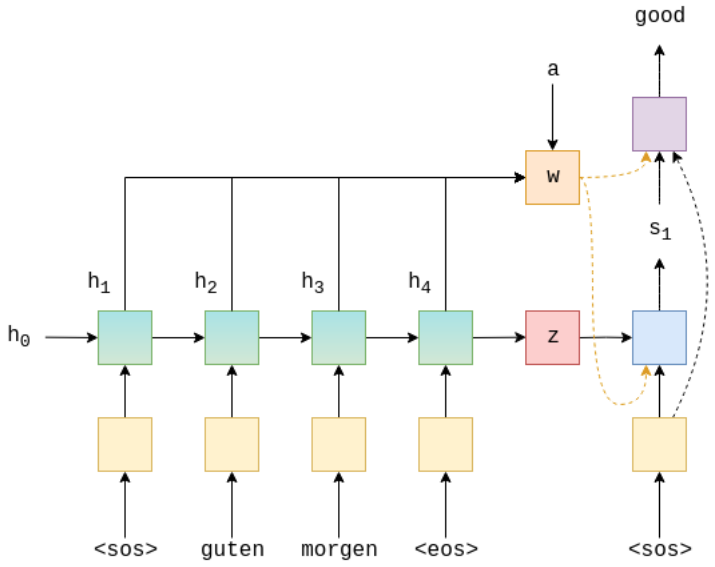

Figura 22: Estructura bàsica del model encoder decoder amb atenció de Jointly learning to align and translate. En aquest pas de la seqüencia s'aplica el vector d'atenció a calculat en la figura 21 a la predicció del decoder mitjançant la suma ponderada amb els pesos w. Extret de [35]

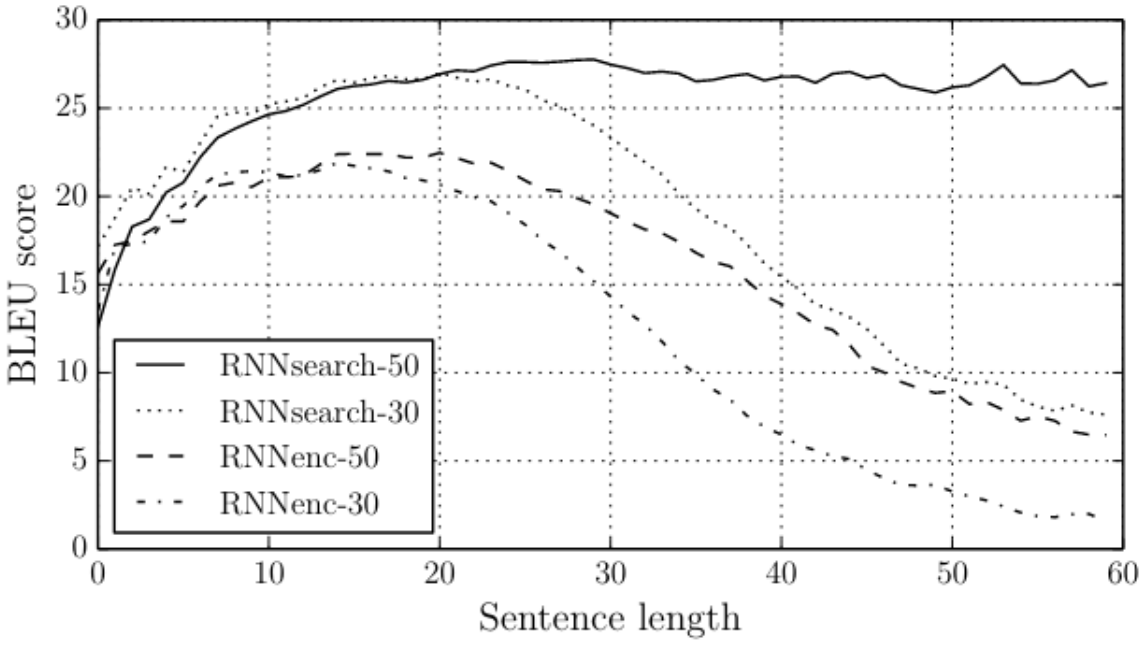

Figura 23: Relació entre BLEU score que indica la qualitat de la traducció i la longitud de les seqüencies d'entrada en seq2seq amb atenció. Imatge extreta de [3]

2.6 Transformers

Al 2017 l'arquitectura *transformer* proposada a l'article 'Attention is all you need' [36] va revolucionar el camp del NMT. Basada totalment en els mecanismes d'atenció i el model encoder-decoder, va abandonar l'arquitectura RNN que havia dominat fins el moment per tornar a una arquitectura DNN.

El conceptes claus del *transformer* son la auto-atenció o *self-attention*, el *multi-head attention* i la codificació posicional o *positional encoding*.

L'arquitectura del *transformer* es basa en els mecanismes d'auto-atenció. L'auto-atenció de forma similar a l'atenció dels models seq2seq permet relacionar els diferents elements o tokens d'una seqüencia entre ells, creant per cada un, un vector d'atenció en relació a tots els elements de la seqüencia.

2.6.1 Arquitectura

El *transformer* esta basat en una estructura general de codificador decodificador.Cada part esta a la vegada formada per N capes iguals compostes per dos components: la capa d'auto-atenció i una capa de feedforward (figura 24) .

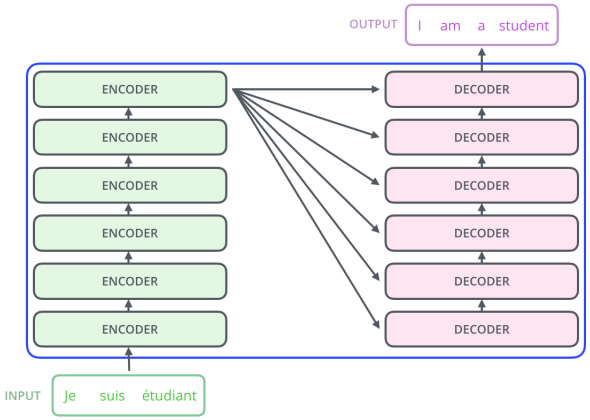

Figura 24: Estructura bàsica del transformer capa d'entrada, N codificadors i decodificadors i capa de sortida. Imatge extreta de [1]

2.6.2 Codificació posicional

Un dels problemes de les xarxes de feed forward respecte les seqüencies es com codificar la informació posicional. En el transformer això s'aconsegueix amb una capa que codifica mitjançant una senyal periòdica les posicions de cada token en la seqüencia tant de les entrada del codificador com del decodificador.

2.6.3 Codificador

La funció dels codificadors es codificar la informació d'atenció sobre l'entrada pels decodificadors. El procediment es generar en la capa d'auto-atenció les matrius abstractes de Query Q, Key K i Value V a partir de l'entrada X. Aquests valors

generats a partir de 3 matrius entrenades en la capa WQ, WK i WV ens serveixen per generar la matriu d'atenció que passarem al següent encoder:

$$Z = Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}}V)$$

2.6.4 Multihead attention

Un conjunt de matrius (WQ, WK, WV) s'anomena a *Attention head*, i en cada capa encoder o decoder del model tenim múltiples. Una *Attention head* pot codificar la rellevància per cada token de tota la resta però múltiples permeten diversificar les relacions que es codifiquen de la seqüencia d'entrada

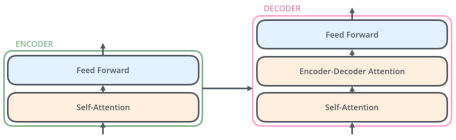

Figura 25: Estructura bàsica del encoder i el decoder amb mecanismes de auto atenció. Imatge extreta de [1]

2.6.5 Decodificador

Cada decodificador esta compost per tres elements principals: la capa d'auto atenció, un mecanisme d'atenció cap als codificadors i una capa de feed-forward.

La funció dels decodificadors es similar als codificadors, però mitjançant la capa d'atenció extreuen informació rellevant de les matrius d'atenció generades. Finalment l'ultim decoder es seguit per una transformació lineal i una capa softmax per produir les probabilitats del vocabulari (figura 26).

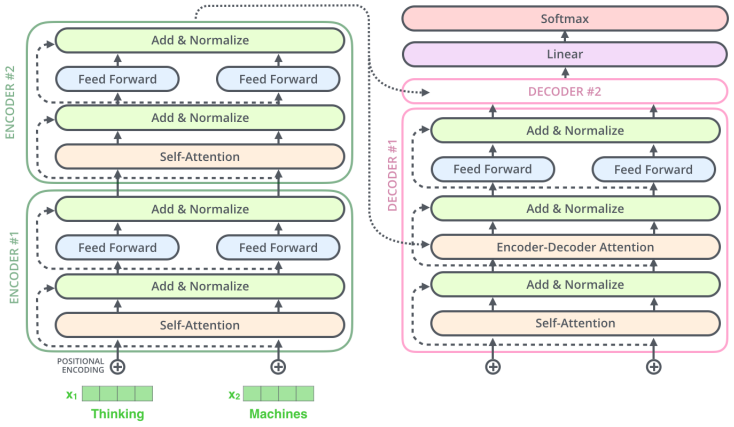

Figura 26: Estructura completa del transformer simplificant a N=1 les capes de encoders i decoders. Imatge extreta de [1]

2.7 Alineament de paraules

La tasca d'alineació de paraules a partir de corpus bilingües alineats a nivell de frase es una tasca tradicional del NLP per la importància que te en el desenvolupament de models de llenguatge i models de traducció per el SMT. Aquesta tasca consisteix en la creació de les correspondències mínimes del procés de traducció entre els elements de dos frases paral·leles. Aquestes correspondències per tant han de basar-se en el context de la frase i no en traduccions idealitzades prèvies.

El primer pas en aquesta tasca es tokenitzar les sentencies a alinear per tal de transformar el alineaments de paraula a paraula a token a token. Aquests alineaments formen grafs bipartits fàcils de visualitzar com matrius que permeten una supervisió del funcionament intern dels sistemes de traducció automàtica.

En la tasca d'alineació de paraules s'han de tenir en compte el tipus d'alineació que volem aconseguir:

- Alineació única/múltiple: depenen de l'objectiu de la tasca les alineacions poden ser de un a molts tokens o únicament de un a un. En la figura 27 podem veure un exemple d'alineament múltiple entre el token 'implemented' i la seva alineació amb els tokens 'mis', 'en' i 'applications'.
- Alineació nul·la/no nul·la: en molts casos els tokens no tenen contraparts clars en el altre idioma o directament no tenen cap relació directe. En la figura 27 veiem una alineació nul·la del token 'And'.
- Alineació suau/dura: en ocasions interessa obtenir alineacions suaus entre els diferents elements, basades en el calcul de les probabilitats d'alineació entre els diferents elements. Aquest tipus d'alineació es la mes útil per inferir el funcionament intern dels models i extreure'n informació. Per altra banda, la alineació dura sol ser mes fàcil d'implementar de forma manual.

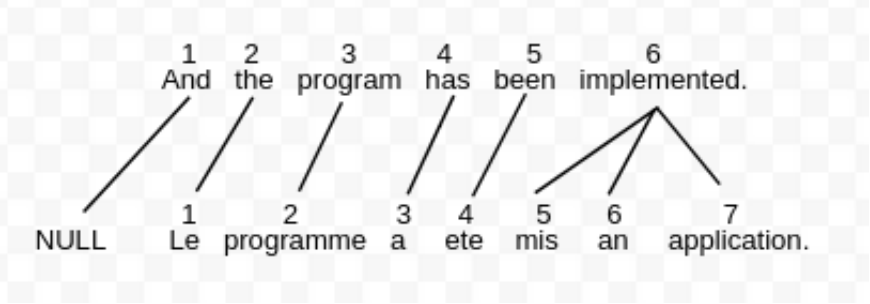

Figura 27: Estructura bàsica d'una alineació paraula a paraula entre dos frases en Anglès i Francès formant un graf bipartit.

2.7.1 Alineació mitjançant SMT

L'alineament de paraules ha estat una tasca fonamental en SMT degut a la necessitat de corpus bilingües alineats a nivell de paraula. Actualment els models estadístics basats en models de llenguatge de n-grams com fast align [10] o GIZA++ son capaços de produir alineaments amb taxes elevades de precisió i baixos recalls i

son utilitzats per comparar el rendiment en alineació de models de NMT. Aquests algoritmes empren mètodes no supervisats d'aprenentatge per relacionar seqüencies de tokens en diferents idiomes, fent en certa mesura una tasca prèvia a la de traducció. El principal problema d'aquests mètodes estadístics es la dificultat per utilitzar el context de les seqüencies per portar a terme l'alineament, sent susceptibles a les ambigüitats del llenguatge.

2.7.2 Alineació mitjançant NMT

Els models de NMT basats en atenció no sempre compleixen el role d'alineament de paraula. En "Six Challenges for Neural Machine Translation"[18] de 2017 es compara l'atenció d'un model de RNN amb atenció amb l'estat de l'art de SMT: fast align. En aquest estudi, encara que els mecanismes d'atenció funcionen de forma general com es pot veure en la figura 28, no sempre prediuen alineaments suaus correctes i poden tenir errors sistemàtics a l'hora d'identificar correctament els elements mes relacionats entre dues seqüencies de tokens com es pot veure en la figura 29. Els errors mes comuns detectats son desplaçaments de l'alineament a la dreta o a l'esquerra i falsos positius repetits amb diversos tokens.

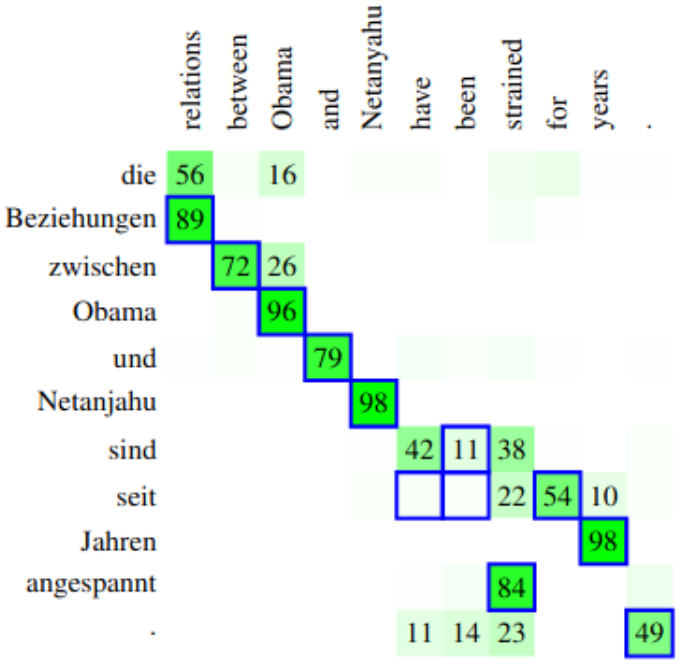

Figura 28: Matriu d'alineació entre dos frases en Anglès i Alemany. Es pot visualitzar l'alineació de fast align amb blau i l'atenció de 0 a 1 en escala de verd. Imatge extreta de [18].

Com alternativa a l'atenció per generar alineaments tenim el mètode basat en la prominència o saliencia. [9]De forma similar que en el concepte de saliencia o importància d'un input en funció del gradient en el camp de la visió per computador, en l'anàlisi d'imatges en les xarxes neuronals podem definir la saliencia com el gradient parcial de la funció de cost respecte l'input o píxel [31] i es proposa extendre

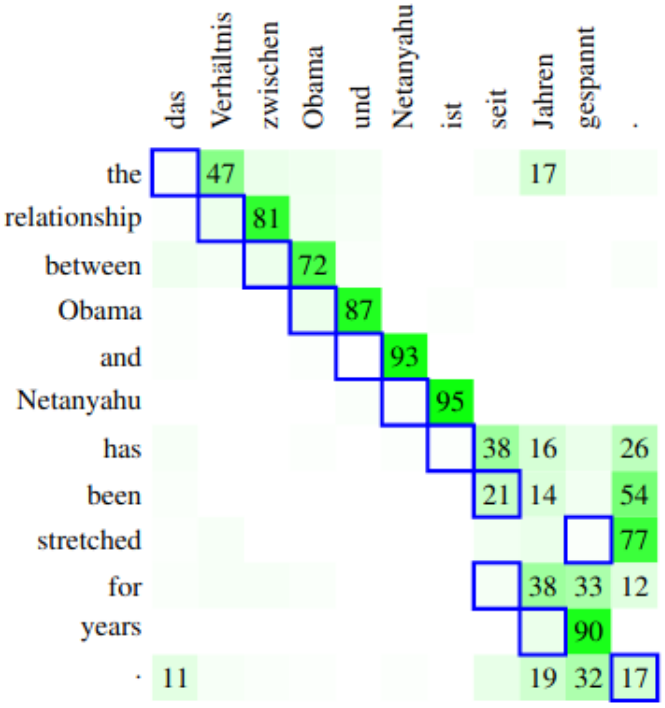

Figura 29: Matriu d'alineació entre dos frases en Anglès i Alemany. Es pot visualitzar l'alineació de fast align amb blau i l'atenció de 0 a 1 en escala de verd. Imatge extreta de [18].

aquesta idea al NMT mitjançant una analogia entre píxels i la capa d' embedding de tokens. Aquesta tècnica retorna alineaments comparables a fast align i a tècniques basades en l'atenció, demostrant un gran potencial. Avui en dia es qüestiona l'ús dels mètodes basats en atenció i es proposa la saliencia com alternativa general al problema de l'alineament.[4]

Per altre banda en l'article del 2020 "End-to-End Neural Word Alignment Outperforms GIZA++"[38] s'utilitza un model basat en l'arquitectura *transformer* entrenat en tasques de NMT per predir alineaments de paraules de forma no supervisada. En aquest estudi l'alineació a partir de l'atenció supera el model GIZA++ en tots els conjunts de dades.

3 Mètodes i propostes

Proposem implementar un model de xarxa neuronal recurrent basada en encoder-decoder amb atenció. Un cop implementada la arquitectura bàsica procedirem a extreure la matriu d'atenció generada i juntament amb les inferències del model en mode de decodificació forçada procedirem a generar alineaments amb les dades.

La necessitat de GPUs per portar a terme l'entrenament dels models ens ha portat a treballar en els entorns virtuals de Google Colab i Kaggle.

Finalment entrenarem el model en la tasca de traduir del anglès al francès. Per fer-ho utilitzarem el conjunt de dades proporcionat en el HLT/NAACL del 2003 del corpus dels Hansards. Amb el model entrenat procedirem a avaluar els resultat, compararem els resultats amb les dades de la tasca compartida [19] i extraurem conclusions.

La implementació d'aquest model es pot trobar a la següent adreça: `https://github.com/polsafont/TFG`.

3.1 Xarxa neuronal recursiva amb atenció

Pel model enfocat en xarxes recurrents ens hem basat en l'article 'Neural machine translation by jointly learning to align and translate' [3] seguint el model encoder-decoder amb una capa d'atenció. Per guiar-nos en aquesta part del treball hem utilitzat recursos, guies i tutorials en línia. [27] [35] [33]

```
Seq2Seq(
  (encoder): Encoder(
    (embedding): Embedding(5398, 64)
    (rnn): GRU(64, 128, bidirectional=True)
    (fc): Linear(in_features=256, out_features=128, bias=True)
    (dropout): Dropout(p=0.6, inplace=False)
  )
  (decoder): Decoder(
    (attention): Attention(
      (attn): Linear(in_features=384, out_features=128, bias=True)
      (v): Linear(in_features=128, out_features=1, bias=False)
    )
    (embedding): Embedding(6358, 64)
    (rnn): GRU(320, 128)
    (fc_out): Linear(in_features=448, out_features=6358, bias=True)
    (dropout): Dropout(p=0.6, inplace=False)
  )
)
```

Figura 30: Estructura interna del model amb les diferents capes.

3.1.1 Encoder

Pel encoder tenim dues parts diferenciades: primer una capa d'entrada de *embedding*, una segona capa recurrent bidireccional basada en GRU, i una capa lineal de sortida.

L'encoder accepta els index dels tokens de la seqüencia a traduir i retorna el vector de context i les sortides de cada pas que seran utilitzades per la capa d'atenció i el decodificador.

Embedding

La capa d'embedding d'entrada esta estructurada en una capa lineal. La funció de l'embedding es transformar el one hot encodding del vocabulari d'entrada a un vector mitjançant *word embedding*. Definim la capa amb la funció que accepta com a entrada els index dels tokens i retorna el corresponent vector.

GRU bidireccional

La capa GRU bidireccional [30] s'encarrega de la recursivitat en el encoder. Aquesta capa te en realitat 2 capes de GRU independents que processen la seqüencia d'entada de dreta a esquerra i de esquerra a dreta. Aquesta capa accepta el vector del encoding i genera un sortida amb l'estat ocult h_t per cada element de la seqüencia i un vector de context amb l'ultim. Un cop processada l'entrada, concatenem els 2 vectors de context.

Capa lineal de sortida

Aquesta capa lineal agafa els vectors concatenats de context i els retorna amb la mida correcte per ser utilitzats per el decoder.

3.1.2 Atenció

La capa d'atenció calculara el vector d'atenció a cada pas partir de les sortides del encoder i de la ultima sortida del decoder (en el cas del primer pas, el vector de context).

En el modul d'atenció tenim 2 capes lineals. La primera accepta les entrades, aplicant una funció tanh calculem l'energia. La segon capa transforma el tensor resultant a la longitud de la seqüencia d'entrada, doncs volem prestar atenció a tots els elements. Finalment apliquem una softmax per normalitzar els valors i retornem l'atenció.

3.1.3 Decoder

Pel decoder tenim una estructura molt similar al encoder excepte que la capa GRU no es bidireccional.

Per una banda tenim la capa d'embedding, la capa de recursivitat de GRU i la capa lineal de sortida. El decoder accepta a cada pas el vector d'atenció, la sortida del decoder del pas anterior i el hidden state o la seva pròpia sortida del pas anterior.

El decoder calcula a cada pas el pes ponderat w a partir dels hidden state del encoder i del vector d'atenció.

Embedding

La capa d'embedding pren els valors calculats per la capa de sortida el decoder, on el primer element serà el token <SOS>. De la mateixa manera que la capa d'embedding del encoder, s'encarrega de transformar els index dels tokens en vectors.

GRU

La capa recursiva de GRU rep les dades de la capa de embedding, del vector de pes ponderat w, del estat previ del decoder (en el primer pas el vector de context). La capa recurrent retorna la sortida que anirà a la capa de sortida.

Capa lineal de sortida

La capa de sortida s'encarregara de fer la predicció de sortida amb la informació de la sortida de la capa d'embedding, el pes w i la sortida de la GRU.

3.1.4 Hiperparàmetres

El nombre de paràmetres del model varia en funció de la mida dels vocabularis de entrada o SRC i de sortida o TRG que definim i del mida de les capes. Finalment ens hem decidit per un model petit de 4.011.222 paràmetres entrenables per evitar problemes d'overfitting que teníem amb models mes grans. La inicialització d'aquests paràmetres es realitza amb un distribució normal $\mu = 0$ i $\cdot\sigma = 0.01$

Batchsize = 64

Drop-out encoder i decoder = 0.5

Optimitzador: Adam

Loss function: crossEntropyLoss

Clip = True

3.2 Generació d'alineaments

Decodificació forçada

Similar al *forced teaching*, obliguem a la decodificador a prendre la predicció correcte en el pas anterior com a entrada en cada pas sense importar la predicció anterior. Això ens permet al conèixer la sortida inferir amb menys error la matriu d'atenció a la frase objectiu. Si no apliquem *force decoding* la predicció s'allunya molt de la traducció del *gold standard* i la matriu d'atenció no serviria per la tasca de generar alineaments.

Criteri de selecció d'alineaments

Per xarxes recurrents d'una sola direcció, es a dir, enfocades a la tasca de traduir origen a destí un mètode comú es la extracció dels alineaments utilitzant com a criteri el valor mes alt d'atenció en el vector.[38] Per xarxes bidireccionals, on disposem de

dos vectors d'atenció, un per cada direcció de la traducció es pot fer un producte element a element o de Hadamard entre els dos tensors per obtenir una matriu d'atenció que maximitza les probabilitats en les dos direccions. [38]

Pel nostres models ens hem decidit per utilitzar dos tècniques. Per una banda implementem un algoritme avariciós on prenem el valor mes gran del vector d'atenció per cada paraula com l'alineament mes probable per retornar en funció d'un llindar un alineament segur (S) o probable (P).

I en segon lloc utilitzarem un criteri mixta basat en llindars i cerca avariciosa, on depenen dels valors de triatge, classificarem els diferents valors en alineaments segurs i probables. Mitjançant tres valors (segur = 0.5, probable = 0.3 i nul = 0.1) generarem les alineacions. Pels valors superior al llindar d'atenció 0.5 els assignarem un alineació Segura, per valors superiors a 0.3 Probable i en cas de no obtenir cap alineació buscarem el valor màxim i si supera el llindar de nul li assignarem una probabilitat P. En aquest cas tindrem un llindar mínim de no alineat a partir del qual no generarem cap predicció.

Els valors dels llindars han sigut triats a base de observar els valors retornats pels vectors d'atenció i mitjançant prova i error amb l'objectiu de generar alineacions variades per poder comparar les diferents taxes d'error entre P i S amb el gold standard.

4 Resultats experimentals i discussió

4.1 Conjunts de dades

4.1.1 Descripció

Les dades utilitzades son el corpus bilingües anglès-francès alineat a nivell de frase dels Hansards o transcripcions del 36é Parlament de Canada juntament amb un petit conjunt de dades alineades manualment a nivell de paraula. Proporcionats per parlament de Canada, tractades i alineades per Ulrich Germann `https://www.isi.edu/division3/natural-language/download/hansard/` i obtingudes des de la web de la tasca compartida del HLT-NAACL 2003 `https://web.eecs.umich.edu/~mihalcea/wpt/` a traves de l'enllaç `https://web.eecs.umich.edu/~mihalcea/wpt/data/English-French.training.tar.gz`.

L'origen del corpus bitextual dels Hansards es degut al bilingüisme del país. Les actes son traduïdes tant al Francès com al Anglès per part del equips de traductors i lingüistes del parlament. Aquestes actes tenen un gran valor jurídic i polític, i per tant tenen un alt nivell de qualitat.

4.1.2 Dades d'entrenament

La mida total dels corpus es de 1130105 línies i el conjunt de test te un total de 447 línies.

Per tal de tenir un conjunt de dades valides d'entrenament caldrà realitzar 3 tasques principals: transformar el text al format de tokens, netejar les dades per millorar la seva validesa eliminant elements erronis o *outliers* i finalment preparar les dades per l'entrenament dels models.

El primers pas consisteix a transformar les cadenes en conjunts de tokens. Per fer-ho utilitzarem Spacy, un llibreria de processament de llenguatge natural.

Un cop tokenitzats a nivell de paraula podem veure dos mètriques molt importants del corpus per l'entrenament dels models: la longitud de les seqüencies i les diferencies de longitud entre les parelles.

Analitzant les longituds de les frases (figura 31) trobem que existeixen sentencies que superen els 100 elements. Observant en mes detall en el rang d'entre 200 i 1000 elements en la figura 32 podem veure com existeixen elements propers als 1000 tokens. Per tant un pas important serà eliminar el subconjunt de sentencies extremadament llargues, decidint-nos per eliminar les seqüencies on una de les parelles tingui mes de 30 elements.

Per altre banda, eliminem tots aquells elements que puguin estar desaparellats, es a dir, mal alineats. Per fer-ho observarem les diferencies de longitud entre les parelles (figura 33) trobant que la mantenen una distribució normal dintre de valors raonables però amb bastants elements fora de lloc o *outliers*.

En aquest cas eliminarem tots aquells elements on la diferencia superi els 10 elements entre les parelles.

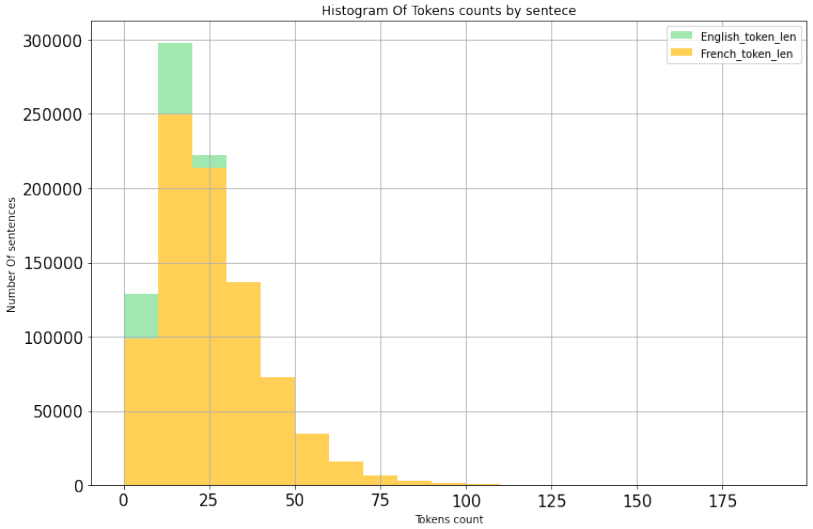

Figura 31: Histograma de distribució de longituds de les frases del corpus en anglès i francès.

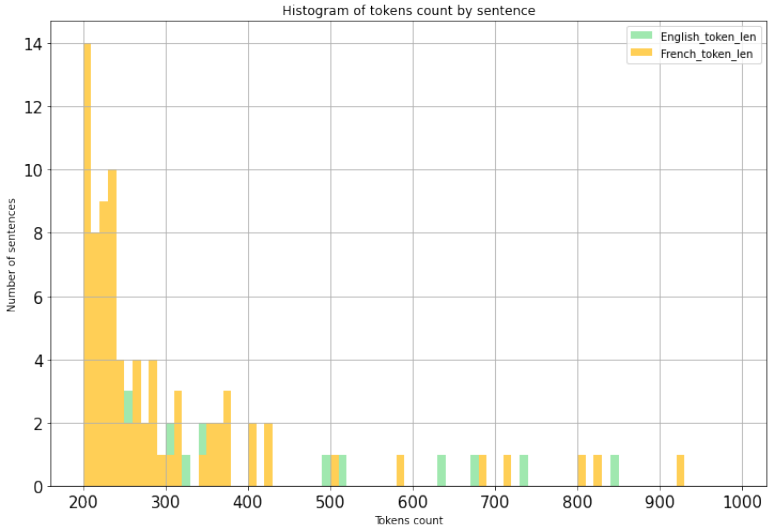

Figura 32: Histograma de distribució de longituds de les frases del corpus en anglès i francès.

Finalment eliminem parelles d'elements duplicats. Sobre el total 1,130,105 sentencies inicials ens quedem amb un subconjunt de dades de 651,181.

El corpus original té un vocabulari en anglès de 57604 paraules i en francès de 78,392, el subconjunt de dades processades te un vocabulari en angles de 46,092 i en francès de 63,140.

Per analitzar els vocabularis ens fixem en la distribució de paraules per nombre d'aparicions que podem veure en les figures 34 i 35. Aquesta segueix la llei de Zipf o distribució de cua llarga:

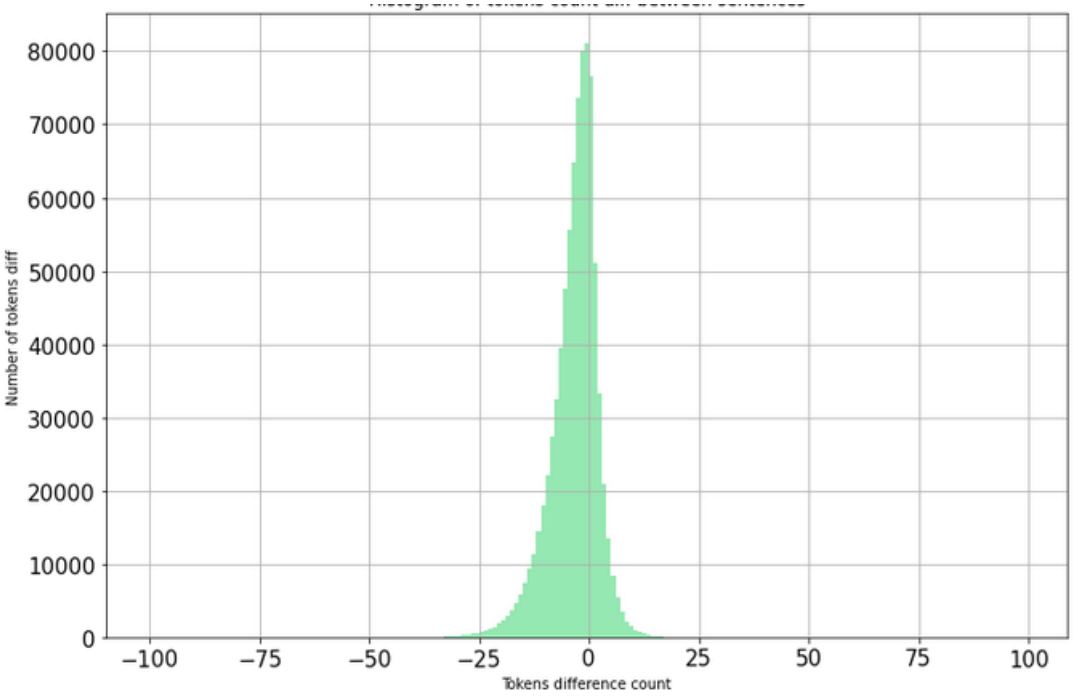

Figura 33: Histograma de distribució de la diferencia de longituds de les frases del corpus en anglès i francès.

$$P_n = 1/n^a$$

on P_n representa el nombre de repeticions d'una paraula en la posició n-èsima (quan les paraules s'ordenen de major a menor freqüència) i a és una constant propera a 1.

Com podem veure la distribució segueix una estructura de cua llarga, on la una gran majoria de tokens tindran freqüències d'aparició molt baixes. Per tal d'escollir el vocabulari final caldrà tenir en compte de triar una freqüència mínima que permeti conservar un vocabulari representatiu, d'entre 2 i 10.

Finalment separem el conjunt de dades en un subconjunt d'entrenament i un de validació.

4.1.3 Dades de test

El *gold standard* bilingüe anglès-francès esta compost per 4 fitxers. Per una banda dos fitxers amb les frases tokenitzades i separades amb espais, i ordenades amb un identificador. I per l'altre dos fitxer d'alineaments. Aquest fitxers s'organitzen en línies amb el identificador, el nombre del token origen, token destí i la probabilitat d'alineació. Aquesta probabilitat pot tenir dos valors: S per alineament Segur i P per alineament Probable.

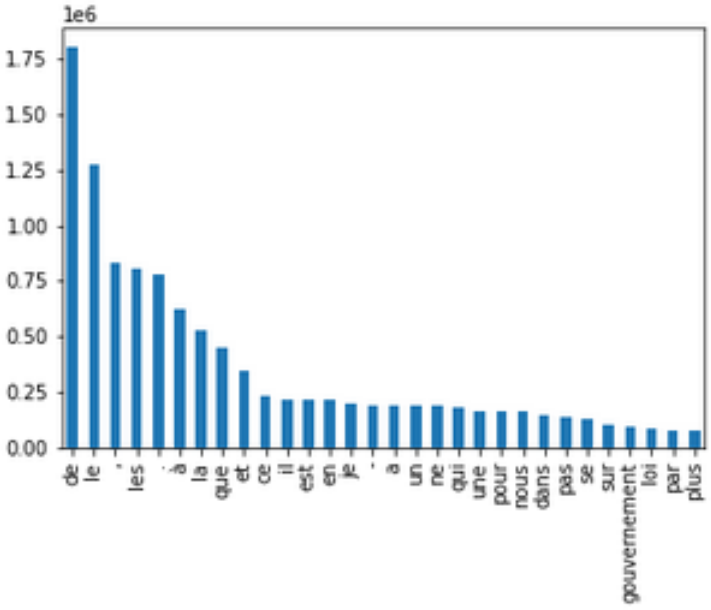

Figura 34: Histograma de distribució de freqüència d'aparició de cada paraula única del corpus francès.

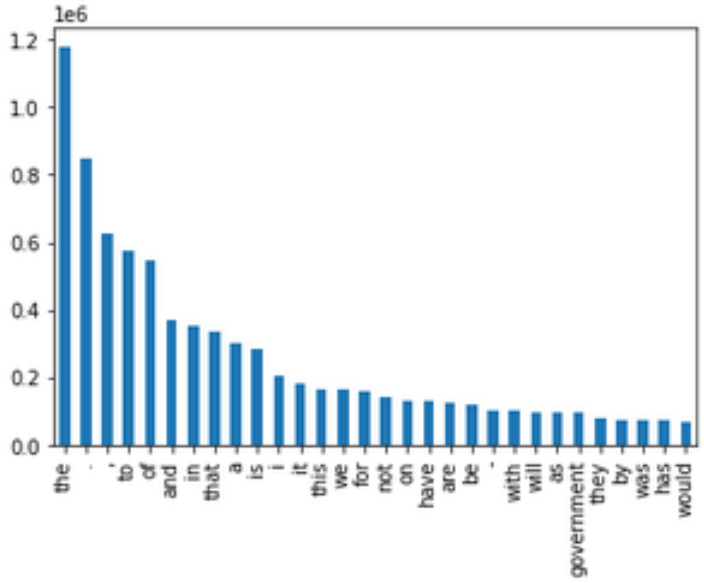

Figura 35: Histograma de distribució de freqüència d'aparició de cada paraula única del corpus anglès.

4.2 Mètriques d'avaluació

Com a mètriques d'avaluació utilitzarem BLEU per les traduccions i el AER per els alineaments. Aquestes dos mètriques son àmpliament utilitzades en el camp del NMT.

4.2.1 BLEU

BLEU (*Bilingual Evaluation Understudy*) [26] es un algoritme per generar mètriques d'avaluació de la qualitat d'una traducció automàtica d'un idioma a un altre. La idea central del algoritme es que una traducció te mes qualitat com major es la semblança amb una traducció de referencia. Es a dir, com mes propera es la traducció automàtica a una traducció professional, millor es la traducció. Aquest algoritme no avalua la correcció gramatical o si la traducció es comprensible. Simplement mesura la semblança a nivell de n-grames.

BLEU calcula les puntuacions a nivell de frase comparant amb un conjunt de traduccions professionals i calcula la mitja de tot el corpus per donar la puntuació final. El valor de BLEU es un nombre entre 0 i 1, i normalment es multiplica per 100 com a nivell de referencia.

El algoritme cerca la precisió en n-grames entre les traduccions i la frase de referencia:

$$p = \frac{\text{n-grames comuns entre traducció i referencia}}{textn - gramestraducci}$$

Per evitar estratègies com la repetició de n-grames comuns, és te en compte el nombre màxim d'aparicions de cada n-grama en la referencia per calcular la precisió. Un altre problema es la comparació entre sentencies de diferent longitud, especialment si la traducció es molt més curta que la de referencia. Per evitar puntuacions altes es penalitza la diferencia de longitud c de la traducció respecte la longitud r de la referencia amb la formula:

$$b_p = \begin{cases} 1 \text{ si } c > r \\ e^{1-r/c} \text{ si } c <= r \end{cases}$$

Finalment calculem el BLEU:

$$\text{BLEU} = b_p \exp\left(\sum_{n=1}^{N} w_n \log p_n\right)$$

On N es l'ordre de n-grames que analitzarem, b_p la penalització per brevetat, w_n son els pesos de cada n-grames i p_n es la precisió.

Per calcular la qualitat de les traduccions utilitzarem BLEU sobre el conjunt de test amb per n-grames de fins a N=4 amb pesos iguals de $w_n = 0.25$.

4.2.2 Alignment Error Rate

En quant a l'avaluació de la tasca ens trobem amb el problema de com comparar els alineaments generats. Una mètrica proposada en el camp del SMT es la comparació d'alineaments de forma indirecte amb la comparació dels resultats de dos traduccions

d'un mateix model basada cadascuna en una alineació. Aquest sistema però es lent i no ens dona una referencia sobre la qual comparar els resultats.

La solució a aquest problema es l'ús de corpus bitext alineats a nivell de paraula de forma manual. Aquest tipus de corpus ens permet calcular de forma automàtica el error d'aquests sistemes a nivell estadística. [19] Donat un alineament predit A compost per A_S i A_P i un del estàndard d'or G format per alineaments G_S i G_P on P es Probable i S Segur, calculem la precisió P_T, el reclam o *recall* R_T, la mesura F_T on T pot ser el conjunt de P o el conjunt de S, i el AER pels dos conjunts.

La precisió es la fracció de veritables positius del total de positius predits. En aquest cas el conjunt respecte el total de prediccions.

$$P_T = \frac{|A_T \cap G_T|}{||A_T}$$

El reclam o sensibilitat es la proporció de casos positius identificats correctament respecte el total.

$$R_T = \frac{|A_T \cap G_T|}{|G_T|}$$

La mesura F es una mesura que combina la precisió i el reclam, i es la seva mitja harmònica.

$$F_T = \frac{2P_T R_T}{P_T + R_T}$$

El AER [23] es una mètrica comú per mesurar els alineaments. Combina la precisió i el reclam de manera que un alineament perfecte ha de tenir tots els alineaments segurs S i pot tenir alguns dels alineaments possibles P.

$$\text{AER} = 1 - \frac{|A_P \cap G_S| + |A_P \cap G_P|}{|A_P + G_S|}$$

Per calcular la qualitat dels alineaments utilitzarem la precisió, el reclam, la mesura F i el AER sobre el conjunt de test amb el gold standard d'alineació no nul·la.

4.2.3 Avaluació de resultats

Overfitting

A l'hora d'entrenar el model l'*overfitting* ha estat un problema recurrent. Encara que el model convergia ràpidament en tots els casos, l'error de validació no disminuïa i en certs casos (figura 36) augmentava.

Figura 36: Entrenament amb 100.000 dades i 20 milions de paràmetres interns.

Això es deu al gran nombre de paràmetres intern dels primers models. Incrementant el nombre de dades i reduint el nombre de paràmetres interns de forma significativa a (figura 37) el model comença a generalitzar per el conjunt de validació de forma correcte.

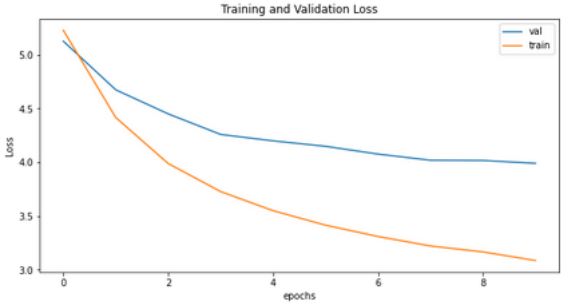

Figura 37: Entrenament amb 500.000 dades i 4 milions de paràmetres interns.

Un dels principals reptes ha estat escollir la mida dels vocabularis ja que tenen un gran impacte en la mida del model i per tant en el overfitting i l'aprenentatge i en els resultats de les mètriques de traducció especialment BLUE. Amb un diccionari del voltant de 10.000 paraules per cada idioma hem trobat un bon equilibri

Matriu d'atenció i generació d'alineaments

En quant a la generació de matrius d'atenció en force decoding i la generació d'alineaments, els resultats varien molt. Encara que la matriu d'atenció crea patrons interessants i aproxima en gran mesura les relacions entre paraules, cal un model molt mes potent i entrenat amb mes dades per ser fiable. Tot i aixi els resultats son sorprenents.

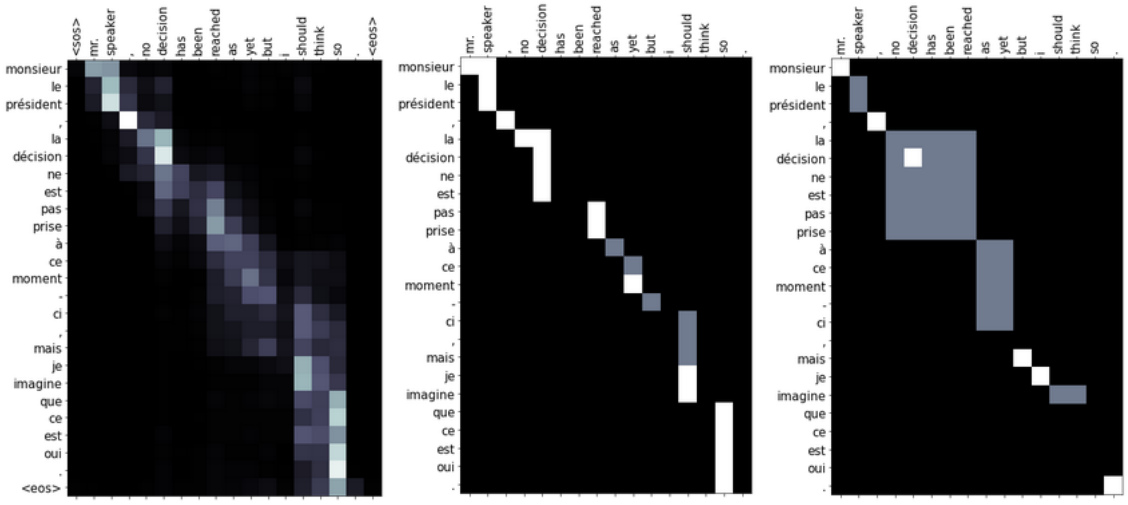

Figura 38: Comparació entre matriu d'atenció esquerra, matriu d'alineament centre i estàndard d'or a la dreta.

Sistema	P_S	R_S	F_S	P_P	R_P	F_P	AER
Greedy	20.61%	8.07%	11.43%	24.99	11.13%	15,27%	73.02%
Threshold	21.16%	28.13%	22.42%	41.92%	24.55%	27.58%	51.23%

Taula 1: Resultats, Precisió i Reclam, mesura F per alineaments Segurs i Probables, i AER.

4.3 Resultats e la qualitat de les traduccions

Resultats BLEU

Els resultats del BLEU ens han permès estimar el correcte entrenament del model al incrementar les dades d'entrenament. Encara que la millora de la puntuació no ha estat tan bona com esperàvem al augmentar el conjunt de dades d'entrenament, ha estat consistent amb el creixement logarítmic esperat. [14]

Figura 39: Resultats BLEU en funció de la mida de les dades d'entrenament.

4.4 Resultats de la qualitat dels alineaments

Resultats AER

Els resultats basats en AER han donat errors molt elevats (figura 40). Únicament per els models entrenats amb 500.000 i 600.000 parelles de frases hem rebaixat el AER fins el 50% amb l'enfoc de llindars i alineament mes suau. L'enfoc greedy com a línia de referencia no ha superat el 70% en cap cas. Com podem veure en la taula 1 els millors resultats son per la versió Threshold amb una precisió del 41.92% per alineaments Probables.

Comparats amb els resultats del HLT/NAACL del 2003 el model s'ha quedat a 10 punts del pitjor model estadístic en quant a AER (figura 40).

Per alineaments de seqüencies llargues, la generació d'alineaments basats en l'atenció tenen un rendiment molt baix. La dispersió i normalització de l'atenció entre molts possibles valors amb la funció Softmax la converteix en poc significativa (figura 41).

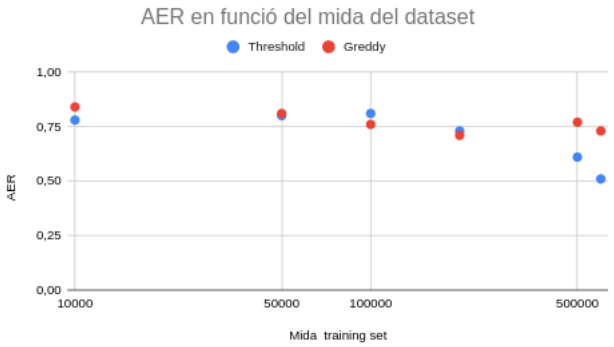

Figura 40: Comparació resultats AER en funció de la mida de les dades d'entrenament amb l'algoritme greedy en vermell i l'algoritme threshold en blau.

Sistema	AER
Attention Greedy	73.02%
Attention Threshold	51.23%
BiBr.EF.1	28.23%
BiBr.EF.4	28.01%
BiBr.EF.7	29.38%
Ralign.EF.1	18.50%
UMD.EF.1	38.47%
XRCE.Base.EF.1	16.23%
XRCE.Nolem.EF.2	8.93%
XRCE.Nolem.EF.3	8.53%

Taula 2: Resultats AER del HLT/NAACL (2003) amb models propis Attention Greedy i Attention Threshold. Tots amb recuroso limitats i alineament no nul. Dades extretes de [19].

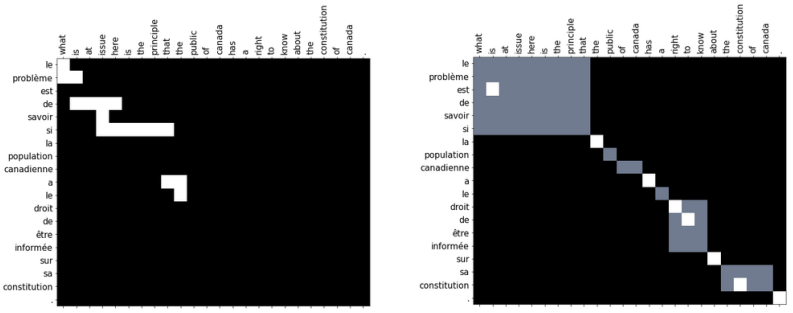

Figura 41: Alineament d'or a l'esquerra i alineament generat a la dreta

Però en frases curtes millora molt i aproxima molts bons resultats amb correlacions interessants (figura 42).

Encara que els resultats generals dels algoritmes de generació d'alineaments han estat inferiors als esperats, caldria investigar més a fons les causes d'aquest comportament. Per una banda podria ser causat per un overfitting en el aprenentatge per part dels models. Per altre banda al netejar en excés les dades d'entrenament (hem partit de

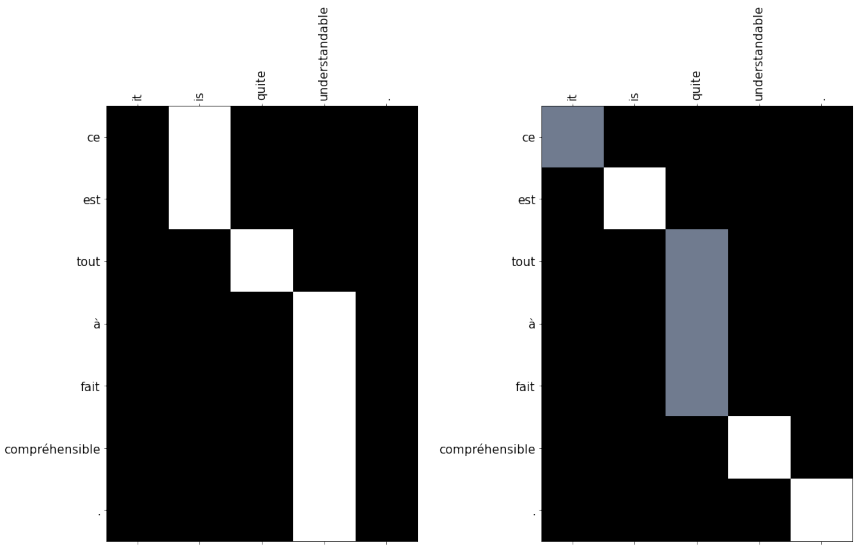

Figura 42: Comparació resultats d'alineació greedy a l'esquerra i alineament amb el gold standard a la dreta.

1.100.000 frases per acabar entrenant amb aproximadament la meitat) i per tant la utilització d'un conjunt de dades reduït podria ser la causa d'aquest baix rendiment.

5 Conclusions i propostes

En aquest projecte hem investigat el camp del processament del llenguatge natural i la traducció automàtica de la mà de les xarxes neuronals, un camp fascinant amb molta historia i futur per endavant.

El primer objectiu d'aquest treball, l'aprenentatge sobre la teoria i les bases del *Deep Learning*, ha sigut tot un repte. A pesar dels grans recursos disponibles sobre aquesta temàtica que han facilitat la recerca aquest es un camp complex en innovació constant i on sempre hi han nous articles o arquitectures que el revolucionen. Dintre del deep learning he pogut endinsar-me en el món de les xarxes recurrents, una arquitectura desconeguda per a mi amb grans possibilitats i en especial dels mecanismes d'atenció.

El segon objectiu, el disseny i implementació de xarxes recurrents te una gran complexitat tècnica i matemàtica. La implementació de la xarxa recurrent, les primeres proves amb text, les correccions d'errors, les optimitzacions i iteracions han sigut gran part de la feina i aprenentatge del projecte. En principi el tercer objectiu era la implementació de l'arquitectura *transformer* però malgrat avançar molt en el seu disseny no ha sigut possible incorporar-la al treball final. Encara que els resultats de les mètriques d'alineació no han sigut tan satisfactoris com podríem haver desitjat, no queda cap dubte que els mecanismes d'atenció son una revolució en el camp de les xarxes neuronals.

Finalment com a futurs desenvolupaments en aquest àrea hi han dos grans temes a explorar. L'ús de la saliencia com a eina complementaria a l'atenció aixi com de mecanismes auto explicables te un gran potencial. [4] I per l'altre, la generalització dels mecanismes d'atenció mes enllà de seqüencies textuals en l'article *"Perceiver: General Perception with Iterative Attention"* [17] proposa una nova revolució en el camp del *Deep Learning*.

Bibliografia

[1] J Alammar. The illustrated transformer[blog post], 06 2018. consulta: 20 de gener del 2022.

[2] David Alvarez-Melis and Tommi S. Jaakkola. Towards robust interpretability with self-explaining neural networks, 2018.

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2016.

[4] Jasmijn Bastings and Katja Filippova. The elephant in the interpretability room: Why use attention as explanation when we have saliency methods?, 2020.

[5] Peter Brown, John Cocke, Stephen Della Pietra, Vincent Dellapietra, Fredrick Jelinek, John Lafferty, Robert Mercer, and Paul Roossin. A statistical approach to machine translation. *Computational Linguistics*, 16, 07 2002.

[6] Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches, 2014.

[7] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation, 2014.

[8] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling, 2014.

[9] Shuoyang Ding, Hainan Xu, and Philipp Koehn. Saliency-driven word alignment interpretation for neural machine translation, 2019.

[10] Chris Dyer, Victor Chahuneau, and Noah A. Smith. A simple, fast, and effective reparameterization of IBM model 2. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 644–648, Atlanta, Georgia, June 2013. Association for Computational Linguistics.

[11] Daniel C. Elton. Self-explaining ai as an alternative to interpretable ai. *Lecture Notes in Computer Science*, page 95–106, 2020.

[12] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[13] Alex Graves. Generating sequences with recurrent neural networks, 2014.

[14] Alon Halevy, Peter Norvig, and Fernando Pereira. The unreasonable effectiveness of data. *IEEE Intelligent Systems*, 24(2):8–12, 2009.

[15] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors, 2012.

[16] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

[17] Andrew Jaegle, Felix Gimeno, Andrew Brock, Andrew Zisserman, Oriol Vinyals, and Joao Carreira. Perceiver: General perception with iterative attention, 2021.

[18] Philipp Koehn and Rebecca Knowles. Six challenges for neural machine translation, 2017.

[19] Rada Mihalcea and Ted Pedersen. An evaluation exercise for word alignment. 03 2004.

[20] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.

[21] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality, 2013.

[22] Marvin Minsky. Perceptrons : an introduction to computational geometry, 1988.

[23] Franz Josef Och and Hermann Ney. A comparison of alignment models for statistical machine translation. In *COLING 2000 Volume 2: The 18th International Conference on Computational Linguistics*, 2000.

[24] C. Olah. Understanding lstm network, 08 2015. consulta: 20 de gener del 2022.

[25] Cathy O'Neil. *Weapons of math destruction: How big data increases inequality and threatens democracy*. Crown, New York, first edition edition, 2016.

[26] Kishore Papineni, Salim Roukos, Todd Ward, and Wei jing Zhu. Bleu: a method for automatic evaluation of machine translation. pages 311–318, 2002.

[27] Sean Robertson. Nlp from scratch: Translation with a sequence to sequence network and attention, 05 2021. consulta: 20 de gener del 2022.

[28] Sebastian Ruder. An overview of gradient descent optimization algorithms, 2017.

[29] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, October 1986.

[30] Mike Schuster and Kuldip Paliwal. Bidirectional recurrent neural networks. *Signal Processing, IEEE Transactions on*, 45:2673 – 2681, 12 1997.

[31] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps, 2014.

[32] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[33] Tim Sullivan. Introduction to recurrent neural networks (rnns), 05 2020. consulta: 20 de gener del 2022.

[34] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks, 2014.

[35] Ben Trevett. Pytorch seq2seq, 07 2018. consulta: 20 de gener del 2022.

[36] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[37] Paul J. Werbos. Generalization of backpropagation with application to a recurrent gas market model, January 1988.

[38] Thomas Zenkel, Joern Wuebker, and John DeNero. End-to-end neural word alignment outperforms giza++, 2020.