UNIVERSITAT POLITÈCNICA DE VALÈNCIA

MIARFID Official Master's Degree in Artificial Intelligence, Pattern Recognition and Digital Imaging

Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia

Traducción automática neuronal sensible al contexto

TRABAJO FIN DE MÁSTER

Máster Universitario en Inteligencia Artificial, Reconocimiento de Formas e Imagen Digital

Autor: Àngel Andújar Carracedo

Tutor: Francisco Casacuberta Nolla

Curso 2020-2021

Resum

Els sistemes de traducció automàtica neuronal normalment tradueixen els documents frase per frase de forma aïllada ignorant la informació extra que pot aportar el context entre oracions d'un mateix document. Així doncs, el fet que els sistemes siguen conscients del context ajuda a millorar la qualitat de les traduccions millorant la seua coherència i evitant errors en casos d'ambigüitat.

En aquest treball s'estudia els beneficis que aporta que els sistemes neuronals de traducció automàtica siguen sensibles al context, és a dir, que en el moment de realitzar la traducció d'un text, el model obtinga per a cada oració informació addicional de les oracions que l'envolten. Per a això, es realitzen diferents integracions en l'arquitectura del model neuronal *Transformer* a partir de l'ús d'un multicodificador encarregat de capturar el context.

Amb la finalitat de mostrar la utilitat d'aquest enfocament, es desenvolupa una àmplia experimentació variant el conjunt de dades i el paradigma de multicodificador integrat en el model de traducció. Per consegüent, els resultats exposen que els sistemes sensibles al context milloren la qualitat de les traduccions.

Paraules clau: Traducció automàtica neuronal; Traducció automàtica neuronal sensible al context; Traducció a nivell de document; Aprenentatge profund.

Resumen

Los sistemas de traducción automática neuronal normalmente traducen los documentos frase por frase de forma aislada ignorando la información extra que puede aportar el contexto entre oraciones de un mismo documento. Así pues, el hecho de que los sistemas sean conscientes del contexto ayuda a mejorar la calidad de las traducciones mejorando su coherencia y evitando errores en casos de ambigüedad.

En este trabajo se estudia los beneficios que aporta que los sistemas neuronales de traducción automática sean sensibles al contexto, es decir, que en el momento de realizar la traducción de un texto, el modelo obtenga para cada oración información adicional de las oraciones que la rodean. Para ello, se realizan diferentes integraciones en la arquitectura del modelo neuronal *Transformer* a partir del uso de un multicodificador encargado de capturar el contexto.

Con el fin de mostrar la utilidad de este enfoque, se desarrolla una amplia experimentación variando el conjunto de datos y el paradigma de multicodificador integrado en el modelo de traducción. Por consiguiente, los resultados exponen que los sistemas sensibles al contexto mejoran la calidad de las traducciones.

Palabras clave: Traducción automática neuronal; Traducción automática neuronal sensible al contexto; Traducción a nivel de documento; Aprendizaje profundo.

Abstract

Neural machine translation systems usually translate documents sentence by sentence in isolation, ignoring the extra information that context can provide between sentences in the same document. Thus, the fact that systems are aware of the context helps to improve the quality of the translations by improving their coherence and avoiding errors in cases of ambiguity.

In this project we study the benefits of context-aware neural machine translation models, that is, when translating a text, the model obtains for each sentence additional information from sentences that surround it. For this, different integrations are made in the architecture of the Transformer neural model for the use of a multi-encoder in charge of capturing the context.

In order to expose the usefulness of this approach, extensive experimentation is developed by varying the data set and the multi-encoder paradigm integrated into the translation model. Therefore, the results show that context-aware systems improve the quality of translations.

Key words: Neural machine translation; Context-Aware Neural Machine Translation; Document-Level Translation; Deep learning.

Índice general

Índice de figuras

Índice de tablas

CAPÍTULO 1
Introducción

Desde los orígenes de la humanidad, el lenguaje es la capacidad que tienen las personas para comunicarse y expresarse entre ellas. Sin embargo, la inmensa pluralidad de idiomas existentes en el mundo conlleva un gran desafío para dicha comunicación. Así pues, la traducción apareció con el objetivo de facilitar la interlocución entre individuos con bases lingüísticas diferentes.

La idea de emplear ordenadores para la traducción entre idiomas nació en la segunda guerra mundial en la que se utilizaron computadoras para descifrar códigos enemigos. En un primer momento, se pensaba que la traducción automática iba a ser una tarea que se resolvería fácilmente, no obstante, el avance de las investigaciones en este campo fue más lento de lo esperado por lo que la financiación se redujo considerablemente [20].

En la actualidad, se contemplan diferentes contextos donde la traducción entre idiomas se ha convertido en un factor esencial. Así, por ejemplo, en la educación para realizar la enseñanza de lenguas extranjeras; en organismos e instituciones políticas como la Organización de Naciones Unidas la cual está compuesta por 193 naciones y adopta 6 lenguas oficiales para comunicarse entre ellas; en sanidad o incluso en plataformas digitales.

A pesar de evolución que ha experimentado la traducción automática en la última década, los sistemas no son capaces de resolver la tarea de traducción en su totalidad, por lo que en la mayoría de los escenarios dicha tarea es realizada por las personas. No obstante, en el presente se continúa la búsqueda de nuevos enfoques que mejoren las prestaciones de los traductores automáticos.

El aprendizaje profundo está revolucionando la forma en que se construyen los sistemas de traducción automática actualmente dando lugar a la traducción automática neuronal. Así pues, los sistemas de traducción automática neuronal tradicionales realizan la traducción de un documento a nivel de frase,es decir, considerando de forma aislada cada una de las frases existentes en dicho documento. Este hecho puede perjudicar la calidad de la traducción especialmente en términos de coherencia, cohesión y consistencia. Con el fin de solventar este inconveniente, han surgido nuevas aproximaciones en la traducción automática neuronal construyendo modelos sensibles al contexto, es decir, que al realizar la traducción de un documento, el modelo obtenga para cada frase información adicional de las frases que la rodean.

Este trabajo de final de máster presenta un enfoque desde la traducción automática neuronal. Para ello, se desarrolla un sistema de traducción automática neuronal que sea sensible al contexto. Este sistema utiliza una arquitectura con multicodificador y , por último, se realiza la experimentación sobre diferentes conjuntos de datos en diversos pares de lengua.

1.1 Objetivos

El objetivo principal de este proyecto consiste en el estudio y la experimentación de un modelo de traducción automática neuronal sensible al contexto, utilizando para ello las oraciones que forman parte de un documento con el fin de otorgar una traducción de una oración correctamente. El trabajo se divide en tres objetivos:

- Recopilación de corpus que permitan la traducción a nivel de documento.
- Implementación de la arquitectura multicodificador.
- Traducción automática neuronal sensible al contexto.
  - Uso de la arquitectura multicodificador con diferentes configuraciones.
  - Evaluación de las configuraciones y análisis de los resultados.

1.2 Estructura

El presente trabajo está constituido por siete capítulos. Seguidamente, se detallan los temas tratados en cada capítulo:

En el **capítulo 1. Introducción**, se realiza la introducción del tema que se aborda a lo largo del proyecto, además de la descripción de los objetivos a cumplir.

En el **capítulo 2. Traducción automática**, se expone de forma teórica al lector las diferentes aproximaciones de la traducción automática existentes y sus características principales.

En el **capítulo 3. Traducción automática neuronal sensible al contexto**, se muestra de forma teórica al lector los diversos enfoques existentes de la traducción automática neuronal sensible al contexto y, además se describe la arquitectura del modelo de traducción dependiente del contexto que se ha empleado en este proyecto.

En el **capítulo 4. Marco experimental**, se presenta las tecnologías que han sido empleadas en el desarrollo del trabajo, los corpus que se han recopilado, la forma de evaluar las traducciones producidas, el hardware disponible en el entrenamiento de los modelos y se exponen los experimentos que se han realizado.

En el **capítulo 5. Resultados experimentales**, se realiza la evaluación del modelo sobre los diferentes conjuntos de datos y se muestra un estudio de los resultados obtenidos, además de la exposición de los problemas originados y las medidas tomadas con el fin de paliarlos.

En el **capítulo 6. Conclusiones y trabajos futuros**, se presenta las conclusiones obtenidas a partir de la experimentación realizada y se exponen diferentes caminos a tomar en próximos trabajos relacionados con traducción neuronal a nivel de documento.

Asimismo, a estos seis capítulos se referencia la bibliografía que se ha consultado en la realización del trabajo.

CAPÍTULO 2
Traducción automática

La traducción automática es un área de la lingüística computacional que consiste en realizar la conversión de un idioma de entrada a otro de salida a partir de ordenadores. Así pues, las primeras referencias al concepto de traducción automática se remontan al siglo 17, donde filósofos como Leibniz reflexionaron acerca de la búsqueda de relaciones entre palabras de lenguajes distintos [29]. Sin embargo, el área de investigación en la traducción automática emergió a mediados del siglo 20 y, hasta el presente, se han originado diferentes enfoques con el objetivo de solventar el problema de la traducción adecuadamente.

En relación con la perspectiva empleada para abordar la tarea de la traducción automática, cabe destacar que puede ser ordenada bajo diferentes criterios. En primer lugar, con respecto al tipo de la entrada del sistema, a través del habla o de texto; en segundo lugar, en base a la tecnología empleada en la traducción, es decir, emplear modelos basados en reglas o modelos establecidos en corpus; y, por último, según el tipo de aplicación en la que se van a emplear las traducciones, aquellas que consultan la traducción en una base de datos, aplicaciones que realizan una traducción medida y son mejoradas por usuarios a partir de la posedición, otras que efectúan traducciones interactivas con la ayuda del usuario y, las que están completamente automatizadas [26].

2.1 Sistemas basado en reglas

Los sistemas basados en reglas analizan la sintaxis del idioma de entrada y se ayudan de reglas lingüísticas y diccionarios para traducir el significado al idioma de salida. Por consiguiente, hay que señalar que dichas reglas determinan cómo realizar la traducción y, además, son creadas por traductores humanos. Por lo tanto, estos sistemas requieren de un gran esfuerzo en su desarrollo y son poco flexibles.

Con respecto a la forma en la que realizan las traducciones, hay que resaltar que se ejecutan en tres pasos, el de análisis, el de transferencia y el de generación. Así pues, el paso de análisis consiste en extraer información del texto de entrada, por otro lado, el paso de transferencia trata de convertir el resultado del paso anterior en una representación abstracta con el objetivo de reducir la complejidad del análisis. El paso de generación consiste en generar el texto de salida. Por último, cabe destacar que los sistemas basados en reglas dependiendo de la información asignada a los pasos de análisis pueden clasificase en tres enfoques diferentes, el directo, el de transferencia y el de interlingua [29].

2.2 Sistemas basado en corpus

Los sistemas basados en corpus se fundamentan en el uso de conjuntos de datos compuestos por ejemplos de traducciones de una lengua a otra. Por ello, estos sistemas se basan en la convicción de que no existen soluciones preestablecidas para el problema de la traducción, pero la mayoría de las soluciones se encuentran en textos ya traducidos por profesionales. Con relación al enfoque que se aplique en estos sistemas, existen diferentes tipos de traducción automática basada en corpus como los sistemas basados en ejemplos y los estadísticos.

Por un lado, la traducción automática basada en ejemplos emplea un conjunto de pares de traducciones como su fuente principal de conocimiento. Con el objetivo de producir las traducciones, este tipo de traducción realiza dos pasos, el de comparación y el de recombinación. En primer lugar, el paso de comparación consiste en analizar la frase a traducir y partir la entrada en fragmentos para encontrar los equivalentes entre los ejemplos existentes en el conjunto de entrenamiento; en segundo lugar, el paso de recombinación trata de combinar los ejemplos equivalentes y generar la frase completa traducida [26].

Por el otro lado, los sistemas de traducción automática estadística realizan las traducciones utilizando modelos estadísticos. Por consiguiente, este tipo de traducción necesita de una gran cantidad de pares de textos debido a que dichos textos son usados para estimar los parámetros de los modelos estadísticos y, por lo tanto, sean capaces de inferir una nueva traducción con un texto de entrada diferente [4].

2.3 Traducción automática estadística

Los sistemas de traducción automática estadística tratan el problema de la traducción automática desde un enfoque estadístico, ya que aplican modelos estadísticos en el proceso de traducción. Así pues, cabe destacar que dichos modelos utilizan los pares de textos existentes en los conjuntos de datos para estimar sus parámetros.

En relación con la traducción automática estadística, cabe destacar que se puede entender como la búsqueda de la hipótesis de traducción más probable dada una frase de origen. Por ello, desde una perspectiva formal, dada una oración de entrada x en un idioma de entrada, el problema de la traducción automática consiste en encontrar su correspondiente traducción y en el idioma de salida. Este problema es formalizado de la siguiente forma [5]:

$$\hat{y} = arg \max_{y} Pr(y|x) \tag{2.1}$$

No obstante, estos modelos suelen ser combinados con el modelo log-linear para el término $Pr(y|x)$, por lo que el problema se modela de la siguiente manera [19]:

$$\hat{y} = arg \max_{y} \left\{ \sum_{\mathrm{n}=1}^{\mathrm{N}} \lambda_{\mathrm{n}} * \log(\mathrm{f_n}\,(\mathrm{y}, \mathrm{x}))\right\} \tag{2.2}$$

en el que $f_n\,(y, x)$ se corresponde con cualquier característica que sea importante para la traducción, λ_n representa los pesos asignados a la combinación log-linear para cada característica y, por último, N es la cantidad de características.

Así pues, el proceso de construir un sistema de traducción automática que sigue la regla de decisión de Bayes implica abordar tres problemas: el problema del modelado,

sobre cómo estructurar las dependencias de las oraciones de los idiomas de entrada y salida; el problema del entrenamiento, es decir, cómo estimar los parámetros del modelo a partir del conjunto de datos; el problema de búsqueda, en relación a cómo realizar la búsqueda del mejor candidato de traducción entre todas las posibles oraciones en el lenguaje objetivo.

2.4 Traducción automática neuronal

El aprendizaje profundo está revolucionando la forma en que se construyen los sistemas de traducción automática en la actualidad. Si bien los modelos basados en la traducción automática estadística otorgan buenos resultados en la tarea de traducción, en el presente la traducción automática neuronal, mediante el uso de redes neuronales, ha logrado mejorar el rendimiento de los sistemas anteriores en una gran cantidad de pares de lenguas convirtiéndose así en principal enfoque para abordar el problema de la traducción automática [32].

La traducción automática neuronal también adopta el marco probabilístico, debido a que su objetivo consiste en estimar una distribución condicional desconocida,$P(y|x)$, dado el conjunto de datos D, donde x e y son variables aleatorias que representan las frases de entrada y las frases de salida. Con respecto a la forma de modelar el problema de la traducción, hay que señalar que la traducción automática neuronal puede modelarlo a diferentes niveles, ya sea a nivel de documento, de párrafo o a nivel de oración.

En relación con el caso de la traducción a nivel de oración, se entiende que tanto la entrada como la salida son oraciones. Por lo tanto, asumiendo una frase de entrada $x = \{x_1, \ldots, x_S\}$, en el que S se corresponde con el número total de frases de entrada, y una frase de salida $y = \{y_1, \ldots, y_T\}$, en el que T comprende el número total de oraciones de salida, empleando la regla de la cadena, la distribución condicional puede factorizarse como:

$$\hat{y}_1^T = \arg\max_{T, y_1^T} \prod_{t=1}^{T} Pr_\theta(y_t | y_1^{t-1}, c(x_1^S)) \tag{2.3}$$

donde y_t representa la palabra traducida actual, que es generada a partir de las palabras anteriores y_1^{t-1} que han sido traducidas previamente con un tipo de representación denotado por la función c de la oración de origen x_1^S, y empleando los parámetros del modelo θ.

En la actualidad, la gran mayoría de los modelos de traducción automática neuronal emplean el modelo codificador-decodificador junto al uso de arquitecturas neuronales profundas.

2.4.1. Redes neuronales recurrentes

Los codificadores y decodificadores son los elementos clave de las arquitecturas de traducción automática. Existen una gran cantidad de métodos para desarrollar codificadores y decodificadores potentes, y entre ellos están las redes neuronales recurrentes (*RNN*). La característica principal de estas redes es que incorporan la retroalimentación entre neuronas por lo que se consigue crear temporalidad, permitiendo a la red que tenga memoria.

Así pues, las *RNN* contienen un estado oculto h y una salida opcional y que depende de una frase de longitud variable $x = \{x_1, \ldots, x_T\}$, dicho estado oculto es actualizado para cada periodo de tiempo t:

$$h_t = f_h(x_t, h_{t-1}) \tag{2.4}$$

$$y_t = f_o(h_t) \tag{2.5}$$

en el que h_t se corresponde con el estado oculto de la red en el tiempo t. Respecto a la representación que se escoja para f_h,f_o y h_t se han originado diferentes arquitecturas, entre ellas cabe destacar las *RNN* de Elman [12]. La red de Elman puede ser descrita formalmente como [2]:

$$h_t = f(Vx_t + Wh_{t-1}) \tag{2.6}$$

$$y_t = g(Uh_t) \tag{2.7}$$

en la que V representa la entrada de los pesos ocultos, W los pesos recurrentes de las capas ocultas y U los pesos de salida ocultos. Por otro lado, f y g son las funciones de activación de las neuronas ocultas y de las de salida respectivamente.

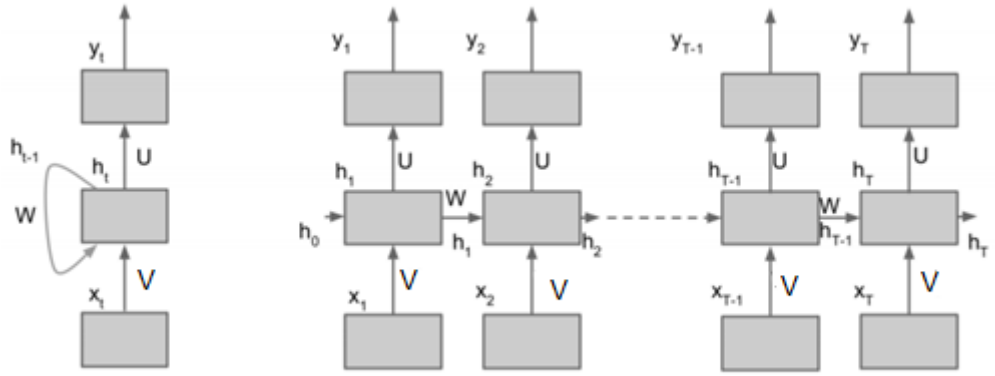

Figura 2.1: Arquitectura *RNN* de Elman [2].

No obstante, *RNN* sufren de un grave problema en el entrenamiento debido a que los gradientes retropropagados tienden a crecer enormemente o a desvanecerse con el paso del tiempo, con el fin de resolver dicho problema se suelen combinar con capas ocultas como las *LSTM*, *long short-term memory*, [16] o las *GRU*, *gated recurrent unit*, [10]. Por tanto, las *LSTM* son una extensión de las *RNN*, que principalmente amplían su memoria para recordar sus entradas durante un largo período de tiempo. Esta memoria se puede comprender como una celda que la neurona *LSTM* decide si almacenar o eliminar información dentro de ella en base al peso que tenga asignado dicha información. Además, en una neurona *LSTM* hay tres puertas a las celdas de información, la de entrada, la de olvido y la de salida. Estas puertas determinan si se permite o no una nueva entrada, se elimina la información porque no es importante o se deja que afecte a la salida en el período de tiempo actual.

Por último, hay que destacar la existencia de la arquitectura compuesta por redes neuronales recurrentes bidireccionales (*BRNN*), las cuales conectan dos capas ocultas en direcciones contrarias a la misma salida [30]. De esta forma de aprendizaje profundo generativo, la capa de salida es capaz de obtener información de estados pasados y futuros simultáneamente. Además, esta técnica es muy útil cuando es combinada con el uso de *LSTM*.

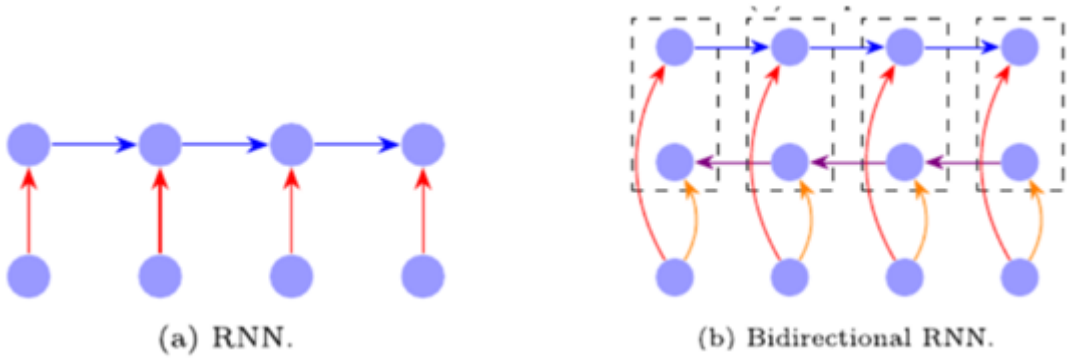

Figura 2.2: Comparación entre *RNN* y *BRNN* [32].

2.4.2. Arquitectura codificador-decodificador

El principal objetivo de este modelo es codificar una oración en la lengua de entrada de longitud variable en una representación vectorial de longitud fija y, decodificar el vector de longitud fija en una oración en la lengua objetivo de longitud variable. Normalmente, esta arquitectura suele utilizar símbolos como «<bos>» y «<eos>» para indicar el comienzo y el final de una secuencia respectivamente. Además, cabe destacar que el enfoque codificador-decodificador consiste en cuatro componentes básicos, las capas de *embedding*, las redes de codificación y las de decodificación y una capa de clasificación [32].

En relación con la capa de embedding, hay que señalar que representa el concepto de representación continua de las oraciones de entrada. Esta capa mapea un vector de símbolos discretos x_t en un vector continuo $x_t \in \mathbb{R}^d$, en el que la d indica el tamaño del vector.

Por otro lado, la red codificadora se encarga de mapear los *embedding* de entrada en representaciones continuas ocultas. Asimismo, con el objetivo de que el codificador aprenda las representaciones expresivas, tiene que ser capaz de modelar el orden y las dependencias que existen en el idioma de la frase de entrada. Las *RNN* son una opción apropiada para modelar las oraciones de longitud variable, por lo tanto, el codificador es una *RNN* que lee cada simbolo de una secuencia de entrada x, el estado oculto de la *RNN* cambia con respecto a la ecuación 2.8. Después de leer el final de secuencia, el estado oculto de la *RNN* es un resumen c de toda la secuencia de entrada. Por ello, la computación que envuelve al codificador puede ser representada como:

$$h_t = RNN_{COD}(x_t, h_{t-1}) \tag{2.8}$$

y

$$c = q(h_1, \ldots, h_S) \tag{2.9}$$

Así pues, aplicando iterativamente la función de transición de estado RNN_{COD} sobre la frase de entrada, se puede emplear el estado final h_S, siendo S el último símbolo de la oración de entrada, como la representación de toda la oración y utilizarla para introducirla en el decodificador. Además de que c es el vector de contexto generado en base a la secuencia de estados ocultos,mientras q es una función no lineal.

Con respecto a la red decodificadora, cabe destacar que puede ser contemplada como un modelo de lenguaje condicionado por h_S. En consecuencia, el decodificador extrae

la información necesaria de la salida del codificador y, de igual modo, modela las dependencias de larga distancia entre las palabras objetivo. Dado el símbolo de comienzo $y_0 = < bos >$ y el estado inicial $s_0 = h_S$, el decodificador *RNN* comprime la historia decodificada $\{y_0, \ldots, y_{t-1}\}$ en un vector de estados $s_t \in \mathbb{R}^d$:

$$s_t = RNN_{DEC}(y_{t-1}, s_{t-1}, c) \tag{2.10}$$

Por último, la capa de clasificación se encarga de predecir la distribución de los tokens de salida. La capa de clasificación normalmente es una capa lineal con una función de activación *softmax*. Por lo tanto, suponiendo que el vocabulario del idioma de salida es V, y $|V|$ es el tamaño del vocabulario, dada la salida del decodificador $s_t \in \mathbb{R}^d$, la capa de clasificación mapea h a un vector z en el espacio del vocabulario $\mathbb{R}^{|V|}$ con un mapa lineal. Una vez mapeado, se aplica la función *softmax* para afianzar que el vector es una probabilidad válida:

$$softmax\,(z_i) = \frac{\exp(z_i)}{\sum_{k=1}^{|V|} \exp(z_k)} \tag{2.11}$$

donde se utiliza z_i para referir el i-ésimo componente del vector z.

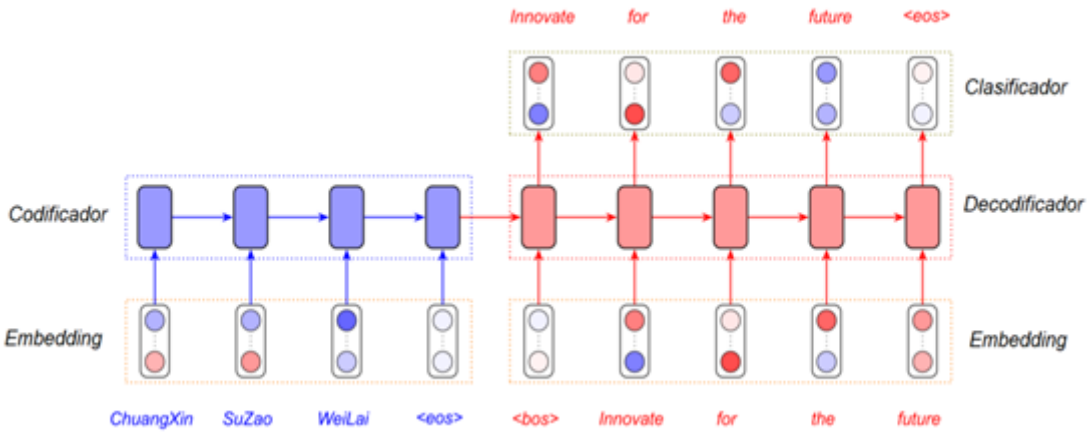

Figura 2.3: Arquitectura codificador-decodificador para la traducción automática neuronal [32].

2.4.3. Mecanismo de atención

La crítica al empleo de *RNN* es que requieren realizar un recorrido largo de la oración de entrada, es decir de palabra por palabra, lo que consume un tiempo considerable y limita la paralelización. Con el objetivo de solucionar este problema, existe un enfoque alternativo que consiste en el reemplazo de las *RNN* por redes neuronales convolucionales . No obstante, el uso de modelos basados en redes convolucionales presenta el inconveniente de que tienen una ventana de contexto limitada para enriquecer las representaciones de palabras [20].

Con el modelo *Transformer* [36] surge un componente capaz de utilizar un contexto amplio y que pueda estar altamente paralelizado, el mecanismo de autoatención. Así pues, mientras que el mecanismo de atención considera las asociaciones entre cada palabra de entrada y cualquier palabra de salida y lo usa para crear una representación vectorial de toda la secuencia de entrada, la idea existente detrás de la autoatención es extender el mecanismo de atención en el codificador. Por tanto, dicha extensión consiste en que en vez de calcular la asociación entre una palabra de entrada y una de salida, la

autoatención computa la asociación entre cualquier palabra de entrada y cualquier otra palabra de la entrada.

Una forma de contemplar el mecanismo de autoatención es que clarifica la representación de cada palabra de entrada enriqueciéndola con palabras de su contexto que ayuden a desambiguarla. Por consiguiente, las consultas, llaves y valores se obtienen normalmente a través de un mapa lineal de las representaciones de entrada. Por lo tanto, el producto punto escalado del mecanismo de autoatención, suponiendo que la entrada consiste en consultas Q y claves K de dimensión d_k y valores V de dimensión d_v, su formulación general se concibe como:

$$Autoatencion\,(Q,K,V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.12}$$

Es necesario recordar que, en este caso se instancia a partir del uso $q = s_{t-1}\, K_t = h_t, V_t = h_t$. Además, si se desea calcular la atención para múltiples consultas, estas se pueden empaquetar en la matriz de consultas Q, por lo que el cálculo se puede expresar en términos de multiplicación de matrices.

Por otro lado, cabe destacar la existencia de un mecanismo multicabezal de autoatención que es una extensión de la red de autoatención, pero con múltiples cabezales paralelos. Con respecto a cada cabezal, cabe destacar que se encarga de atender la información desde diferentes subespacios de los vectores de valor. Como resultado, el mecanismo multicabezal de autoatención puede realizar transformaciones más flexibles que el mecanismo con un solo cabezal.

$$MultiCabezal\,(Q,K,V) = Concat\,(cabezal_1,\ldots,cabezal_h)\,W^o \tag{2.13}$$

donde

$$cabezal_i = Autoatencion(QW_i^Q, KW_i^K, VW_i^V) \tag{2.14}$$

en las que las proyecciones son matrices de parámetros $\mathrm{W}_i^Q \in \mathbb{R}^{d_{modelo}*d_k}$, $W_i^K \in \mathbb{R}^{d_{modelo}*d_k}, W_i^V \in \mathbb{R}^{d_{modelo}*v}, W^o \in \mathbb{R}^{d_{modelo}*hd_v}$.

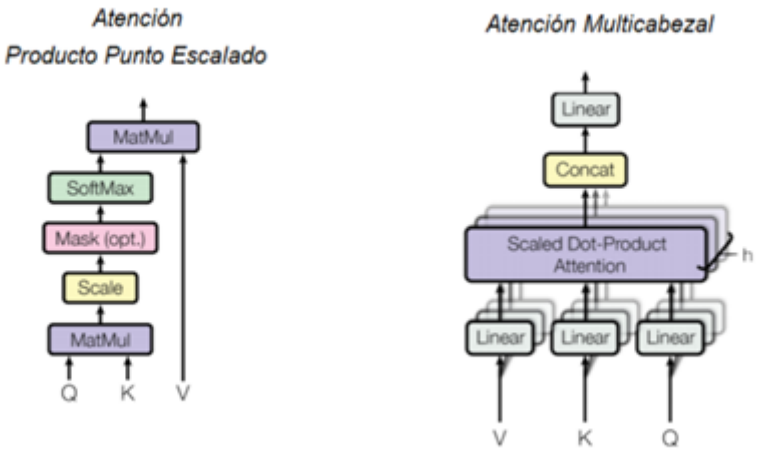

Figura 2.4: (izquierda) Mecanismo de atención con producto punto escalado. (Derecha) Mecanismo multicabezal de atención que consta de diversas capas de atención dispuestas en paralelo [36].

2.4.4. Transformer

La arquitectura neuronal *Transformer* reemplaza las *RNN* por redes neuronales basadas en la autoatención empleadas tanto en el codificador como en el decodificador. En este modelo, el codificador y el decodificador están compuestos por varias capas de autoatención enlazadas formando diferentes bloques.

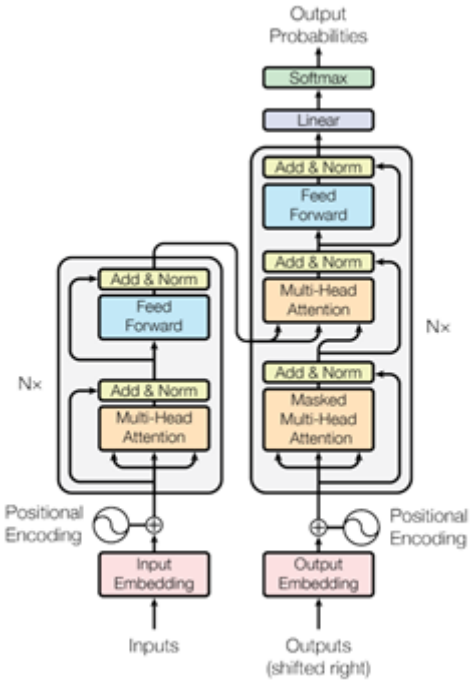

Figura 2.5: Arquitectura del modelo *Transformer* [36].

Por un lado, el codificador está constituido por la conexión entre 6 bloques iguales, donde cada bloque está compuesto por dos capas. En relación con la primera capa, hay que señalar que se corresponde con un mecanismo multicabezal de autoatención, y la segunda es una simple red completamente conectada *feed-forward*. Además, se emplea una conexión residual sobre cada una de las dos capas, seguida de una capa de normalización.

Por otro lado, el decodificador está formado por la unión de 6 bloques, en los que la estructura es similar a los bloques del codificador, pero se añade una capa multicabezal de autoatención en la que el valor y las claves pertenecen a la salida del codificador y la consulta a la salida de la primera capa atencional del decodificador. También se añaden conexiones residuales sobre las tres capas seguidas de una capa de normalización. Además, en la primera capa de atención se emplea enmascarado, que combinado que el hecho de que las salidas de los embeddings se compensan en una posición, asegura que las predicciones para la posición i pueda depender únicamente de las salidas conocidas en las posiciones menores de i.

Asimismo, cabe destacar que en el modelo *Transformer* existen tres lugares donde emplear la autoatención, por ello existen los vectores Q, K, V. En primer lugar, en los mecanismos de autoatención del codificador los vectores Q, K, V, tienen el mismo valor que se corresponde con la frase fuente. En segundo lugar, respecto a los mecanismos de autoatención del decodificador, los valores de los vectores se corresponden con la traducción objetivo. Por último, con respecto a los estados de los mecanismos de atención compartidos entre el codificador y el decodificador, el vector Q se corresponde con la traducción objetivo mientras que los vectores K y V se relacionan con la oración de entrada.

CAPÍTULO 3
Traducción automática neuronal sensible al contexto

Los sistemas de traducción automática neuronal representan el actual estado del arte para las tecnologías de traducción automática e incluso existen enfoques que afirman haber alcanzado el rendimiento de las personas en esta tarea [15]. En relación con los modelos convencionales de traducción automática neuronal, cabe destacar que realizan la traducción de un documento a nivel de oración, es decir considerando de forma aislada cada una de las frases existentes en dicho documento. Así pues, este hecho puede perjudicar la calidad de la traducción especialmente en términos de coherencia, cohesión y consistencia [37].

Con el objetivo de solucionar dicho problema, los investigadores han estudiado diferentes enfoques de los modelos a nivel de documento para la traducción automática neuronal, es decir teniendo en cuenta el contexto para una oración dada las oraciones que la rodean, y han observado resultados prometedores en términos de la puntuación BLEU, así como con la evaluación humana [22] mejorando la actuación de los modelos a nivel de oración.

En relación con la incorporación de los contextos en los modelos de traducción automática neuronal, hay que señalar que existen dos enfoques comunes: la vía más simple consiste en concatenar el contexto a la oración actual con el fin de crear una oración de entrada sensible al contexto [3, 34], mientras que el otro enfoque más utilizado emplea redes neuronales adicionales para codificar el contexto de las oraciones [37, 17, 23, 35].

En cuanto al último enfoque basado en alterar la arquitectura neuronal de los modelos de traducción automática, normalmente se encuentran opciones basadas en utilizar un único codificador o las que utilizan varios codificadores, no obstante, también existen aproximaciones que utilizan redes jerárquicas de atención [38] tanto en el codificador como en el decodificador [25]. Por otro lado, también aparecen propuestas en las que no se modifica la arquitectura neuronal y se introduce la información contextual fusionando la salida del decodificador con modelos de lenguaje [13].

A pesar de que tales enfoques han logrado mejoras significativas, existen tras inconvenientes principales que pueden limitar el desarrollo de la traducción automática neuronal a nivel de documento [24]. El primer problema, se trata de que los datos para entrenar modelos robustos a nivel de documento son relativamente escasos; el segundo consiste en que los experimentos en los diferentes proyectos son realizados con conjuntos de datos propios que han sido modificados en tamaño, dominio e idiomas empleados; por último, al existir diferentes modelos a nivel de documento implementados en diferentes arquitecturas, es difícil construir robustamente modelos y realizar una comparativa justa con los diferentes enfoques desarrollados en diferentes conjuntos de datos o arquitecturas.

3.1 Arquitectura multicodificador

En este apartado, se va a realizar la explicación del modelo sensible al contexto que se ha implementado en este proyecto, el cual se basa en modificar la arquitectura del modelo *Transformer* añadiendo un codificador adicional [37] encargado de tratar las oraciones que forman parte del contexto de la oración de entrada y dejando la parte del decodificador intacta. Así pues, en este modelo existen el codificador de entrada y el codificador del contexto.

En relación con el codificador de entrada, cabe destacar que está compuesto por un conjunto de N capas. Las primeras $N-1$ capas son idénticas y representa las capas originales del codificador del *Transformer*. Por otro lado, la última capa incorpora la información contextual como se puede observar en la Figura 3.1. Además del multicabezal de autoatención para el codificador de las frases de entrada, tiene un bloque que actúa como multicabezal de atención sobre la salida del bloque del codificador de las oraciones del contexto. Las salidas de los dos mecanismos de atención son combinados mediante una *gated sum*.

De manera más precisa, dada una oración de entrada x_i a ser traducida, se consideran sus K oraciones previas en el mismo documento como contexto de entrada $C = x_{i-K}, ..., x_{i-1}$. El codificador de entrada utiliza el multicabezal de autoatención, visto en la sección 2.4.3, para transformar una oración de entrada en una secuencia de representaciones $O^h = \{o_i^h, \ldots, o_I^h\}$ con:

$$o_i^h = cabezal_i = Autoatencion\left(QW_i^Q, KW_i^K, VW_i^V\right) \in \mathbb{R}^{d/H} \tag{3.1}$$

en la que h es uno de los H cabezales, Q, K, V, respectivamente representan los vectores de consulta, claves y valores, además $W_i^Q, W_i^K, W_i^V \in \mathbb{R}^{d*d}$ se corresponden con los parámetros entrenables y d indica el tamaño oculto. El codificador del contexto emplea las mismas redes que el codificador de la entrada para obtener la salida del contexto $\hat{O}$. Finalmente, las dos salidas de los codificadores O y $\hat{O}$ son combinadas mediante una gated sum, como:

$$\lambda_d = \sigma(W_\lambda \left[O_d, \hat{O}_d\right] + b_d) \tag{3.2}$$

$$\mathbf{O}' = \lambda_d \odot O_d + (1 - \lambda_d) \odot \hat{O}_d \tag{3.3}$$

donde es la función logística sigmoide y W_λ los parámetros del modelo. $\mathbf{O}'$ es la representación final a nivel de documento la cual se acaba utilizando en la segunda capa del decodificador.

Con respecto a la relación de los vectores Q, K, V con las oración de entrada del codificador contextual, hay que señalar que tienen el mismo valor que la frase de entrada (que se corresponde con el contexto con la oración fuente del codificador de entrada). En segundo lugar, respecto a los mecanismo de atención del codificador de entrada, los valores de los vectores se corresponden con la frase fuente. En tercer lugar, en relación con los mecanismos de atención del decodificador, los valores de los vectores se corresponden con la traducción objetivo. En cuarto lugar, respecto los mecanismos de atención compartidos entre ambos codificadores, el vector Q se corresponde con las frases de entrada, mientras que los vectores K y V están asociados a las oraciones del contexto de la frase de entrada. Por último, con respecto al mecanismo de atención compartido entre los codificadores y el decodificador, el vector Q se corresponde con la traducción objetivo

mientras que los vectores K y V están asociados a la salida del mecanismo de atención que combina las oraciones de entrada y las del contexto.

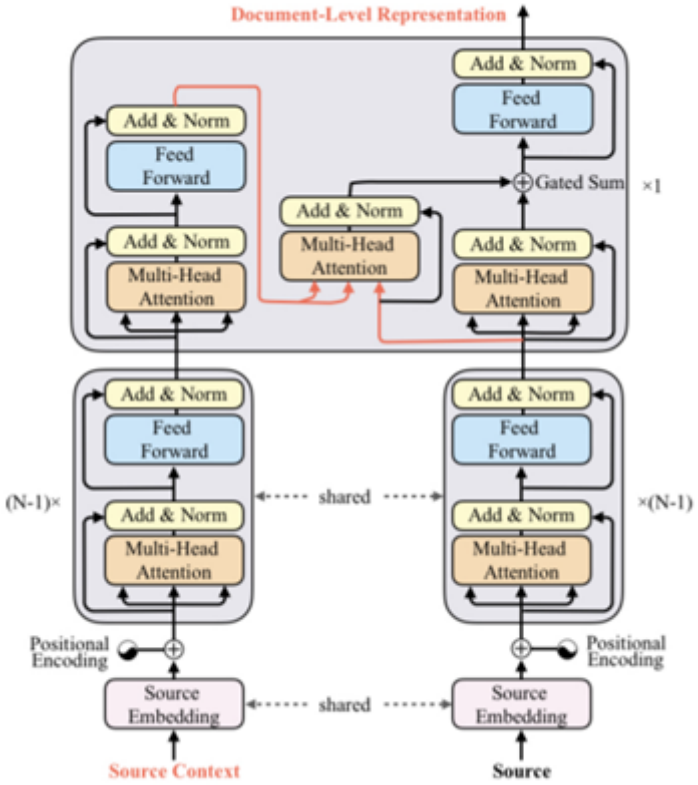

Figura 3.1: Arquitectura del modelo multicodificador sensible al contexto [24].

Con respecto al codificador del contexto, también está compuesto por un conjunto de N capas. Las primeras $N-1$ capas son idénticas y representa las capas originales del codificador del *Transformer*. En este caso, se utiliza la estrategia basada en compartir los parámetros de las primeras $N-1$ capas con el codificador de la entrada debido a que otorga mejores resultados [37] que empleando diferentes configuraciones para cada codificador. No obstante, aunque la estrategia de compartir pesos funciona de forma adecuada y ayuda a mejorar los resultados, existe otro tipo de estrategia para este modelo que consiste en entrenarlo en dos etapas [39].

Así pues, la primera etapa de esta estrategia consiste en realizar un primer entrenamiento del modelo utilizando únicamente las oraciones de entrada sin tener en cuenta el contexto, es decir, entrenar el modelo a nivel de oración (los módulos del contexto en la Figura 3.2 resaltados en rojo (b) están desactivados en esta primera etapa). Una vez realizada la primera etapa, la siguiente trata de congelar los pesos del codificador de las oraciones de entrada y del decodificador con el objetivo de no sobreajustar el modelo y, realizar el entrenamiento a nivel de documento añadiendo el codificador del contexto. Por último, cabe destacar que, empleando esta estrategia de dos etapas, dependiendo del tamaño del conjunto de datos, el empleo de una sola capa para el codificador del contexto puede otorgar resultados similares que con 6 capas [23].

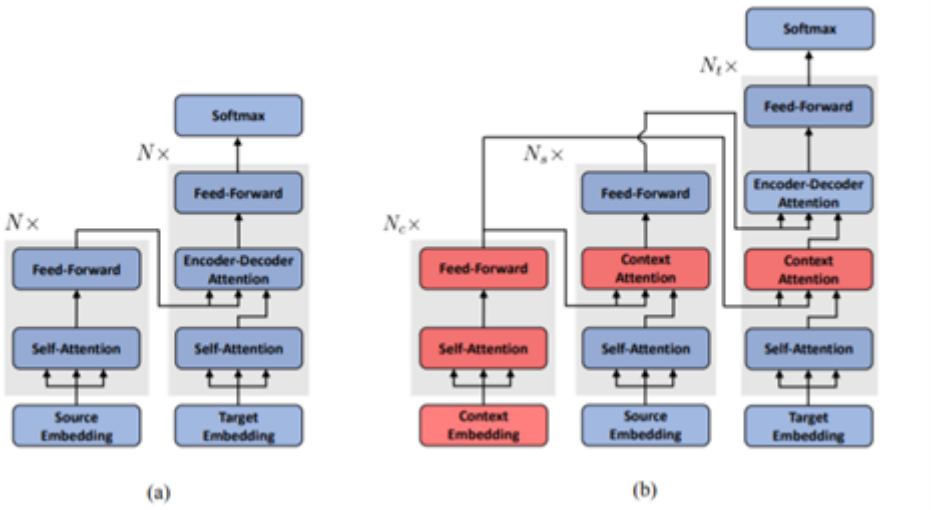

Figura 3.2: (a) El modelo de traducción original *Transformer* y (b) el *Transformer* extendido que utiliza el contexto a nivel de documento [39]

CAPÍTULO 4
Marco experimental

En este capítulo se expone el entorno y las librerías que se han empleado, se propone el problema a resolver y los diferentes procesos que se han desarrollado con el objetivo de implementar una solución.

4.1 Tecnologías utilizadas

El lenguaje de programación que se ha empleado en el avance de la fase de experimentación de este proyecto es Python. Este es un lenguaje de tipado dinámico cuyo fin primordial es alcanzar un código legible y sencillo de entender, además es multiparadigma y multiplataforma.

De igual forma, Python tiene una gran variedad de librerías para trabajar en el ámbito del procesamiento del lenguaje natural y en el del aprendizaje automático. Seguidamente, se presentan las herramientas principales que se han usado en este proyecto.

Keras

Keras [11] es una librería que se ha empleado en este trabajo junto a *Tensorflow*. Esta herramienta simplifica la construcción de sistemas de redes neuronales y además proporciona utilidades para realizar el entrenamiento y la evaluación de estos.

Moses

Moses [21] es una librería de código abierto que implementa técnicas de traducción automática estadística además de aportar diferentes herramientas como es el tokenizador y limpiador empleadas en los conjuntos de datos.

Numpy

Numpy [14] es una librería que proporciona una gran cantidad de funcionalidades para operar con matrices y vectores.

Tensorflow

Tensorflow [1] es una plataforma de código abierto de extremo a extremo para el aprendizaje automático. Cuenta con un ecosistema integral y flexible de herramientas, bibliotecas y recursos de la comunidad que les permite a los investigadores innovar con

el aprendizaje automático y, a los desarrolladores, compilar e implementar con facilidad aplicaciones con tecnología de aprendizaje automático.

4.2 Evaluación

Después del proceso de entrenamiento se tiene que realizar una evaluación del modelo obtenido con el fin de valorar la calidad de las traducciones que proporciona. En este trabajo, se emplea la métrica BLEU [27] para realizar la evaluación automática de las traducciones producidas por los sistemas. Esta métrica calcula la media geométrica de una precisión modificada por n-gramas, p_n, multiplicado por el factor BP que penaliza las oraciones cortas. Así pues, BLEU puede modelizarse de la siguiente manera:

$$BLEU = BP * exp\left(\sum_{n=1}^{N} \frac{log\ p_n}{N}\right) \tag{4.1}$$

La definición más común de BLEU se calcula sobre la concatenación de todas las frases de test, y normalmente emplea n-gramas de orden 4. Por otro lado, el resultado final de la métrica es un valor entre 0 y 1, a mayor valor mejor es la traducción. Dicho valor suele multiplicarse por 100 para obtener una mayor capacidad de interpretación.

4.3 Corpus

En esta sección, se introducen los corpus que se han utilizado en el marco experimental. Dichos conjuntos de datos están estructurados a nivel de documento. Además, los corpus se han *tokenizado* y limpiado mediante la herramienta Moses. Por último, cabe destacar que se han empleado cuatro conjuntos de datos en este proyecto.

4.3.1. TED Talks

TED Talks ofrece un gran abanico de conjuntos de datos procedentes del apartado de traducción automática de IWSLT [9]. Los corpus de TED Talks son difíciles de traducir debido a la gran variedad de temas que se encuentran en conjuntos de entrenamiento pequeños. Este conjunto de datos contiene un total de 100 pares de lenguas. En este trabajo, se han empleado el par de lenguas español-inglés y español-alemán.

TED Talks español-inglés

Este corpus forma parte de la campaña de evaluación IWSLT de 2014, del cual se emplea el conjunto dev2010 para validación, y los conjuntos test2010-2012 para la evaluación del modelo de traducción.

Tabla 4.1: Estadísticas del corpus TED-14 para el par de idiomas es-in. M representa millones y K representa miles.

Conjunto	Español - Inglés	
Entrenamiento	Oraciones	0.2M
	Documentos	1.5K
	Palabras	6.5M
Validación	Oraciones	0.8K
	Documentos	8
Test	Oraciones	4.7K
	Documentos	42

TED Talks español-alemán

Este conjunto de datos forma parte de la campaña de evaluación IWSLT de 2020, del cual se emplea el conjunto dev2010 para validación, y los conjuntos test2010-2018 para la evaluación del modelo de traducción.

Tabla 4.2: Estadísticas del corpus TED-20 para el par de idiomas es-al. M representa millones y K representa miles.

Conjunto	Español - Alemán	
Entrenamiento	Oraciones	0.3M
	Documentos	2.5K
	Palabras	10M
Validación	Oraciones	1K
	Documentos	8
Test	Oraciones	13K
	Documentos	123

4.3.2. News Commentary

El corpus News Commentary [33] fue creado con el fin de ser utilizados como datos de entrenamiento para la Conferencia para la Campaña de Evaluación de Traducción Automática Estadística. Además, este conjunto de datos está construido principalmente a partir de noticias con una temática económica o política. En relación con el número de pares de lenguas que presenta, hay que señalar que contiene 12 pares de lenguas, no obstante, en este proyecto se utilizan el corpus con el par de lenguas español-inglés y español-alemán.

News Commentary español-inglés

Para este corpus y el par de idiomas español-inglés, se emplea el conjunto de entrenamiento de la edición 2016, el conjunto newtest2008 [6] para validación, y los conjuntos newtest2009-2013 [8, 7] para la evaluación del modelo de traducción.

Tabla 4.3: Estadísticas del corpus News-Commentary-16 para el par de idiomas es-in. M representa millones y K representa miles.

Conjunto	Español - Inglés	
Entrenamiento	Oraciones	48K
	Documentos	1K
	Palabras	2.3M
Validación	Oraciones	2K
	Documentos	1
Test	Oraciones	14K
	Documentos	140

News Commentary español-alemán

Con respecto a este conjunto de datos, para el par de lenguas español-alemán se emplea el conjunto de entrenamiento de la edición 2016, el conjunto dev2008 para validación, y los conjuntos test2009-2013 para la evaluación del modelo de traducción.

Tabla 4.4: Estadísticas del corpus News-Commentary-16 para el par de idiomas es-al. M representa millones y K representa miles.

Conjunto	Español - Alemán	
Entrenamiento	Oraciones	0.3M
	Documentos	7K
	Palabras	12M
Validación	Oraciones	2K
	Documentos	1
Test	Oraciones	14K
	Documentos	140

4.4 Implementación

En este apartado, se exponen los detalles de la implementación que se ha desarrollado en este proyecto del sistema de traducción automática neuronal sensible al contexto construido en Keras junto a Tensorflow.

4.4.1. Preparación de los datos

En relación con los conjuntos de datos seleccionados para el desarrollo de la experimentación, hay que señalar que se ha creado un script específico para cada uno de ellos debido a los diferentes formatos en los que se encuentran. Así pues, el objetivo del *script* consiste en crear tres ficheros de texto para el entrenamiento, validación y test, es decir 9 en total. El primer fichero de texto se corresponde con cada una de las frases del lenguaje de entrada, el segundo con cada una de las oraciones del lenguaje de salida y, por último, un fichero asociado a las frases que pertenecen al contexto de cada una de las frases del lenguaje de entrada.

Con respecto al fichero que contiene las frases del contexto, cabe destacar que en este trabajo únicamente se tiene en cuenta la frase anterior a la frase correspondiente del lenguaje de entrada. Además, como los corpus están compuestos a nivel de documento, cada vez que se inicia un nuevo documento, la primera oración del idioma de entrada de

ese documento tendrá asociada en el contexto el token «INIC». En la siguiente ilustración se muestra un ejemplo de este preproceso.

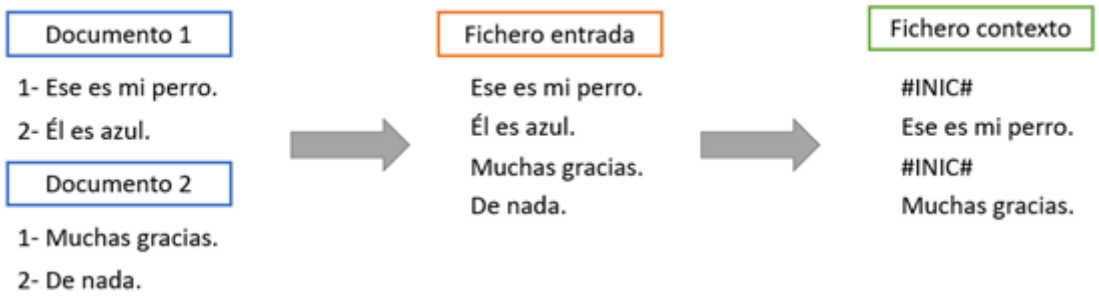

Figura 4.1: Ejemplo de la composición de los conjuntos de datos.

Por lo tanto, ya procesados los corpus, se utiliza la herramienta Moses para *tokenizarlos* y limpiarlos. El siguiente paso consiste en convertir cada una de las frases de los diferentes conjuntos de datos en su correspondiente representación vectorial empleando números enteros. Para ello, se ha utilizado la técnica basada en subpalabras *SubwordTextEncoder* que proporciona la herramienta Tensorflow, ya que además de codificar las oraciones, permite reducir el vocabulario empleando subpalabras y por tanto permitir ahorrar un mayor espacio tanto a niveles de GPU como de RAM en los modelos neuronales.

Por consiguiente, al igual que la técnica basada en la construcción de palabras BPE [31], son entrenadas de una forma no supervisada confiando en la distribución de secuencias de caracteres, pero sin tener en cuenta las propiedades morfológicas de los idiomas en cuestión. Además, cuando se entrenan de forma conjunta con pares de idiomas, son capaces de identificar y beneficiarse de las palabras que comparten su raíz en idiomas diferentes.

Por último, una vez ya codificadas las oraciones de los corpus en sus representaciones numéricas vectoriales, se crea, para el conjunto de entrenamiento, validación y test, un objeto *Dataset* de Tensorflow compuesto por el corpus codificado de las oraciones con el idioma de entrada, el de las oraciones con el contexto de las oraciones de entrada, y las oraciones con el idioma objetivo que serán utilizados para el entrenamiento, validación y evaluación del modelo de traducción respectivamente.

4.4.2. Construcción del modelo

En cuanto a la implementación de la arquitectura multicodificador explicada en la sección 3.1, hay que resaltar que en primer lugar se ha construido como base el modelo neuronal *Transformer* y, a partir de este, se modifica su arquitectura añadiendo un codificador extra que sea sensible al contexto. Para ello, cabe destacar que se ha empleado la implementación del *Transformer* original proporcionada por Tensorflow [1], y a partir de esta, en este proyecto se ha modificado el código para añadir el contexto en la entrada del modelo y, además, se ha implementado y desarrollado la estructura del codificador contextual[2].

[1] Modelo *Transformer*. https://www.tensorflow.org/text/tutorials/transformer?hl=en

[2] Modelo *Transformer* multicodificador sensible al contexto. https://colab.research.google.com/drive/1bb44H0qzG4mA0smc4_hWt2tlVgH7_dkd

Transformer

Por lo tanto, el Transformer original está compuesto de un codificador, llamado como *Encoder*, por un decodificador, *Decoder*, y por una capa densa final con el tamaño del vocabulario del lenguaje de salida.

A su vez, el codificador obtiene los *embeddings* y la codificación posicional de las oraciones de entrada. La salida de la codificación posicional se une a la primera de las N capas codificadoras, *EncoderLayer*. Como se expone en el apartado 2.4.4 cada capa codificadora está compuesta por un mecanismo multicabezal de atención y una red completamente conectada *feed-forward* con conexiones residuales y salidas normalizadas.

Por otro lado, el decodificador extrae los *embedding* y la codificación posicional de las oraciones objetivo. La salida de la codificación posicional se conecta con la primera de las N capas decodificadoras, *DecoderLayer*. Cada capa decodificadora está formada por un primer mecanismo multicabezal de autoatención donde el valor las claves y las consultas están compuestas por la misma entrada, luego por un segundo mecanismo multicabezal de autoatención en la que el valor y las claves pertenecen a la salida del codificador y la consulta a la salida de la primera capa atencional del decodificador, y finalmente por una última subcapa que comprende una red completamente conectada *feed-forward*. De igual manera que en las capas codificadoras, todas las subcapas decodificadoras están conectadas con conexiones residuales y sus salidas son normalizadas.

Además, cabe destacar que previamente a realizar la normalización de las capas tanto el codificador como en el decodificador se les aplica una capa con *dropout*. Por consiguiente, el *dropout* consiste en desactivar un número de neuronas de una capa de forma aleatoria en cada iteración en el entrenamiento de la red neuronal, obligando así a que las neuronas que son cercanas no aprendan patrones que puedan dar lugar al sobreajuste del modelo.

Transformer sensible al contexto

A partir de la construcción de la arquitectura del *Transformer* original, se desarrolla la creación de la arquitectura multicodificador del *Transformer* sensible al contexto añadiendo un codificador adicional y reduciendo el número de capas codificadoras de N a $N-1$.

Así pues, el codificador del contexto tiene la misma estructura que el codificador normal, únicamente varía que la entrada de este codificador se corresponde con las oraciones correspondientes al contexto de la oración de entrada. Igualmente, las capas del codificador del contexto también tienen la misma estructura que las del codificador original. Por lo tanto, el codificador original y el de contexto están compuestos por $N-1$ capas, donde la última capa se corresponde con la capa que combina ambas salidas.

Así pues, en este modelo *Transformer* basado en el contexto, la salida del codificador de las oraciones de entrada se combina con la salida del codificador de las oraciones del contexto. Para ello, se construye un último codificador, *EncoderContext*, compuesto por una sola capa, *EncoderLayerContext*. Dicha capa, está compuesta por dos mecanismos multicabezal de atención, el primero en el que el valor y las claves pertenecen a la salida del codificador de contexto y la consulta a la salida del codificador original, y el segundo en el que el valor, las claves y la consulta son de la salida del codificador original.

La salida de estos dos mecanismos de atención es concatenada, y, a esa concatenación, se le aplica una función de salida sigmoide a partir de una capa densa. Por tanto, se construye una *gated sum* en la que la salida de la función sigmoide regula el peso que

tiene cada uno de los multicabezales de autoatención. Además, la salida de la *gated sum* se conecta a una red completamente conectada *feed-forward*.

4.4.3. Entrenamiento del modelo

Una vez ya preparado el conjunto de datos y creados los modelos de traducción automática, el paso a seguir es entrenar tales modelos. Para ello, es necesaria la especificación de los parámetros asociados a los sistemas de traducción. Entre los diferentes parámetros a establecer en el entrenamiento, se encuentra el tamaño del *batch* que indica la frecuencia con la que el gradiente se actualiza; por último, el número de épocas para señalar la cantidad de veces que se entrena el modelo sobre los datos proporcionados.

Con respecto a los aspectos de los sistemas neuronales, hay que indicar el tamaño del modelo *Transformer*, el número de cabezales a emplear en los mecanismos de autoatención, el número de capas codificadoras y decodificadoras, la dimensionalidad de las capas internas, la ratio de *dropout*, tamaño del vocabulario de entrada y salida y la máxima longitud para las oraciones. Por otro lado, también se debe especificar el optimizador a elegir, junto a la ratio de aprendizaje y el planificador que la calcule además de la métrica y la pérdida a emplear. Otro aspecto que se debe seleccionar es cómo realizar el entrenamiento del modelo sensible al contexto, si compartiendo los parámetros de los dos codificadores, o realizándolo en dos pasos.

4.5 Experimentación

Con el objetivo de corroborar que la adición de la información contextual a las oraciones de entrada permite que el modelo de traducción proporcione traducciones con una mayor calidad que los sistemas de traducción a nivel de oración, se va a realizar una experimentación con los diferentes corpus presentados anteriormente.

Así pues, se establece como *baseline* el modelo *Transformer*, a nivel de sentencia, compuesto por tres capas tanto en el codificador como en el decodificador, con un tamaño de 512, una dimensionalidad de 2048 unidades en las capas internas, de 8 cabezales de autoatención, un tamaño de las sentencias de entrada y salida configurada en 70 y con *dropout*. Con respecto al número de capas empleados, hay que señalar que debido a que el tamaño de los conjuntos de datos utilizados no es muy grande el *Transformer* con 3 capas en el codificador y en el decodificador, presenta resultados similares que empleando un mayor número de capas.

Para el entrenamiento, se utiliza un tamaño de *batch* de 200, y un total de 30 épocas. Además, se emplea el optimizador Adam [18] con $\beta 1 = 0{,}9$, $\beta 2 = 0{,}98$, $\epsilon = 10^{-9}$ y se varía la ratio de aprendizaje.

Por otro lado, para los modelos sensibles al contexto se utiliza la misma composición de las capas, aunque dependiendo del tipo de entrenamiento que se realice, ya sea compartiendo los pesos entre los codificadores o entrenamiento en dos etapas, se varía el número de capas del codificador contextual. Además, hay que señalar que debido a las limitaciones de hardware se emplea un tamaño de *batch* de 64 unidades para las arquitecturas pendientes al contexto.

Por último, cabe destacar que la experimentación desarrollada en este trabajo se ha basado principalmente en comprobar qué técnica de entrenamiento con la arquitectura multicodificador presenta los mejores resultados, aplicando diferentes valores de *dropout*.

4.6 Hardware

En este proyecto se han entrenado los diferentes modelos en una máquina con una GeForce RTX 2080. Dependiendo del corpus para el que se entrene los modelos la duración del entrenamiento varía.

En relación con al corpus TED-14 con los pares de idiomas español-inglés, cada época de entrenamiento tarda alrededor de 385 segundos, comprendiendo un entrenamiento total de 3 horas. Por otro lado, para el conjunto de datos TED-20 con los pares de lenguas español-alemán, cada época de entrenamiento dura unos 553 segundos, por lo que el entrenamiento completo transcurre en 4.6 horas. Con respecto a News Commentary español-inglés, una época dura unos 89 segundos y el entrenamiento completo unos 44 minutos. Finalmente, con News Commentary español-alemán las épocas tardan 574 segundos, comprendiendo un entrenamiento final de 4.78 horas.

CAPÍTULO 5
Resultados experimentales

En este apartado se exponen los resultados obtenidos a partir de la experimentación realizada para cada uno de los conjuntos de datos recopilados, asimismo se extraen las conclusiones relacionadas con estos. Además, hay que señalar que en las tablas de resultados se han representado como TS la estrategia de entrenamiento en dos etapas, y como WS la estrategia de entrenamiento compartiendo pesos entre los codificadores.

TED Talks español-inglés

En primer lugar, con el fin de comparar que estrategia de entrenamiento es más eficaz en el corpus TED-14 español-inglés, para el modelo multicodifcador sensible al contexto, por un lado, se ha probado compartiendo completamente los pesos entre los codificadores, y por el otro mediante el entrenamiento en dos etapas. También, se expone los resultados obtenidos con diferentes ratios en el parámetro de *dropout*.

Tabla 5.1: Comparación del sistema sensible al contexto con las dos estrategias de entrenamiento en el corpus Ted-14 es-en.

Sistema		Capas	BLEU	
			Dropout = 0.1	Dropout = 0.3
Contexto	WS	3	29.97	32.89
	TS	3	31.05	34.80
	TS	1	31.04	34.83

A partir de los resultados de la Tabla 5.1 se observa que, para este conjunto de datos, el modelo basado en el contexto que comparte los pesos entre el codificador de las oraciones de entradas y el codificador del contexto en el entrenamiento otorga los peores resultados en la métrica BLEU . Con respecto al *dropout*, hay que resaltar que todos los sistemas tienen un mejor rendimiento con el valor de 0.3.

Por otro lado, el modelo sensible al contexto entrenado con la estrategia de dos etapas consigue mejorar el rendimiento del modelo basado en compartir los pesos. Adicionalmente, se contempla como con esta estrategia la diferencia entre el modelo empleando 1 sola capa en el codificador contextual presenta un rendimiento similar a emplear 3 capas, este hecho puede deberse principalmente al tamaño del conjunto de datos.

TED Talks español-alemán

En relación con el conjunto de datos TED-20 español-alemán, también se ha desarrollado la experimentación en base al tipo de estrategia de entrenamiento y al valor del *dropout* aplicado. Así pues, se han obtenido los siguientes resultados.

Tabla 5.2: Comparación del sistema sensible al contexto con las dos estrategias de entrenamiento en el corpus Ted-20 es-de.

Sistema		Capas	BLEU	
			Dropout = 0.1	Dropout = 0.3
Contexto	WS	3	16.11	18.72
	TS	3	16.27	18.88
	TS	1	16.23	18.86

A la vista de los resultados de la Tabla 5.2 se observa que con este corpus el uso de modelos sensibles al contexto utilizando la compartición de pesos en el entrenamiento no consigue mejorar las prestaciones del modelo con la estrategia del entrenamiento en dos etapas en la medida BLEU. De igual manera, se muestra una diferencia notable entre el empleo de un valor de *dropout* u otro, siendo 0.3 el que produce los mejores resultados. Asimismo, con la estrategia de dos etapas, la diferencia entre el empleo de una capa y el de tres es prácticamente inexistente.

News Commentary español-inglés

Acerca del corpus News Commentary-16 español-inglés, hay que señalar que a partir de la experimentación realizada a partir de la estrategia del entrenamiento y *dropout* aplicado se han alcanzado los resultados de la próxima tabla.

Tabla 5.3: Comparación del sistema sensible al contexto con las dos estrategias de entrenamiento en el corpus NC-16 es-en.

Sistema		Capas	BLEU	
			Dropout = 0.1	Dropout = 0.3
Contexto	WS	3	9.48	12.17
	TS	3	9.51	12.27
	TS	1	9.56	12.29

Como se puede contemplar a través de la Tabla 5.3 de resultados, el entrenamiento de los modelos multicodificadores mediante la estrategia de dos etapas proporciona los mejores resultados en este conjunto de datos, además resaltar que el empleo de una capa en el codificador contextual obtiene un mayor BLEU que con tres capas, aunque la diferencia es muy pequeña. Igualmente, para todos los sistemas la aplicación de un *dropout* de 0.3 permite alcanzar mejores resultados que con un valor de 0.1.

News Commentary español-alemán

Con respecto al conjunto de datos News Commentary-16 español-alemán, en la siguiente tabla se muestran los resultados obtenidos a partir de la experimentación con las diferentes estrategias de entrenamiento y el valor del *dropout* aplicado.

A partir de los resultados de la Tabla 5.4 se aprecia que el mejor modelo contextual se obtiene empleando la estrategia de entrenamiento en dos etapas con una sola capa en el

Tabla 5.4: Comparación del sistema sensible al contexto con las dos estrategias de entrenamiento en el corpus NC-16 es-de.

Sistema		Capas	BLEU	
			Dropout = 0.1	Dropout = 0.3
Contexto	WS	3	9.87	11.28
	TS	3	9,95	11.32
	TS	1	9.90	11.33

codificador, no obstante, la diferencia entre el valor en la medida BLEU es ínfima. Además, hay que destacar que existe una gran diferencia dependiendo del *dropout* empleado en los modelos, siendo 0.3 el mejor valor.

Resultados globales

Una vez expuestos los resultados obtenidos para cada uno de los conjuntos de datos recopilados en la experimentación, en la siguiente tabla se muestra la comparativa entre el mejor sistema sensible al contexto y el modelo base a nivel de secuencia para cada uno de los conjuntos.

Tabla 5.5: Comparación entre el mejor modelo sensible al contexto y el baseline de cada corpus.

Sistema	TED Talks		News Commentary	
	Es-En	Es-De	Es-En	Es-De
Baseline	34.29	18.60	11.79	11.25
Contexto	34.83	18.88	12.29	11.33

Como se observa en la Tabla 5.5, la construcción de modelos que aportan información contextual a las oraciones de entrada consigue mejorar los resultados en la métrica BLEU que los modelos tradicionales que solo tienen en cuenta la oración. Además, cabe destacar que para todos los conjuntos de datos la estrategia de entrenamiento basada en dos etapas ha otorgado mejores resultados que la estrategia de compartir los pesos entre los codificadores.

Por otro lado, hay que señalar que el valor del *dropout* tiene una gran influencia en las prestaciones de los modelos de traducción, ya que el empleo de un valor de 0.3 mejoraba generalmente en todos los conjuntos de datos al menos 2 puntos en la medida BLEU que empleando un valor de 0.1.

En relación con el par de idiomas español-inglés, se observa que dependiendo del conjunto de datos se obtiene un BLEU diferente. Esto puede deberse a que el conjunto de datos TED-14 para este par de lenguas, contiene una cantidad de 200 mil oraciones para el entrenamiento, y alrededor de 5 mil para la evaluación, mientras que en el corpus News Commentary-16 solo se emplean 48 mil frases para entrenamiento, pero 14 mil para evaluación. Así pues, el modelo de TED-14 puede que otorgue un BLEU más alto ya que tiene un mayor número de datos con el que aprender las relaciones entre idiomas y un menor número de evaluaciones por lo que los errores en las traducciones pueden ser menos perceptible que con el modelo de News Commentary-16.

Con respecto al par de lenguas español-alemán, de igual manera que con el par anterior, que dependiendo del corpus se obtienen BLEU diferentes. No obstante, ambos conjuntos de datos disponen prácticamente de la misma cantidad de datos para entrenamiento y test. Así pues, esta diferencia entre rendimientos puede deberse a que el modelo

de TED-20 en el entrenamiento contempla y aprende traducciones que aparecen más frecuentemente en el conjunto de evaluación que el modelo de News Commentary-16.

También se observa que los modelos para los pares de idiomas español-inglés proporcionan traducciones con mayor calidad que con los pares de lenguas español-alemán. Este hecho se podría deber a que los modelos tienen una mayor capacidad de comprender y establecer las relaciones existentes entre las traducciones del español al inglés que las del español al alemán.

Análisis de los resultado

Con el objetivo de analizar las diferencias entre las traducciones del modelo base con el modelo contextual, se han extraído diferentes ejemplos de traducciones realizadas por ambos modelos entrenados a partir del corpus TED-14 con el par de lenguas español-inglés.

Para la oración de entrada en castellano: «Les animo a que lean el libro de Steve o vean la película para entender no sólo el bello vínculo que se creó entre estos dos hombres sino cómo la música ayudó a moldear ese vínculo, y finalmente jugó un papel decisivo para ayudar a Nathaniel a salir de las calles.», el modelo base traduce la frase como:

«I encourage you to read Steven's book or see the movie to understand not only the beautiful link that was created between these two men, but how music helped shape that link, and eventually he played a decision-making role to help Nathaniel stand out of the street.»

Y el modelo contextual:

«I encourage you to read Steven's book or see the film to understand not only the beautiful link that was created between these two men, but how music helped shape that link, and finally played a critical role to help Nathaniel go out street.»

Como se puede contemplar, las traducciones de los dos modelos son bastante similares, no obstante, en la última parte de la oración el modelo contextual realiza una traducción más cercana al significado real.

Con respecto a la frase de entrada en castellano, «Él acababa de escuchar una interpretación de la primera y cuarta sinfonía de Beethoven y vino detrás del escenario a presentarse.», el modelo base traduce la oración tal que:

«He just heard a performance of the first and fourth symphony of Beethoven and came to the stage to come.»

Mientras que el modelo contextual:

«He just heard a performance of the first and fourth symphony of Beethoven and came behind the stage to present.»

En este ejemplo, se observa como el modelo contextual realiza una mejor traducción de la parte final de la oración, ya que la traducción correspondiente del modelo base no tiene sentido respecto a la oración de entrada. Finalmente, la secuencia de entrada, «Y hablamos sobre música. Y unos días después recibí un correo electrónico de Steve diciendo que Nathaniel estaba interesado en una lección de violín conmigo.», es traducida por el modelo base como:

«And we talked about music. And a few days later, I got a email from Steve saying that Nathaniel was interested in a fiddle with me.»

Y por el modelo contextual:

«And we talked about music. And a few days later, I got a email from Steve saying that Nathaniel was interested in a lesson of violin with me.»

Para este último ejemplo, el traductor contextual también proporciona una traducción más acertada y cercana a la oración original que el modelo base.

Por último, hay que destacar que el modelo sensible al contexto ayuda a perfeccionar la calidad de las traducciones del modelo base, por lo que las mejoras en la métrica BLEU se pueden percibir en el momento de comparar las traducciones entre ambos modelos.

CAPÍTULO 6
Conclusiones

6.1 Conclusiones

En este proyecto, se ha desarrollado la implementación de un sistema de traducción neuronal automática sensible al contexto con el objetivo de incrementar la calidad de las traducciones de los modelos neuronales tradicionales.

Esta implementación ha sido probada empleando diferentes arquitecturas y estrategias en el entrenamiento de los sistemas en distintos conjuntos de datos, obteniendo resultados positivos. Estos resultados, exponen como en los diferentes corpus los modelos contextuales mejoran la calidad de las traducciones realizadas por los sistemas corrientes.

Por otro lado, se ha comprobado que para la arquitectura multicodificador implementada del modelo *Transformer* pendiente al contexto la estrategia de entrenamiento basado en dos etapas proporciona mejores resultados que con la estrategia de compartir los pesos entre los diferentes codificadores. Además, se ha observado la importancia que tiene el valor empleado en la ratio de *dropout* sobre los resultados obtenidos, ya que dependiendo de dicho valor la calidad de las traducciones pueden verse empeoradas en al menos dos puntos de BLEU.

En conclusión, cabe destacar que los resultados muestran que la vía de añadir información del contexto a las oraciones de entrada en los modelos de traducción automática neuronal mediante una arquitectura multicodificador ayudan a alcanzar mejores traducciones, no obstante, todavía queda mucho trabajo que realizar en este ámbito.

6.2 Trabajos futuros

Una futura línea de investigación consistiría en emplear un sistema sensible al contexto incorporando en la arquitectura original del modelo *Transformer* con redes jerárquicas de atención tanto en el codificador como en el decodificador y comprobar si la adición de estos componentes proporciona mejores resultados que con la arquitectura multicodificador que se ha implementado en este proyecto.

Un aspecto que considerar en la experimentación de este trabajo es que únicamente se ha tenido en cuenta como contexto la frase anterior a la oración en cuestión. En un futuro proyecto, sería interesante experimentar con diferentes números de frases previas en el contexto, o incluso emplear tanto frases anteriores como posteriores y observar la influencia que ejercen sobre la calidad de las traducciones finales.

Por último, otro posible trabajo futuro sería añadir la funcionalidad de poder utilizar información contextual en la herramienta NMT-Keras [28], además de añadir en el

catálogo de modelos el *Transformer* multicodificador pendiente al contexto que ha sido implementado en este proyecto.

Bibliografía

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] Sivanand Achanta, Rambabu Banoth, Ayushi Pandey, Anandaswarup Vadapalli, and Suryakanth V Gangashetty. Contextual representation using recurrent neural network hidden state for statistical parametric speech synthesis. In *9th ISCA Speech Synthesis Workshop*, pages 172–177, 2016.

[3] Ruchit Rajeshkumar Agrawal, Marco Turchi, and Matteo Negri. Contextual handling in neural machine translation: Look behind, ahead and on both sides. In *21st Annual Conference of the European Association for Machine Translation*, pages 11–20, 2018.

[4] Peter F Brown, John Cocke, Stephen A Della Pietra, Vincent J Della Pietra, Frederick Jelinek, John Lafferty, Robert L Mercer, and Paul S Roossin. A statistical approach to machine translation. *Computational linguistics*, 16(2):79–85, 1990.

[5] Peter F Brown, Stephen A Della Pietra, Vincent J Della Pietra, and Robert L Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, 19(2):263–311, 1993.

[6] Chris Callison-Burch, Cameron Shaw Fordyce, Philipp Koehn, Christof Monz, and Josh Schroeder. Further meta-evaluation of machine translation. In *Proceedings of the third workshop on statistical machine translation*, pages 70–106, 2008.

[7] Chris Callison-Burch, Philipp Koehn, Christof Monz, Kay Peterson, Mark Przybocki, and Omar Zaidan. Findings of the 2010 joint workshop on statistical machine translation and metrics for machine translation. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 17–53, Uppsala, Sweden, July 2010.

[8] Chris Callison-Burch, Philipp Koehn, Christof Monz, and Josh Schroeder. Findings of the 2009 Workshop on Statistical Machine Translation. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 1–28, Athens, Greece, March 2009.

[9] Mauro Cettolo, Christian Girardi, and Marcello Federico. Wit3: Web inventory of transcribed and translated talks. In *Conference of european association for machine translation*, pages 261–268, 2012.

[10] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[11] François Chollet et al. Keras. https://keras.io, 2015.

[12] Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

[13] Eva Martínez Garcia, Carles Creus, and Cristina España-Bonet. Context-aware neural machine translation decoding. In *Proceedings of the Fourth Workshop on Discourse in Machine Translation (DiscoMT 2019)*, pages 13–23, 2019.

[14] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[15] Hany Hassan, Anthony Aue, Chang Chen, Vishal Chowdhary, Jonathan Clark, Christian Federmann, Xuedong Huang, Marcin Junczys-Dowmunt, William Lewis, Mu Li, et al. Achieving human parity on automatic chinese to english news translation. *arXiv preprint arXiv:1803.05567*, 2018.

[16] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[17] Sébastien Jean and Kyunghyun Cho. Context-aware learning for neural machine translation. *arXiv preprint arXiv:1903.04715*, 2019.

[18] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[19] Philipp Koehn. *Statistical machine translation*. Cambridge University Press, 2009.

[20] Philipp Koehn. *Neural Machine Translation*. Cambridge University Press, 2020.

[21] Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the association for computational linguistics companion volume proceedings of the demo and poster sessions*, pages 177–180, 2007.

[22] Samuel Läubli, Rico Sennrich, and Martin Volk. Has machine translation achieved human parity? a case for document-level evaluation. *arXiv preprint arXiv:1808.07048*, 2018.

[23] Bei Li, Hui Liu, Ziyang Wang, Yufan Jiang, Tong Xiao, Jingbo Zhu, Tongran Liu, and Changliang Li. Does multi-encoder help? a case study on context-aware neural machine translation. *arXiv preprint arXiv:2005.03393*, 2020.

[24] Siyou Liu and Xiaojun Zhang. Corpora for document-level neural machine translation. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 3775–3781, 2020.

[25] Lesly Miculicich, Dhananjay Ram, Nikolaos Pappas, and James Henderson. Document-level neural machine translation with hierarchical attention networks. *arXiv preprint arXiv:1809.01576*, 2018.

[26] Daniel Ortiz Martínez. *Advances in fully-automatic and interactive phrase-based statistical machine translation*. PhD thesis, Universitat Politècnica de València, 2011.

[27] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

[28] Álvaro Peris and Francisco Casacuberta. Nmt-keras: a very flexible toolkit with a focus on interactive nmt and online learning. *arXiv preprint arXiv:1807.03096*, 2018.

[29] Thierry Poibeau. *Machine translation*. MIT Press, 2017.

[30] Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681, 1997.

[31] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015.

[32] Zhixing Tan, Shuo Wang, Zonghan Yang, Gang Chen, Xuancheng Huang, Maosong Sun, and Yang Liu. Neural machine translation: A review of methods, resources, and tools. *AI Open*, 1:5–21, 2020.

[33] Jörg Tiedemann. Parallel data, tools and interfaces in opus. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*.

[34] Jörg Tiedemann and Yves Scherrer. Neural machine translation with extended context. *arXiv preprint arXiv:1708.05943*, 2017.

[35] Zhaopeng Tu, Yang Liu, Shuming Shi, and Tong Zhang. Learning to remember translation history with a continuous cache. *Transactions of the Association for Computational Linguistics*, 6:407–420, 2018.

[36] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[37] Elena Voita, Pavel Serdyukov, Rico Sennrich, and Ivan Titov. Context-aware neural machine translation learns anaphora resolution. *arXiv preprint arXiv:1805.10163*, 2018.

[38] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. Hierarchical attention networks for document classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 1480–1489, 2016.

[39] Jiacheng Zhang, Huanbo Luan, Maosong Sun, Feifei Zhai, Jingfang Xu, Min Zhang, and Yang Liu. Improving the transformer translation model with document-level context. *arXiv preprint arXiv:1810.03581*, 2018.