UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Detección del idioma original en textos traducidos por máquina

Trabajo de graduación presentado por Pablo Gadhi Rodríguez Marcucci
para optar al grado académico de Licenciado en Ciencias de la
Computación y Tecnologías de la Información

Guatemala,

2020

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Detección del idioma original en textos traducidos por máquina

Trabajo de graduación presentado por Pablo Gadhi Rodríguez Marcucci
para optar al grado académico de Licenciado en Ciencias de la
Computación y Tecnologías de la Información

Guatemala,

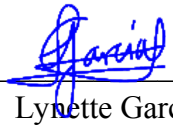
2020

Vo. Bo. :

(f) 

Lynette García

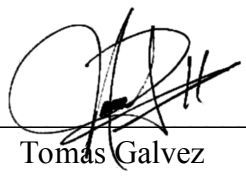
Tribunal Examinador:

(f) 

Lynette García

(f) 

Douglas Barrios

(f) 

Tomás Galvez

Fecha de aprobación: Guatemala, 14 de Diciembre 2020

PREFACIO

Desde que comencé a explorar un poco el tema en las clases de la universidad me ha gustando mucho todo lo que gira alrededor del Aprendizaje de Máquina, en especial el Procesamiento de Lenguaje Natural. Me parece uno de los retos más interesantes del ámbito computacional pues está completamente ligado a la lingüística, uno de los aspectos más importantes de las civilizaciones humanas. Esta es la razón principal por la que escogí trabajar en este proyecto, luego de cruzarme con varias investigaciones relacionadas con el tema.

Le agradezco a mi asesora Lynette García y a mi catedrático del curso de Diseño e Innovación en Ingeniería Tomás Gálvez por guiarme en el transcurso de este proyecto para que no cometiera tantos errores que podría haber lamentado profundamente. A mi familia por apoyarme en todo momento y de todas las maneras posibles para que consiguiera graduarme de la carrera de computación, y también por soportar los malos humores que me surgieron durante estos años. Por último agradezco las oportunidades que los eventos aleatorios de mi vida me han dado, los cuales me han permitido crecer y me han echo la persona que soy hoy.

ÍNDICE

LISTA DE CUADROS

LISTA DE FIGURAS

RESUMEN

El siguiente trabajo propone un método para la identificación del idioma original de un texto traducido automáticamente a español, inglés, alemán o francés. Esto se hizo con la justificación de expandir algunas investigaciones previas sobre el área de traducciones automáticas. Para esto se buscó desarrollar un sistema capaz de proporcionar información que pueda ser relevante para aplicaciones como encontrar la fuente de origen de un texto o mejorar una traducción automática producida por un motor de traducciones conocido. El proyecto se realizó utilizando un motor propio de traducciones con el cual se generaron varias Traducciones-Inversas de un conjunto de textos iterativamente, para cada idioma distinto al de entrada. Luego se calcularon las diferencias de cada Traducción-Inversa producida en las iteraciones utilizando el puntaje BLEU, con el fin producir una tabla que contiene dichas diferencias para cada idioma del conjunto propuesto. Por último se entrenaron cuatro clasificadores utilizando la información de dicha tabla, de manera que intentaran clasificar un serie de traducciones como textos traducido desde el idioma español, inglés, alemán o francés. La mejor precisión obtenida por uno de estos clasificadores fue de 58%.

I. INTRODUCCIÓN

Se sabe que existe un fenómeno en el procesamiento de datos, en el cual la primera iteración de un algoritmo produce muchas más variaciones en los resultados que el resto de iteraciones. Esto se puede observar en los algoritmos de ecualización de imágenes o audio, por ejemplo. Se pueden encontrar muchos menos cambios en los resultados de un algoritmo que se ejecutó sobre una imagen o segmento de audio que ya fue ecualizado antes que en sus versiones originales. Dicho fenómeno también se puede encontrar en algoritmos de traducción automática, por lo que es posible utilizarlo en aplicaciones relacionadas con el tema. (Nguyen-Son *et al.*, 2019)

En el artículo "Detecting Machine-Translated Text using Back Translation" se propone un método para detectar traducciones automáticas utilizando el mismo fenómeno. Este consiste en traducir un texto de entrada hacia un idioma intermedio para luego traducirlo de regreso a su idioma original y medir la cantidad de diferencias entre ambos. La idea de hacer esto es que se deberían encontrar más diferencias entre la Traducción-Inversa de un texto y su versión de entrada, si dicho texto es completamente original, y menos diferencias si este ya es una traducción generada por el mismo algoritmo. (Nguyen-Son *et al.*, 2019)

Partiendo de ese método, se tiene la hipótesis de que al generar las Traducciones-Inversas de un texto traducido automáticamente sobre un conjunto de idiomas intermedios en el que se encuentra el idioma de la versión original de dicho texto, la cantidad de diferencias mostradas entre el texto de entrada y la Traducción-Inversa sobre dicho idioma debería ser menor que la del resto. Es decir, al generar varias Traducciones-Inversas de un texto de entrada en paralelo, del cual se sabe que es una traducción, utilizando varios idiomas intermedios, se debería observar que una contiene menos cambios que el resto, si esta es la que se generó utilizando el idioma original como idioma intermedio. De la misma manera, si dicho proceso se repite a lo largo de varias iteraciones, se notará que las Traducciones-Inversas convergen a un conjunto determinado de palabras mucho más rápido, si estas fueron producidas sobre el idioma original de la traducción de entrada.

Bajo esta hipótesis se cree que es posible utilizar un método similar al que se utiliza en el artículo "Detecting Machine-Translated Text using Back Translation" (Nguyen-Son *et al.*, 2019) para detectar el idioma original de una traducción automática, si este se encuentra dentro de un conjunto determinado de lenguajes. A continuación se propone un proyecto que trata de explorar un poco dicha hipótesis. Primero se expondrán los objetivos y la justificación del mismo. Luego se presentarán los conceptos clave y las teorías detrás del proyecto, en el Marco Teórico. Después se explicará la metodología propuesta y se mostrarán los resultados obtenidos con dicha metodología. Por último, se analizarán estos resultados y se presentarán las conclusiones del proyecto. En las últimas páginas se encuentra toda la bibliografía citada a lo largo de este documento y algunos anexos.

II. OBJETIVOS

A. Objetivo general:

Detectar el idioma original de textos traducidos por máquina desde los idiomas español, inglés, alemán o francés hacia alguno de ellos.

B. Objetivos específicos:

- Desarrollar uno o más modelos de aprendizaje de máquina que sean capaces de traducir textos escritos en los idiomas propuestos.
- Entrenar dichos modelos hasta alcanzar puntajes cercanos a los obtenidos en otras investigaciones relacionadas con el tema, en métricas de evaluación automática de traducciones.
- Implementar un algoritmo que clasifique un texto traducido por máquina hacia uno de los idiomas propuestos, según su idioma original, con el apoyo del modelo o los modelos anteriores.

III. JUSTIFICACIÓN

Con los avances tecnológicos y el nivel de conectividad global que se ha adquirido, la necesidad de romper las barreras del lenguaje se ha vuelto más importante que nunca (Baig, 2020). Habiendo más de 6000 idiomas alrededor del mundo, es natural que se busque desarrollar tecnologías que ayuden a traducirlos y entendernos fácilmente (Cowen, 2017) Esta búsqueda impulsó la creación de los sistemas de traducción automática (TA) que se conocen hoy en día, los cuales han mejorado significativamente con las técnicas de Aprendizaje profundo desarrolladas en los últimos años (Chotard, 2017).

Las principales ventajas que tienen los sistemas de TA sobre los traductores humanos son la inmediatez y la accesibilidad que proveen. Herramientas como Google Translate pueden ser integradas en diversos sistemas empresariales y accedidas con un par de clics. Por ejemplo, en el área de atención al cliente, cualquier consulta o reclamo puede ser procesado por el personal de una empresa con mucho mayor velocidad si se traducen los mensajes y correos electrónicos de los clientes que hablan un idioma extranjero con dichos sistemas. (Chotard, 2017)

Sin embargo, ningún sistema de TA ha logrado alcanzar el nivel de precisión y calidad que ofrecen los traductores humanos, en especial en traducciones de textos largos (Chotard, 2017). Es por eso que es necesario seguir las investigaciones y el desarrollo de estos sistemas, para que poco a poco se vayan mejorando hasta lograr introducirlos en ámbitos más sensibles como los legales, médicos y financieros.

Una de las áreas de Procesamiento de lenguaje natural que apoya al desarrollo de sistemas de TA y se ha estado estudiando durante varios años, es la de detección de traducciones (Kurokawa *et al.*, 2009; Nguyen-Son *et al.*, 2019). No obstante, la mayoría de estas están enfocadas en detectar los patrones que diferencian una traducción realizada por un humano de su fuente original, con el propósito de mejorar los sistemas basados en Traducción automática estadística (SMT, del inglés Statistical Machine Translation) (Baroni & Bernardini, 2006, como se cita en Kurakawa *et al.*, 2009). El estudio de Hoang-Quoc Nguyen-Son y sus colegas (2019) es uno de los únicos en enfocarse directamente en detectar traducciones generadas automáticamente.

El propósito de este proyecto es el de expandir las investigaciones mencionadas anteriormente, desarrollando un método que prediga el idioma original de una TA. Dicha información puede ser relevante para facilitar la búsqueda de la versión original de cualquier texto, para la restauración de un texto traducido por una máquina o para mejorar la calidad de una TA producida por un sistema conocido.

IV. MARCO TEÓRICO

A. Aprendizaje de máquina

Aprendizaje de máquina se le llama a una serie de algoritmos que utilizan técnicas estadísticas para encontrar patrones en grandes cantidades de datos (Hao, 2018). A diferencia de la computación tradicional, los algoritmos de Aprendizaje de máquina no cuentan con instrucciones directas para encontrar dichos patrones. En lugar de eso, a este tipo de algoritmos se les "entrena" para realizar dicha tarea, cambiando los parámetros de una serie de funciones matemáticas en relación a los resultados obtenidos al operarlas sobre un conjunto de datos de entrenamiento (Heath, 2018).

El Aprendizaje de máquina se suele dividir en tres categorías: Aprendizaje supervisado, Aprendizaje no supervisado y Aprendizaje por refuerzo. El Aprendizaje supervisado consiste en etiquetar los datos para "darle una idea" a los algoritmos sobre los patrones que tienen observar. En contraste, los datos que consume un algoritmo de Aprendizaje no supervisado no tienen ninguna etiqueta y es el mismo algoritmo el que tiene que generarlas. Usualmente, esto se logra agrupando los datos con los patrones que encuentra. Por último, los algoritmos de Aprendizaje por refuerzo utilizan un mecanismo de recompensas y penalizaciones, las cuales se otorgan dependiendo de cuanto se aleje o se acerque el algoritmo a su objetivo (Hao, 2018).

B. Traducción automática

La noción de realizar traducciones automáticas de lenguajes naturales se tiene desde el año 1933, pero se comenzó investigar en 1947 con la llegada de las primeras computadoras digitales. Sin embargo, las primeras implementaciones no fueron exitosas. Fue hasta la década de los 90s que comenzó el auge de la traducción automática (TA), con el surgimiento de la Traducción automática estadística (SMT por sus siglas en inglés, Statistical Machine Translation). Este nuevo enfoque permitió que se implementara la traducción automática en aplicaciones como reconocimiento de voz y síntesis del habla. Adicionalmente, luego de la invención del internet, se empezó a utilizar la traducción automática para traducir páginas Web y correos electrónicos. (Hutchins, 2001)

Desde entonces la mayoría de algoritmos utilizaban alguna forma de SMT, hasta que en el año 2014 se publicó un artículo que propuso el primer acercamiento de utilizar redes neuronales para resolver el problema de la traducción automática (Pestov, 2018). Este artículo partió del gran éxito que se consiguió unos años antes para aplicaciones como el reconocimiento de objetos y el reconocimiento de vos. Los autores propusieron utilizar el modelo de una Red neuronal recurrente para que aprendiera representaciones de frases utilizadas en técnicas de SMT (Cho *et al.*, 2014).

Luego de la publicación del artículo mencionado se comenzaron a proponer distintos acercamientos al problema de TA, desde Redes recurrentes bidireccionales y la arquitectura llamada Long Short Term Memory (LSTM) (Pestov, 2018) hasta el diseño propuesto por Google en el 2016, nombrado Google's Neural Machine Translation System (GNMT) (Wu *et al.*, 2016). En esos dos años se lograron sobrepasar los resultados obtenidos durante los últimos veinte años de desarrollo de sistemas de TA. Los sistemas de Traducción automática neuronal (NMT por sus siglas en inglés, Neural Machine Translation) han logrado 50% menos errores de ordenamiento de palabras, 17% menos errores léxicos y 19% menos errores gramaticales (Pestov, 2018).

C. Redes neuronales artificiales

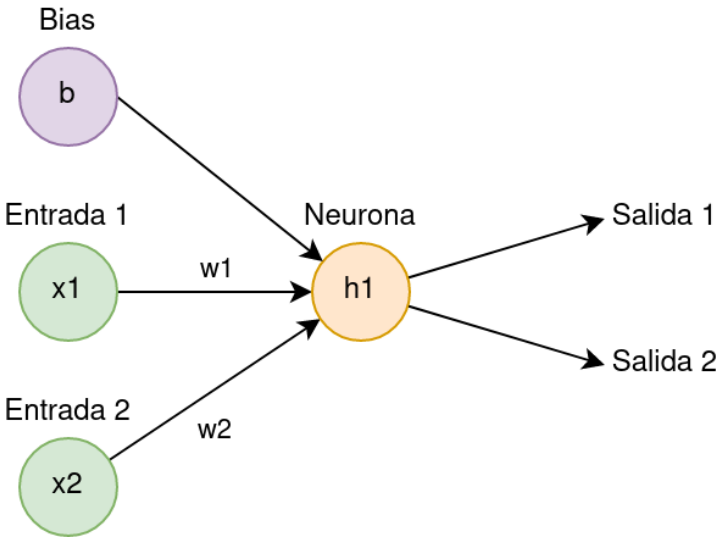
Una Red neuronal artificial es un algoritmo de Aprendizaje de máquina que esta basado, de una manera abstracta, en la forma que opera el cerebro humano. Una red neuronal humana esta compuesta por millones de nodos de procesamiento conectados densamente entre sí. La idea de una Red neuronal artificial es simular esta interconexión de nodos de procesamiento, organizándolos en capas conectadas entre sí que reciben los datos de la capa anterior, los procesan y le pasan los resultados a la capa siguiente (Hardesty, 2017).

Al nodo de procesamiento de una capa se le llama neurona. Estas se encargan de recibir los resultados de la capa anterior, multiplicarlos por el peso que le corresponde a la conexión en la que fue trasportado cada valor recibido, sumar todas estas multiplicaciones y aplicarle una función de activación a dicha suma. La función de activación, por lo general, tiene la tarea de simular el comportamiento de "encender" y "apagar" una neurona, modificando el valor que una neurona le otorga a la siguiente capa (Yiu, 2019). Existen varias funciones de activación, una de las más famosas y la que se utilizará para ejemplificar el funcionamiento de una Red neuronal artificial es la función sigmoide:

$$S(x) = \frac{1}{1+e^{-x}} \quad (1)$$

Esta función en particular se utiliza para convertir cualquier valor real en un número entre 0 y 1, de manera que se pueda interpretar como un valor de probabilidad (Wood, 2020).

Figura 1: Estructura de una Red neuronal artificial simple

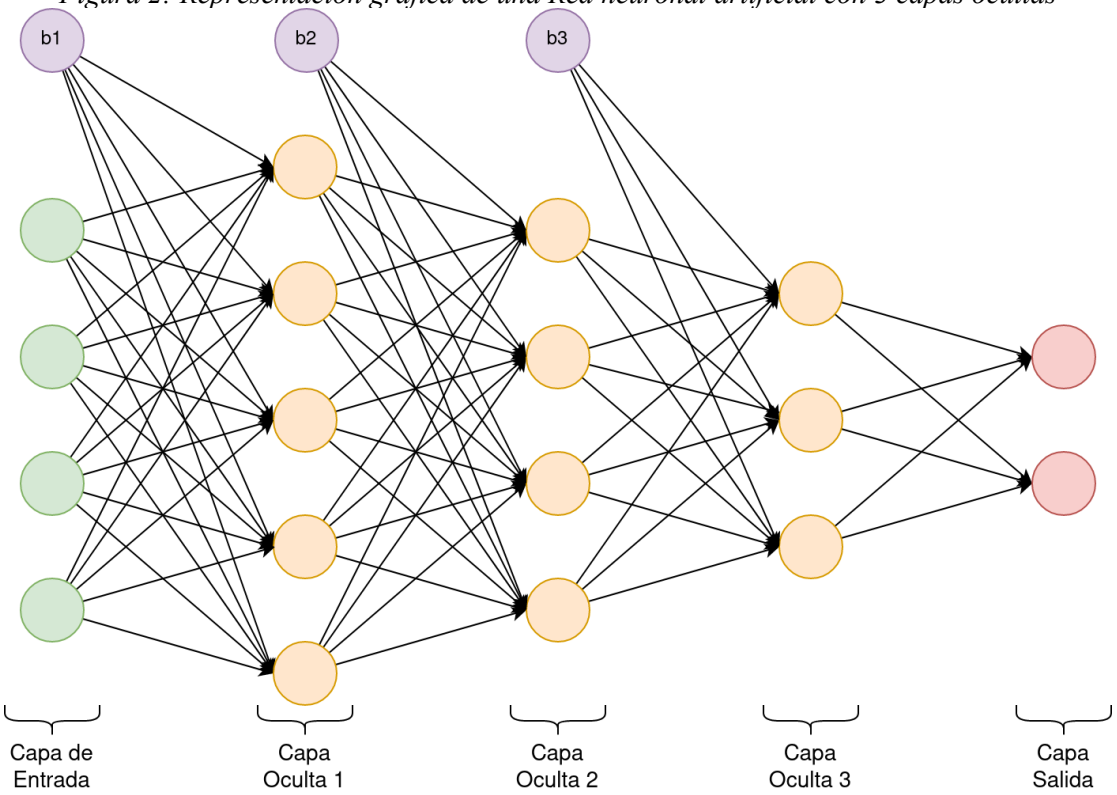


En la Red neuronal de la figura anterior, el cálculo que haría la neurona h1 sería el siguiente:

$$h_1 = S(w_1 * x_1 + w_2 * x_2 + b) \quad (2)$$

asumiendo que la función de activación a utilizar es la función sigmoide. A todas las capas que contienen neuronas se les llama capas ocultas. Usualmente, todas las neuronas de estas capas se encuentran conectadas a un nodo adicional al cual se le llama el nodo de parcialidad (Bias en inglés) (Yiu, 2019). Este se utiliza para poder trasladar la función de activación a lo largo del eje vertical (Gebel, 2020). Una Red neuronal artificial un poco más compleja cuenta con varias entradas, varias capas ocultas cuyos nodos se encuentran completamente conectados con los de sus capas vecinas, y se busca producir varias salidas.

Figura 2: Representación gráfica de una Red neuronal artificial con 3 capas ocultas



Los valores de cada peso y los de parcialidad son los que una Red neuronal debe aprender para acercase los más posible a los valores de salida deseados. Esencialmente, esto se logra de las misma manera que muchos otros modelos de Aprendizaje de máquina, definiendo una función de costo y tratando de minimizarla, utilizando una técnica llamada "Descenso del Gradiente" (Yiu, 2019). Esta técnica consiste en actualizar los parámetros θ en la dirección opuesta al gradiente $\nabla_\theta J(\theta)$ de una función objetivo $J(\theta)$ (la función de costo en este caso), respecto a dichos parámetros (Ruder, 2016).

Sin embargo, las Redes neuronales artificiales tienen la particularidad de que al cambiar el peso o el valor de parcialidad de una neurona, toda la red se verá afectada. Debido a la naturaleza de la arquitectura, un cambio en cualquier parte de la red se propaga en el resto de sus capas, pues estas se encuentran interconectadas. Por esta razón es necesario utilizar una técnica llamada "Propagación Hacia Atrás" en conjunto con el "Descenso del Gradiente". Esta consiste en propagar el error de las predicciones en la dirección contraria en la que se pasaron los resultados de cada capa para generar dichas predicciones. Al hacer esto, cada neurona de cada una de las capas ocultas es capaz de calcular cuánto atribuyen sus pesos y su valor de parcialidad al error de las neuronas a la que se encuentra conectada. De esta manera es capaz de actualizar dichos valores para tratar de disminuir el error que genera (Yiu, 2019).

D. Transformer

El Transformer es una arquitectura de redes neuronales utilizada para resolver

problemas de traducción de secuencias. Esta funciona con un modelo Codificador-Decodificador al igual que la mayoría de algoritmos de Aprendizaje profundo que se utilizan para este tipo de problemas. Sin embargo, el Transformer funciona utilizando mecanismos de atención únicamente, eliminando la necesidad de la recurrencia o las convoluciones que se utilizan en otros algoritmos comunes. Esto facilita su paralelización, permitiéndole alcanzar tiempos de entrenamiento más cortos y puntajes de rendimiento más altos que el resto de algoritmos utilizados previamente. (Vaswani *et al.*, 2017)

Mecanismos de atención (Self-Attention)

La idea principal detrás de utilizar la arquitectura Transformer es la de crear una representación lógica de los símbolos relevantes respecto a uno específico en una secuencia. Para esto, el Transformer utiliza un mecanismo especial del concepto de Atención (Attention) llamado Atención propia (Self-Attention). El concepto usual de Atención se utiliza en las Redes neuronales recurrentes para guardar información de los símbolos procesados anteriormente al símbolo actual. En contraste, en la Atención propia se busca encontrar y guardar la relevancia de un símbolo para el resto en la secuencia, incluyéndose a sí mismo. Por esta razón, el puntaje de atención siempre será más alto para los pares de símbolos iguales. Por motivos de optimización, la Atención propia se calcula sobre conjuntos de secuencias, utilizando operaciones matriciales, de la siguiente manera:

$$Z = softmax\left(\frac{Q\,x\,K^T}{\sqrt{(d_k)}}\right)V \quad (3)$$

Donde Q es la matriz de "consultas", K la matriz de "llaves", V la matriz de "valores" y d_k la dimensión horizontal de la matriz K (el largo de los vectores k). Las matrices Q, K y V se obtienen multiplicando la matriz de entrada X con las matrices de pesos W^Q, W^K y W^V:

$$Q = X\,x\,W^Q \quad (4)$$
$$K = X\,x\,W^K \quad (5)$$
$$V = X\,x\,W^V \quad (6)$$

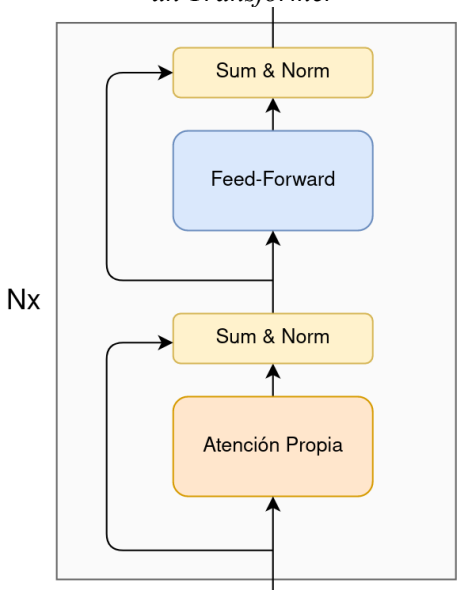
Dichas matrices de pesos son las que se encontrarán durante el entrenamiento del modelo, por lo que son inicializadas con valores aleatorios. Es necesario mencionar que la matriz X contiene las representaciones numéricas de los símbolos del conjunto de datos y no los símbolos en sí. Estas representaciones se consiguen con el algoritmo de empotramiento de palabras, el cual crea relaciones entre palabras o símbolos y números reales. (Alammar, 2018)

En resumen, la Atención propia consiste en encontrar un conjunto de consultas Q y un conjunto de llaves K que al ser utilizadas con el conjunto de valores V, que también es necesario encontrar, nos proporcionará una idea de qué tan importante es un símbolo de una secuencia, para el resto de símbolos de la misma secuencia.

Detalles de la arquitectura

Como se menciona anteriormente, el Transformer sigue un modelo Codificador-Decodificador, pero a diferencia de otros modelos, la arquitectura Transformer está compuesta por una pila de N codificadores conectados entre sí y sus respectivos decodificadores, también conectados entre sí. Los codificadores están compuestos por una capa de Atención propia y una red neuronal Feed-Forward. Luego de cada una de estas capas se encuentra una capa adicional que suma y normaliza las entradas de la capa con las salidas de la misma. Este nuevo resultado es el que se le pasa a la siguiente capa o al siguiente codificador. (Alammar, 2018)

Figura 3: Estructura del codificador de un Transformer



Cada codificador recibe la salida del anterior, con la excepción del primero, el cual recibe una versión especial de las entradas empotradas que contiene información de las posiciones de los símbolos en las secuencias y su distancia en relación al resto de símbolos (Alammar, 2018). Dicha información se puede codificar en las entradas empotradas de varias maneras. La que se propone en el artículo original del Transformer, "Attention Is All You Need" (Vaswani *et al.*, 2017), consiste en calcular las funciones seno y coseno para diferentes frecuencias:

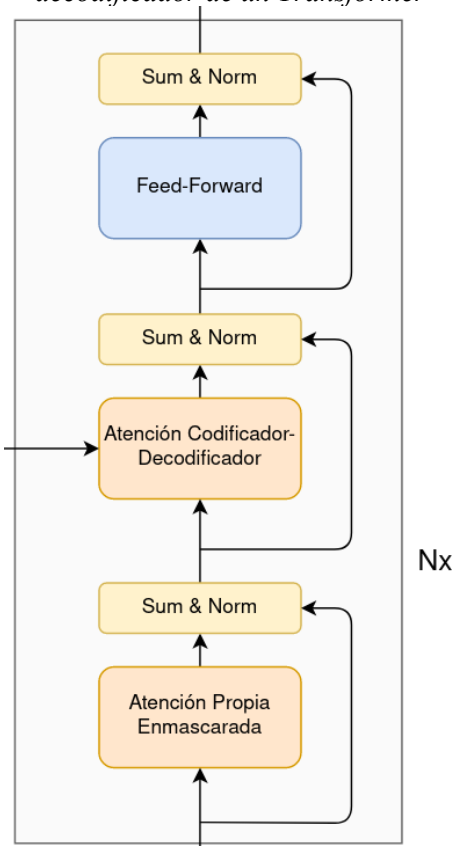
$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d}}\right) \quad (7)$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d}}\right) \quad (8)$$

donde pos es la posición del símbolo, i es la dimensión en la que se encuentra y d es el tamaño de la dimensión horizontal de la matriz que contiene las entradas empotradas.

En cuanto al decodificador, este está formado por tres capas principales: una capa de Atención propia enmascarada, una capa de Atención Codificador-Decodificador y una red neuronal Feed-Forward. La capa de Atención propia enmascarada funciona de la misma manera que la capa de Atención propia de los codificadores, con la diferencia de que utiliza una máscara para ignorar los valores de todos los símbolos que le siguen al que se está procesando en ese momento. Esto se hace para que el decodificador pueda "prestarles atención" únicamente a los símbolos de la secuencia que se encuentran antes del actual, pues su tarea es predecir los símbolos siguientes. La capa siguiente es otra capa de Atención y también tiene el mismo funcionamiento que la del codificador; sin embargo, esta utiliza las matrices K y V producidas por su codificador correspondiente y la matriz Q que se genera en la capa previa de Atención. Por último, la red neuronal Feed-Forward es idéntica a la que se utiliza en el codificador y, de la misma manera que en el codificador, todas las capas están conectadas a una capa intermedia que suma y normaliza las matrices de entrada y salida de cada capa.

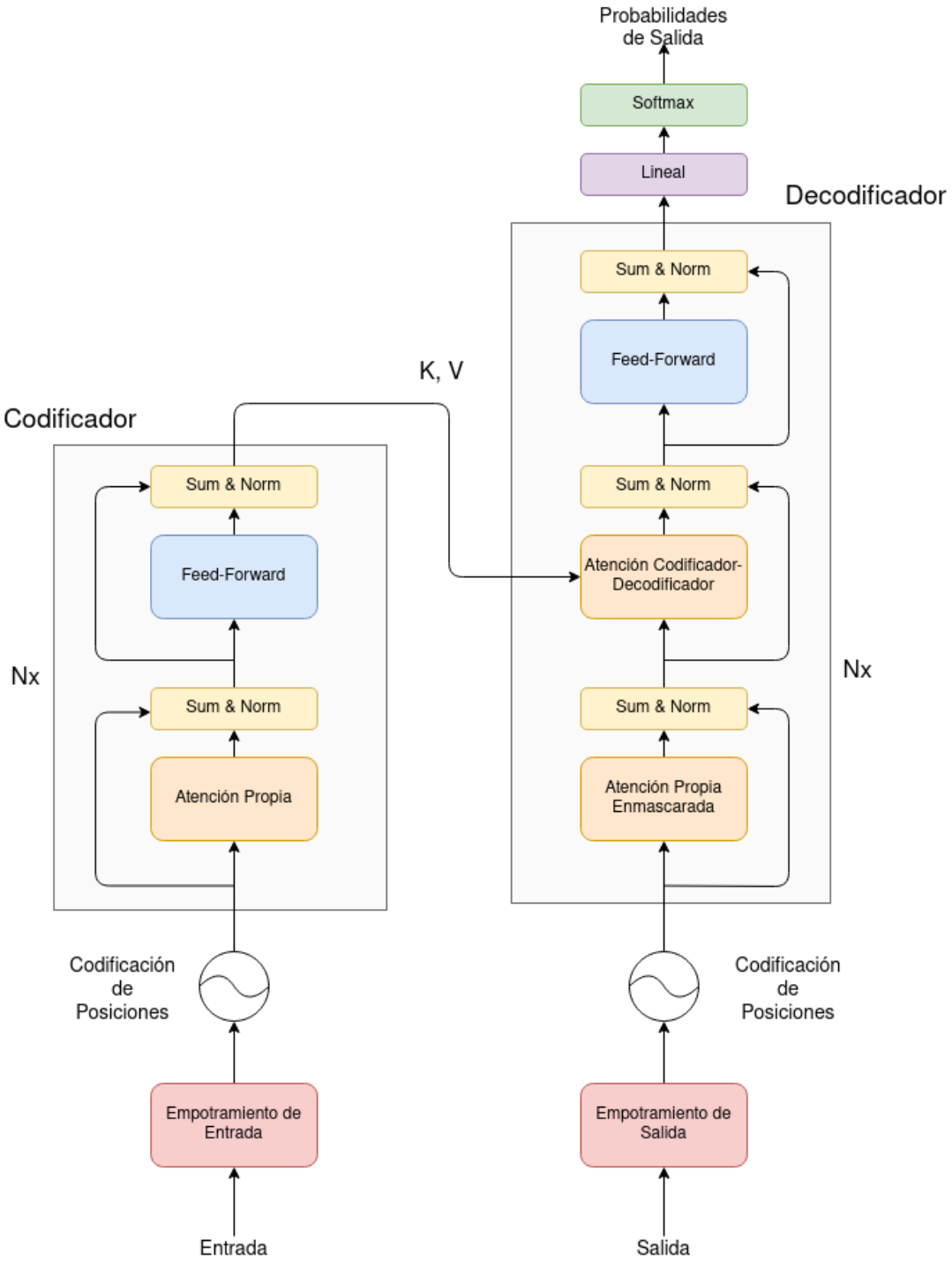
Figura 4: Estructura del decodificador de un Transformer



Al final del último decodificador se produce un vector de números reales que se necesita convertir a probabilidades para todos los símbolos del conjunto objetivo. Para lograr esto se utiliza una capa lineal que proyecta el vector resultante a un nuevo vector del mismo tamaño que el conjunto objetivo. Por último se le aplica la función softmax a este último vector para representar la probabilidad del símbolo a elegir dada la secuencia actual y que la suma de

todas las probabilidades sea 1.

Figura 5: Arquitectura completa del Transformer



E. Métricas de evaluación para traducciones

BLEU

El puntaje BLEU probablemente es la métrica más utilizada para evaluar el rendimiento de un sistema de TA. Esta ganó su popularidad por ser la primera métrica en mostrar una alta correlación con los juicios humanos sobre la calidad de una traducción. Una manera simplista de explicar su funcionamiento es la de medir la correspondencia que tiene un texto traducido por una máquina con una traducción del mismo texto realizada por un humano, entre más cercano sea el texto traducido por máquina a la traducción humana, mayor se considera su calidad. Es por esta razón que se cuestionan mucho los resultados de esta métrica, pues depende totalmente de los textos que se utilicen como referencia y no contempla el uso de sinónimos. Esto hace que el puntaje BLEU no pueda examinar el potencial de una traducción correctamente (Vashee, 2019).

BLEU se calcula de la siguiente manera:

$$BLEU = BP \cdot \exp\left(\sum_{n=1}^{N} w_n \log\left(p_n\right)\right) \quad (9)$$

donde BP es la penalización de brevedad,

$$BP = \begin{cases} 1 & if \;\; c > r \\ e^{\left(1-\frac{r}{c}\right)} & if \;\; c \leq r \end{cases} \quad (10)$$

c es largo de la traducción candidata y *r* es el largo de la referencia. La sumatoria es la media geométrica de la precisiones modificadas p_n para todos los n-gramas de 1 hasta *N*, utilizando los pesos w_n (Papineni *et al.*, 2001).

p_n se puede calcular de la siguiente manera:

$$p_n = \frac{cantidad\ de\ n-gramas\ comunes}{cantidad\ de\ n-gramas\ candidatas} \quad (11)$$

METEOR

METEOR es otra métrica automática de evaluación de traducciones. Esta fue creada para resolver varios problemas que tiene la métrica BLEU, como la falta de considerar el concepto de Recodar (Recall por su nombre en inglés), la proporción de los n-gramas que concuerdan sobre la cantidad total de n-gramas en la referencia. Este concepto es importante a la hora de evaluar una traducción, pues refleja en qué grado cubre la traducción todo el contenido que debería ser traducido. (Banerjee & Lavie, 2005)

Para calcular METEOR se necesitan la precisión de uni-gramas:

$$P = \frac{m}{w_t} \quad (12)$$

Donde m es la cantidad de uni-gramas mapeados de la traducción a la referencia y w_t la cantidad total de uni-gramas en la traducción. También es necesario calcular la razón de Recordar sobre los uni-gramas:

$$R = \frac{m}{w_r} \quad (13)$$

Donde m es el mismo valor que en la precisión y w_r es la cantidad total de uni-gramas en la referencia. Con la precisión y el valor de Recordar se calcula la media armónica, poniéndole la mayor cantidad de peso sobre el valor de Recordar:

$$F - media = \frac{10\,PR}{R + 9P} \quad (14)$$

Debido a que las operaciones anteriores solo toman en cuenta los aciertos de los uni-gramas, METEOR calcula un valor de penalización para incluir aciertos más largos. Para esto, los uni-gramas mapeados entre la traducción y la referencia se agrupan en los pedazos más cortos posibles que mantienen la adyacencia entre ambos textos. Esto hará que el número de pedazos sea más corto entre más largos son los n-gramas. Luego la penalización se calcula de la siguiente manera:

$$Penalización = 0.5 * \left(\frac{número\,de\,pedazos}{uni - gramas\ acertados} \right) \quad (15)$$

Finalmente, se puede calcular METEOR:

$$METEOR = F - media * (1 - Penalización) \quad (16)$$

(Banerjee & Lavie, 2005)

F. Máquina de vectores de soporte

Una máquina de vectores de soporte (SVM por sus siglas en inglés, del nombre Support Vector Machine) es un algoritmo de Aprendizaje de máquina cuyo objetivo es encontrar un hiperplano en un espacio de N dimensiones, en el que N es el número de características, el cual permita clasificar una serie de puntos que representan los datos en dicho espacio. Sin embargo, para separar dichos puntos en una serie de clases o categorías, existen varios hiperplanos posibles. Es por esta razón que se debe encontrar el hiperplano con el máximo margen, es decir, se debe encontrar el hiperplano con la mayor distancia entre sí mismo y los puntos que representan los datos de todas las clases. Esto permite que en un futuro se puedan clasificar datos externos con mayor confianza. (Gandhi, 2018)

Vectores de soporte se le llaman a los puntos que se encuentran más cercanos al hiperplano. Estos influencian la posición y orientación del mismo, y según ellos se maximiza el margen del clasificador. La función de costo que se utiliza para maximizar dicho margen es la siguiente:

$$c(x, y, f(x)) = \begin{cases} 0, & if\ y * f(x) \geqslant 1 \\ 1 - y * f(x), & else \end{cases} \quad (17)$$

El costo es 0, si el valor de la predicción y el valor real tienen el mismo signo. De lo contrario, se calcula la función de pérdida. (Gandhi, 2018)

G. Árbol de decisiones

Un Árbol de decisiones es un algoritmo de Aprendizaje de máquina que puede utilizarse tanto parta problemas de regresión como para problemas de clasificación. Este consiste en ir construyendo un árbol binario cuyas ramas representan los resultados de un decisión que se examina sobre una de las características del set de datos. Para decidir que característica utilizar para crear una nueva división de ramas en el árbol, se toman en cuenta todas las características del conjunto de datos de entrada y se elige una decisión que genere dos grupos con respuestas similares pero también produzca el menor costo posible. La idea detrás de esto es elegir la división que causará la menor pérdida de precisión posible. Para problemas de clasificación, la función de costo utilizada es el puntaje Gini, el cual se describe de la siguiente manera:

$$c = \sum (pk * (1 - pk)) \quad (18)$$

en donde pk es la proporción de los valores de entrada de las misma clase presentes en un grupo en particular, seleccionado por una división en el árbol. Este puntaje da una idea de qué tan buena es una división, revisando qué tan mezclados están los datos de entrada que pertenecen a las categorías a encontrar entre los dos grupos generados por dicha división. Una división perfecta es aquella que genere un grupo en el que se encuentren todos los datos de una clase. (Gupta, 2017)

H. K vecinos más cercanos

El algoritmo de K vecinos más cercanos (KNN por sus siglas en inglés, K-Nearest Neighbors) funciona tanto como método de Aprendizaje de máquina supervisado, como no supervisado. Este es probablemente el más sencillo de los que se utilizaron en la investigación y al igual que el resto, se utilizó como un algoritmo supervisado para problemas de clasificación. Dicho algoritmo consiste en encontrar una cantidad predefinida de puntos, dentro del espacio del conjunto de datos de entrenamiento, que se encuentren lo más cercano posible entre ellos. A cada conjunto de puntos encontrado se le asigna una etiqueta que

represente una de las clases a clasificar, de manera en que se logren etiquetar todos los puntos encontrados en un determinado espacio como los datos que pertenecen a una de las categorías. (scikit-learn developers, 2020)

La cantidad de puntos cercanos a encontrar puede ser tanto una constante, como una densidad. El método para calcular que tan cercano se encuentra un punto de otro puede ser cualquier métrica de distancia. Sin embargo, el más común es el de la distancia euclidiana. (scikit-learn developers, 2020)

V. METODOLOGÍA

El proyecto se elaboró en dos partes principales: el motor de traducciones basado en un modelo NMT* multilingüe y un clasificador de textos que utiliza las diferencias de las Back-Translations de las oraciones de entrada, generadas con dicho motor. Para entrenar los modelos de ambas partes se utilizó un subconjunto distinto del corpus de oraciones paralelas que ofrece el Parlamento Europeo (Koehn, 2005).

A. Preprocesamiento de los datos del motor de traducciones

Durante el preprocesamiento del conjunto de datos para entrenar el motor de traducciones se buscó obtener un conjunto de oraciones totalmente alineadas para cada uno de los idiomas propuestos. Esto quiere decir que se seleccionaron únicamente oraciones que tuvieran una traducción en todos los idiomas utilizados en la investigación.

Para lograr esto, primero se generaron archivos en paralelo del conjunto original con el programa de alineación que se encuentra junto con los datos originales del Parlamento Europeo. Se utilizó como eje central al idioma inglés, por lo que cada par de archivos paralelos contaba con oraciones escritas en inglés en un archivo y sus traducciones hacia uno de los idiomas restantes en el otro. Dichos archivos contenían algunas líneas vacías y etiquetas de metadatos en lenguaje XML, por lo que se removieron con un pequeño programa que mantuviera la alineación original y se asegurara de que cada línea de los archivos contuviera exactamente una oración. La Figura 15 en la sección de anexos muestra un ejemplo del estado del conjunto de los datos antes de realizar esta modificación y la Figura 16 muestra un ejemplo del mismo conjunto después de remover dichas líneas.

Luego se procedió a ejecutar el programa de alineación múltiple sobre el resultado anterior para que eligiera aleatoriamente 800,000 oraciones que tuvieran una traducción en todos los idiomas. Se estimó que dicho proceso tiene una complejidad de $O(n^2)$ por lo que este programa se diseñó alrededor de un sistema de múltiples hilos que aprovechara todo el procesamiento posible del CPU de la máquina que se utilizó.

Con el conjunto de oraciones seleccionadas se entrenó un modelo de BPE (por sus siglas en inglés, Byte Pair Encoding) para cada idioma, con el fin de aprender a separarlas en las sub-palabras más frecuentes y armar los vocabularios que forman el conjunto de datos. Este proceso se llevó acabo utilizando la librería Tokenizer de OpenNMT buscando generar un vocabulario de 30,000 símbolos distintos para cada idioma. Seguido del entrenamiento de los modelos de BPE, se puso en ejecución la separación de las oraciones del conjunto obtenido hasta el momento, en los símbolos encontrados por dichos modelos. En la Figura 17, mostrada en la sección de anexos, se puede observar un ejemplo de oraciones separadas por este proceso.

Por último se juntaron todas las oraciones seleccionadas, creando solamente dos archivos para todo el conjunto de datos. En el primero se encontrarían todas las oraciones

* Por sus siglas en inglés, Neural Machine Translation

repetidas tres veces en cada idioma fuente. Esto se hace así para que el modelo aprenda a traducir cada oración a su equivalente en cada uno de los idiomas distintos. En el segundo se encontrarían todas las traducciones de cada una de las oraciones fuente, en diferentes idiomas, generando un total de 9,600,000 de oraciones. Adicionalmente se le inyectó un símbolo adicional a cada oración del archivo fuente, con la información del idioma fuente y el idioma a traducir. Este símbolo es la clave para crear un modelo multilingüe, como se muestra en el artículo "Training Romance Multi-Way Model" (Senellart, 2018). Consultar las Figuras 18 y 19 para ver un ejemplo de las oraciones resultantes en dichos archivos.

Es necesario resaltar que las cantidades mencionadas anteriormente fueron las que se utilizaron para entrenar la versión final del modelo. Inicialmente se probó utilizar un mismo modelo de BPE para todos los idiomas con 32,000 símbolos distintos. Luego se probó utilizar un modelo distinto para cada idioma, utilizando 15,000 símbolos para cada modelo. Finalmente se decidió utilizar modelos distintos de 30,000 símbolos. Además se extendió el tamaño del conjunto de datos de entrenamiento para la versión final del modelo. Todas las versiones anteriores se entrenaron con 500,000 oraciones distintas, formando un conjunto final de 6,000,000 de oraciones.

B. Entrenamiento del motor de traducciones

Para el motor de traducciones se entrenó un modelo de tipo Transformer de seis capas generales, las cuales contaron con un tamaño de entrada de 512 para las capas de incrustación, las sub-capas del codificador y las del decodificador de cada capa general del modelo, un tamaño de entrada de 2048 para las sub-capas Feed-Forward, 8 cabezas de atención en las sub-capas de Atención propia y un valor de Dropout de 0.1. Este modelo se entrenó inicialmente durante 200,000 épocas en lotes de 4096 utilizando un GPU Nvida Tesla P100. La versión final del modelo se entrenó por 300,000 épocas, utilizando la misma configuración.

C. Arquitectura y exposición del motor de traducciones

Para facilitar el uso del motor de traducciones y agilizar algunas pruebas, se decidió abstraer el modelo en una modalidad de servidor. Por lo que primero se creó una pequeña abstracción de dicho modelo alrededor de la librería CTranslate2, una de las herramientas de OpenNMT que sirve para exponer modelos en un ambiente de producción. Esta librería usa varias optimizaciones que el módulo de traducción del núcleo de OpenNMT no tiene implementado, aumentando la velocidad de las traducciones, lo que permite que mejorar las condiciones del modelo para utilizarse en ambientes de producción (Klein, 2019). Luego se implementó un pequeño servidor REST, utilizando la librería FLASK, que encapsula dicha abstracción con el fin de poder hacer cualquier traducción por solicitudes REST a dicho servidor.

Por último, se creó una configuración de Docker para que el servidor se ejecute en un contenedor. Esto se hizo con el objetivo de poder utilizar el poder de procesamiento de un GPU, ya que la implementación de CTranslate2 para python solamente esta compilada con soporte para utilizar CUDA en un contenedor Docker. Por supuesto, también se contempló la

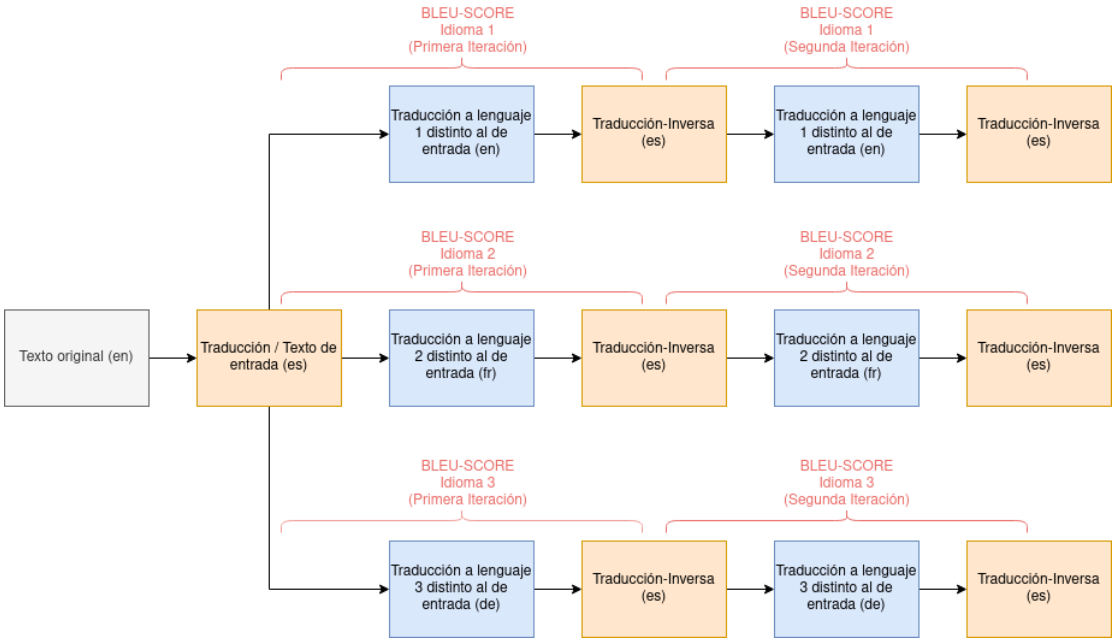
posibilidad de correr el servidor utilizando el CPU de la máquina únicamente, por lo que se generalizó la selección del dispositivo que Ctranslate2 debe utilizar para procesar las traducciones.

D. Preprocesamiento de los datos del clasificador

Durante el preprocesamiento de los datos del clasificador se buscó obtener como resultado una tabla con los puntajes BLEU entre las Traducciónes-Inversas de un conjunto de párrafos traducidos por una máquina. Para lograr esto, primero se generó un subconjunto de 10,000 oraciones totalmente alineadas de los datos del Parlamento Europeo, distinto a los usados en el entrenamiento del motor de traducciones, pero utilizando las mismas técnicas. Para aumentar la cantidad de variaciones que se pueden encontrar en una Traducción-Inversa, se unieron dichas oraciones en párrafos de un mínimo de 100 palabras. Luego se tradujeron estos párrafos a cada uno de los idiomas distintos al suyo con el motor de traducciones, generando un conjunto de 29,070 párrafos traducidos.

Luego se procedió a generar la tabla con apoyo del motor de traducciones y la librería NLTK para calcular los puntajes BLEU de las traducciones inversas sobre las traducciones de cada uno de los párrafos hacia los idiomas distintos al de entrada. Es decir, para cada párrafo, se genera una traducción de él mismo hacia cada uno de los idiomas distintos al actual. Luego se traducen de regreso dichas traducciones al idioma de entrada del párrafo y se calcula el puntaje BLEU entre estas Traducciones-Inversas y el párrafo de entrada, con el fin de medir la cantidad de diferencias entre ellos. Este proceso se repite una vez más, hasta generar un segundo puntaje BLEU para todas las segundas Traducciones-Inversas realizadas sobre las traducciones de los idiomas intermedios, utilizando las Traducciones-Inversas de la iteración anterior como referencia para el cálculo de BLEU.

Figura 6: Proceso de generación de una fila en la tabla de puntajes BLEU



Todos los puntajes obtenidos se asignan a las columnas de una matriz donde la fila tiene el mismo índice que el índice del párrafo al que le pertenecen y se asignan valores de cero para las columnas que representan los puntajes del idioma fuente del párrafo.

Al final se le concatenan tres columnas nuevas a la matriz generada, con la información del idioma de entrada, el idioma original (el idioma del párrafo antes de ser traducido por el motor de traducciones) y el largo del párrafo. Estos datos se trasladan a una tabla de la librería Pandas (The pandas development team, 2020) para su visualización y facilitar el resto del preprocesamiento, como la normalización de los valores y la eliminación de filas problemáticas. Un ejemplo de la tabla generada se muestra en la Figura 20 de la sección de anexos.

E. Entrenamiento del clasificador

Para entrenar el clasificador con la tabla generada que se describe en el inciso anterior se probaron cuatro algoritmos distintos, Red neuronal, una Máquina de vectores de soporte un Árbol de decisiones y un algoritmo de K vecinos más cercanos.

La arquitectura de la Red neuronal cuenta con 5 capas ocultas, una de 1024 nodos, una de 512, una de 128, una de 32 y una de 8. Para todas las capas ocultas se maneja un nodo adicional con el Bias y se les aplica la función de activación Leaky ReLU. Por último, a la salida de la red se le aplica un Dropout de 0.5 para evitar el problema de sobre-ajuste y una función Softmax para producir 4 valores que en total sumen 1. Esta red se entrenó por 4000 épocas.

La Máquina de vectores de soporte se configuró con un núcleo de Función de base polinomial, un parámetro de regularización de 2.0 y parámetro gamma de:

$$\frac{1}{n_{características} * varianza_{entrada}} \quad (19)$$

El árbol de decisiones se configuró para utilizar el puntaje Gini como criterio de decisión de una división. Por último, el algoritmo de K vecinos más cercanos se ejecutó utilizando 141 vecinos (aproximadamente la raíz cuadrada de la cantidad de datos de entrenamiento).

VI. RESULTADOS

A continuación se presentarán los puntajes obtenidos del modelo para el motor de traducciones, sobre el conjunto de pruebas y utilizando distintas configuraciones.

BLEU

Cuadro 1: Puntajes BLEU del modelo entrenado con un mismo BPE†, utilizando un vocabulario total de 32,000 símbolos y un conjunto de entrenamiento de 6,000,000 de oraciones, durante 200,000 épocas

Fuente\Objetivo	En	Es	Fr	De
En		0.339	0.327	0.241
Es	0.328		0.304	0.213
Fr	0.304	0.297		0.208
De	0.278	0.255	0.259	

Promedio: 0.279

Cuadro 2: Puntajes BLEU del modelo entrenado con distintos BPE, utilizando un vocabulario de 15,000 símbolos por idioma y un conjunto de entrenamiento de 6,000,000 de oraciones, durante 200,000 épocas

Fuente\Objetivo	En	Es	Fr	De
En		0.388	0.370	0.274
Es	0.391		0.355	0.252
Fr	0.352	0.339		0.24
De	0.331	0.308	0.310	

Promedio: 0.326

Cuadro 3: Puntajes BLEU del modelo entrenado con distintos BPE, utilizando un vocabulario de 15,000 símbolos por idioma y un conjunto de entrenamiento de 6,000,000 de oraciones, durante 300,000 épocas

Fuente\Objetivo	En	Es	Fr	De
En		0.384	0.368	0.272
Es	0.386		0.354	0.249
Fr	0.350	0.366		0.237
De	0.328	0.305	0.308	

† Por sus siglas en inglés, Byte Pair Encoding.

Promedio: 0.323

Cuadro 4: Puntajes BLEU del modelo entrenado con distintos BPE, utilizando un vocabulario de 30,000 símbolos por idioma y un conjunto de entrenamiento de 9,600,000 oraciones, durante 200,000 épocas

Fuente\Objetivo	En	Es	Fr	De
En		0.385	0.368	0.266
Es	0.391		0.353	0.244
Fr	0.351	0.334		0.233
De	0.331	0.302	0.309	

Promedio: 0.322

Cuadro 5: Puntajes BLEU del modelo entrenado con distintos BPE, utilizando un vocabulario de 30,000 símbolos por idioma y un conjunto de entrenamiento de 9,600,000 oraciones, durante 300,000 épocas

Fuente\Objetivo	En	Es	Fr	De
En		0.389	0.370	0.269
Es	0.395		0.354	0.248
Fr	0.356	0.337		0.237
De	0.334	0.305	0.312	

Promedio: 0.325

METEOR

Cuadro 6: Puntajes METEOR del modelo entrenado con un mismo BPE, utilizando un vocabulario total de 32,000 símbolos y un conjunto de entrenamiento de 6,000,000 de oraciones, durante 200,000 épocas

Fuente\Objetivo	En	Es	Fr	De
En		0.532	0.501	0.436
Es	0.554		0.475	0.400
Fr	0.529	0.491		0.393
De	0.502	0.445	0.426	

Promedio: 0.474

Cuadro 7: Puntajes METEOR del modelo entrenado con distintos BPE, utilizando un vocabulario de 15,000 símbolos por idioma y un conjunto de entrenamiento de 6,000,000 de oraciones, durante 200,000 épocas

Fuente\Objetivo	En	Es	Fr	De
En		0.584	0.550	0.478
Es	0.624		0.534	0.452
Fr	0.587	0.538		0.438
De	0.566	0.510	0.488	

Promedio: 0.529

Cuadro 8: Puntajes METEOR del modelo entrenado con distintos BPE, utilizando un vocabulario de 15,000 símbolos por idioma y un conjunto de entrenamiento de 6,000,000 de oraciones, durante 300,000 épocas

Fuente\Objetivo	En	Es	Fr	De
En		0.580	0.546	0.476
Es	0.621		0.532	0.450
Fr	0.584	0.534		0.435
De	0.566	0.507	0.486	

Promedio: 0.526

Cuadro 9: Puntajes METEOR del modelo entrenado con distintos BPE, utilizando un vocabulario de 30,000 símbolos por idioma y un conjunto de entrenamiento de 9,600,000 oraciones, durante 200,000 épocas

Fuente\Objetivo	En	Es	Fr	De
En		0.586	0.551	0.475
Es	0.629		0.535	0.450
Fr	0.587	0.536		0.435
De	0.571	0.507	0.492	

Promedio: 0.530

Cuadro 10: Puntajes METEOR del modelo entrenado con distintos BPE, utilizando un vocabulario de 30,000 símbolos por idioma y un conjunto de entrenamiento de 9,600,000 oraciones, durante 300,000 épocas

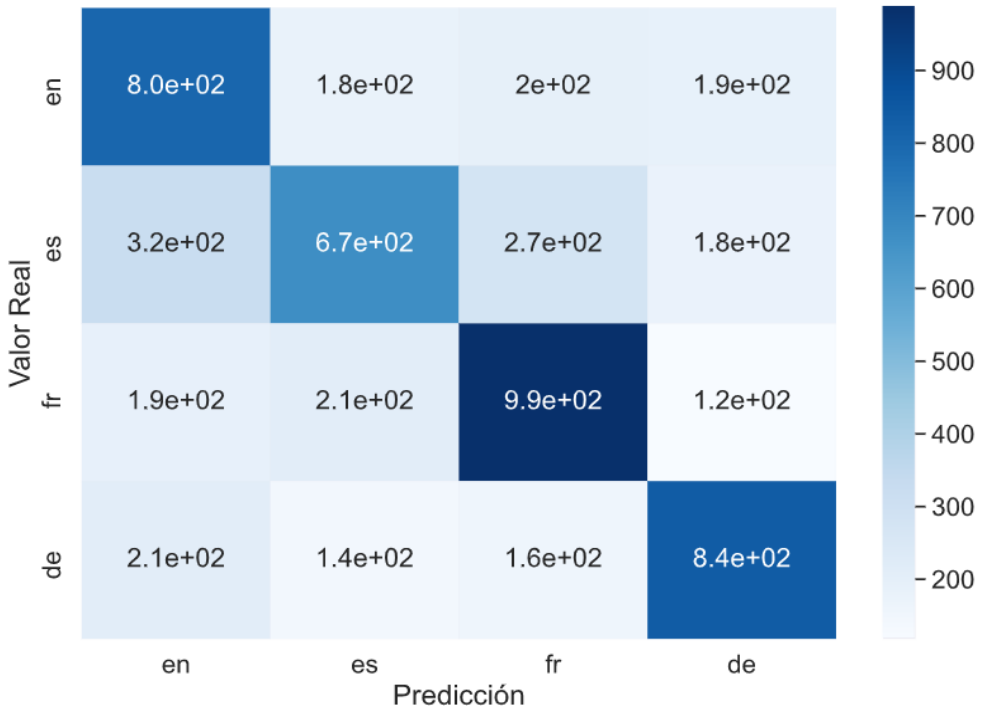
Fuente\Objetivo	En	Es	Fr	De
En		0.590	0.552	0.477
Es	0.632		0.537	0.452
Fr	0.593	0.540		0.440
De	0.574	0.509	0.495	

Promedio: 0.533

A continuación se presenta el rendimiento de los algoritmos de clasificación probados.

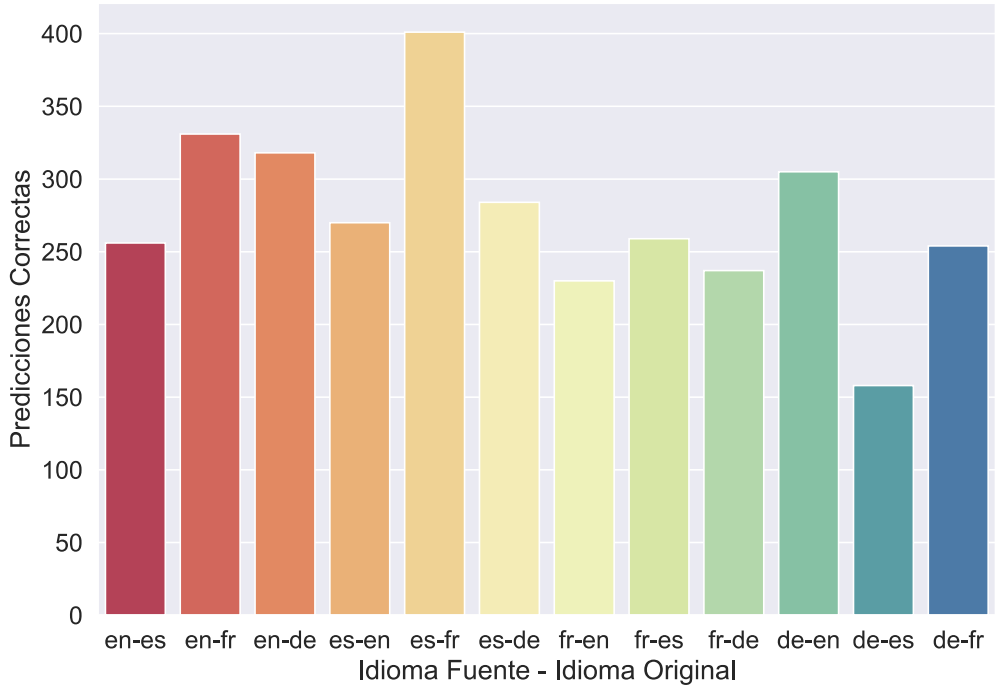
Red neuronal

Figura 7: Matriz de confusión de la Red neuronal sobre el conjunto de datos de prueba



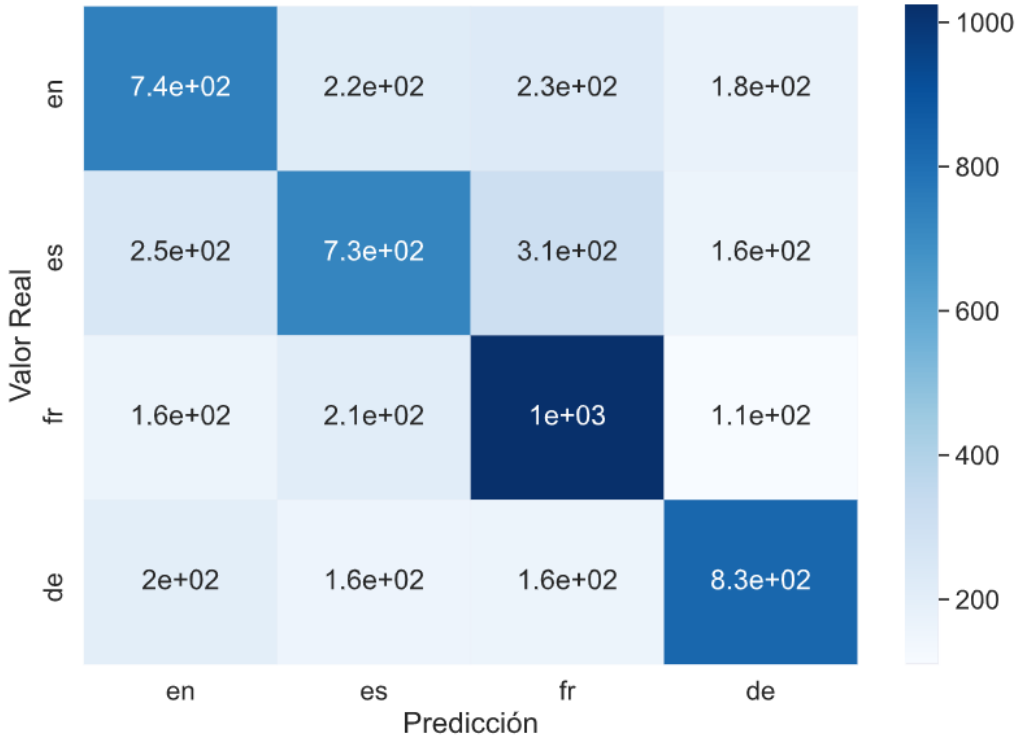
Precisión: 58.21%

Figura 8: Cantidad de predicciones correctas obtenidas por la Red neuronal para cada par de idiomas



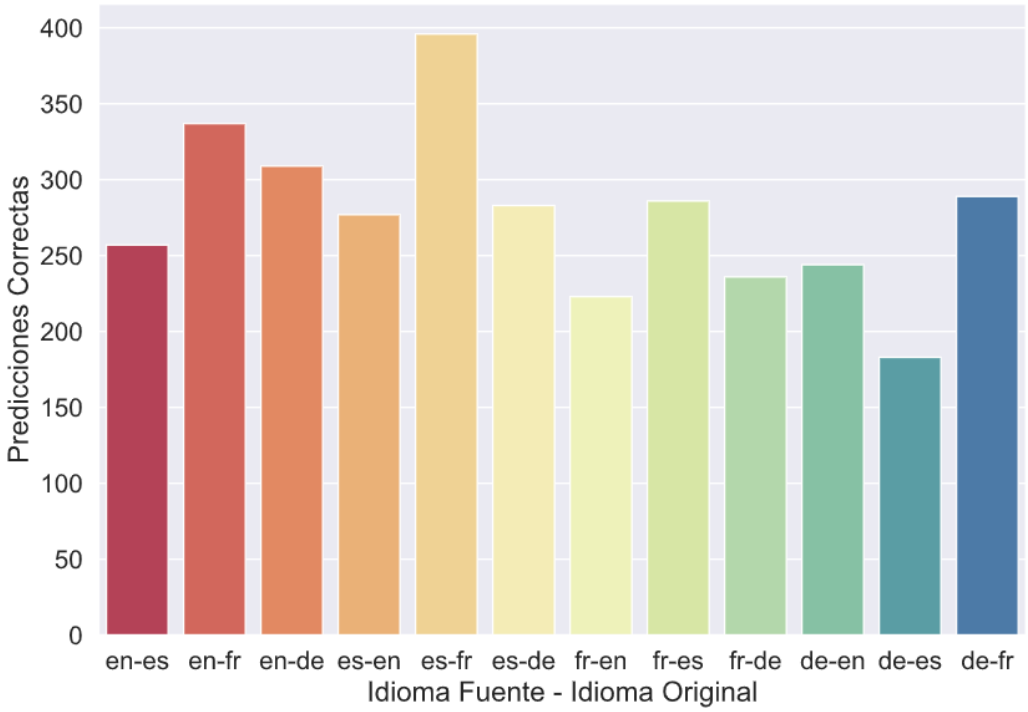
SVM

Figura 9: Matriz de confusión de la SVM sobre el conjunto de datos de prueba



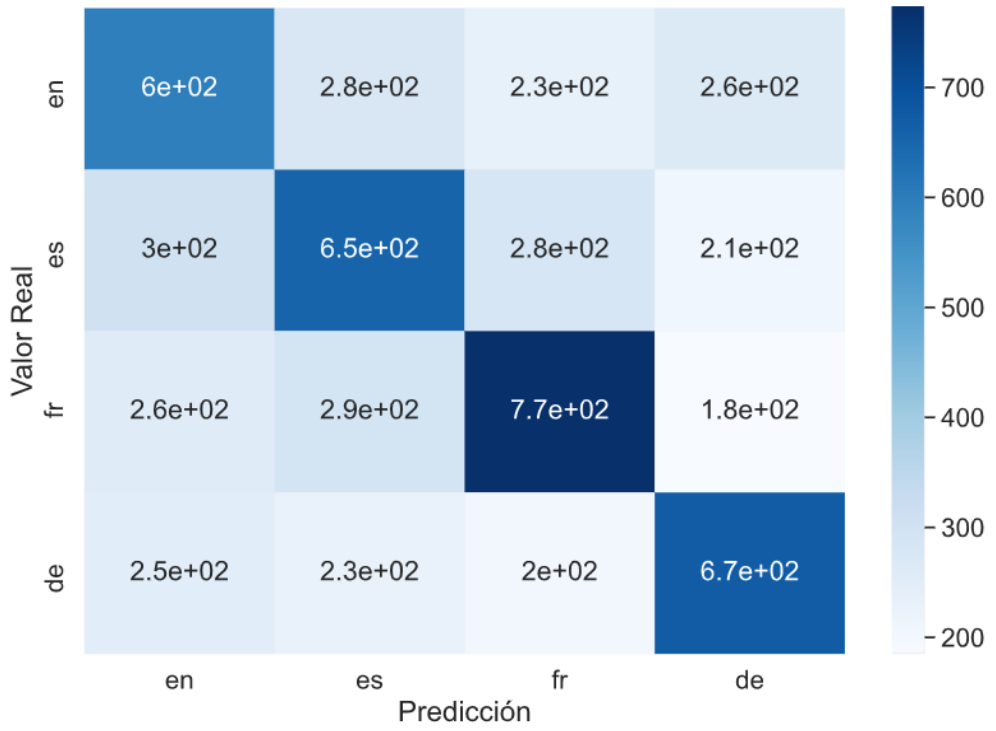
Precisión: 58.51%

Figura 10: Cantidad de predicciones correctas obtenidas por la Máquina de vectores de soporte para cada par de idiomas



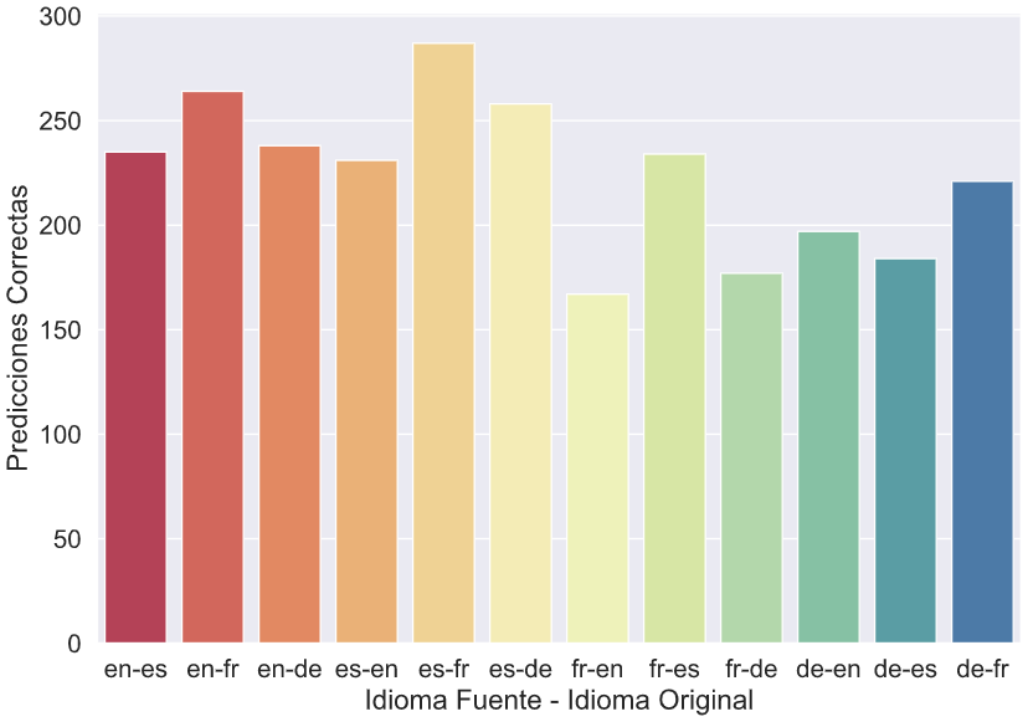
Árbol de decisiones

Figura 11: Matriz de confusión del Árbol de decisiones sobre el conjunto de datos de prueba



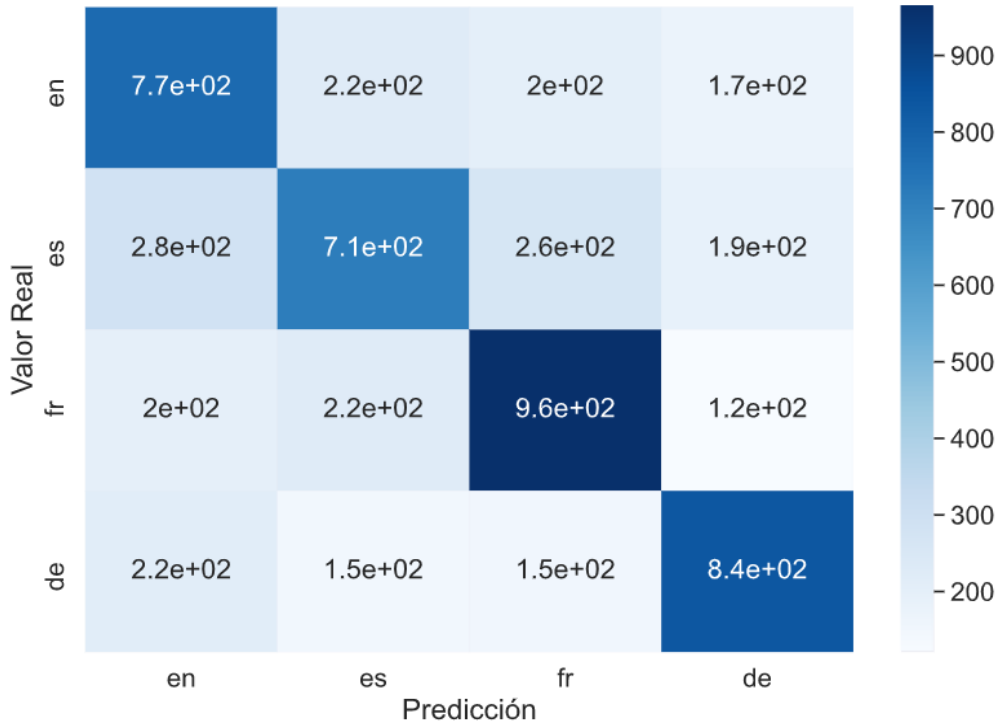
Precisión: 47.46%

Figura 12: Cantidad de predicciones correctas obtenidas por el Árbol de decisiones para cada par de idiomas



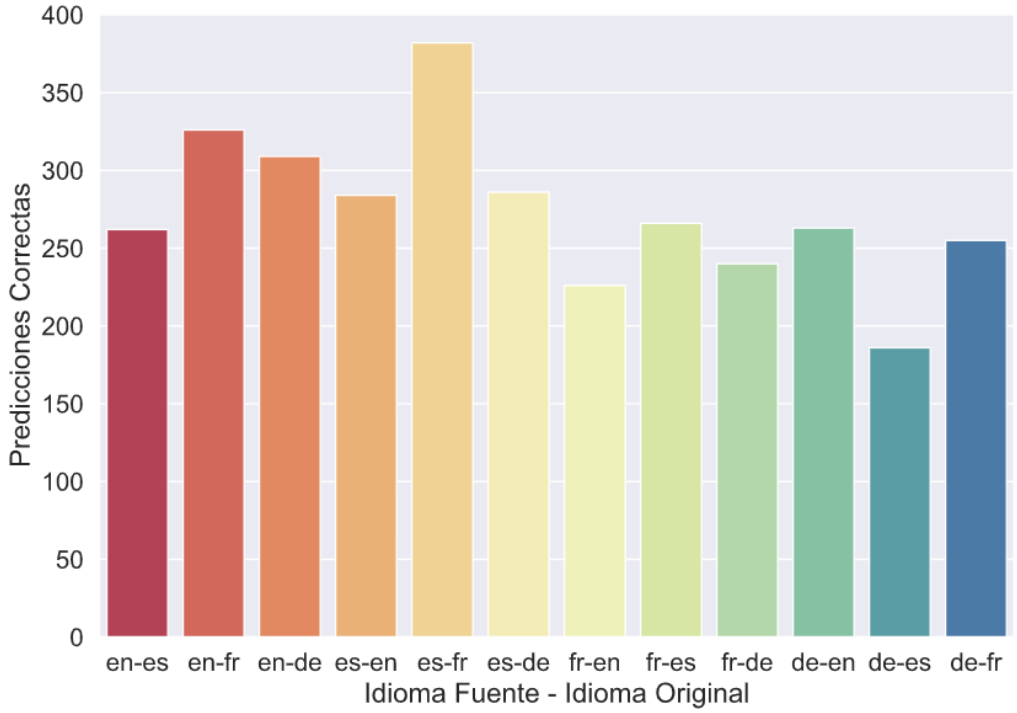
KNN

Figura 13: Matriz de confusión del algoritmo KNN sobre el conjunto de datos de prueba



Precisión: 57.90%

Figura 14: Cantidad de predicciones correctas obtenidas por el algoritmo KNN para cada par de idiomas



VII. ANÁLISIS DE RESULTADOS

La primera mejora obtenida en el rendimiento del motor de traducciones se dió al entrenar un modelo de BPE por separado para cada idioma, como se observan en las diferencias entre el Cuadro 1 y el Cuadro 2 en la sección de resultados. Dicha mejora se muestra tanto en los puntajes BLEU de cada idioma, como en los puntajes METEOR, lo cual se puede observar en las diferencias entre el Cuadro 6 y el Cuadro 7. En el artículo "Training Romance Multi-Way Model" (Senellart, 2018) se muestra la efectividad del acercamiento de utilizar un mismo modelo de BPE para todos los idiomas. Sin embargo, en este artículo se propone utilizar lenguajes de origen latín únicamente. En contraste, en esta investigación se utilizaron dos lenguajes de origen germánico y dos de origen latín. Por esta razón se decidió probar separar el entrenamiento de los modelos de BPE utilizados para la separación de las oraciones en las sub-palabras más frecuentes, lo cual mejoró el promedio de los puntajes BLEU un 4.7% y el promedio de los puntajes METEOR un 5.5%, hecho que se muestra en las diferencias de los promedios sobre los puntajes BLEU del Cuadro 1 y 2, y los puntajes METEOR de los Cuadros 6 y 7.

Al comparar los resultados del Cuadro 2 con los del Cuadro 3 y el Cuadro 7 con el Cuadro 8, se puede ver que, utilizando un vocabulario de 15,000 símbolos por idioma y un conjunto de entrenamiento de 6,000,000 de oraciones, entrenar el modelo por 100,000 épocas adicionales no demostró ser significativo. Es más, los promedios de ambos puntajes disminuyó un 0.3%. Sin embargo, al aumentar el tamaño del conjunto de datos de entrenamiento a 9,600,000 oraciones y el vocabulario de cada idioma a 30,000 símbolos, entrenar el modelo durante 300,000 épocas en lugar de 200,000 mejoró su rendimiento un 0.3% en ambos puntajes, como se puede ver en la comparación de los Cuadros 4 y 5, y en la de los Cuadros 9 y 10 . Esto probablemente se deba a que es más tardado para el mismo algoritmo converger en un valor similar, si se le alimenta con una cantidad mayor de datos y dichos datos tienen una mayor variabilidad.

Este aumento del tamaño del vocabulario y el conjunto de datos se realizó con el objetivo de aumentar el rendimiento en general del modelo, permitiéndole hacer predicciones más variadas. Esta mejora no se puede observar en el puntaje BLEU, pues el mayor promedio de puntajes alcanzado por esta configuración, el cual se muestra en el Cuadro 5, es un 0.1% menor que el mayor promedio alcanzado con la configuración de menor tamaño, el cual se muestra en el Cuadro 3. No obstante, al comparar los resultados del Cuadro 8 y el Cuadro 10, se puede ver una mejora en el promedio de los puntajes METEOR, el cual es 0.4% más alto que el mayor promedio alcanzado por la configuración de menor tamaño.

El mejor puntaje promedio obtenido en la métrica BLEU, utilizando la última configuración de entrenamiento, fue de 0.325. Este se calculó sobre los puntajes que se muestran en el Cuadro 5. El mejor promedio respecto a METEOR fue de 0.533, el cual se calculó con los datos del Cuadro 10. En comparación, en la investigación "Investigating Backtranslation in Neural Machine Translation" utilizan puntajes de 0.2446 en BLEU y 0.2792 en METEOR (Poncelas *et al.*, 2018). En el artículo "Training Romance Multi-Way Model" consiguen un puntaje BLEU de 0.327 para traducciones de español a francés y de

0.329 para traducciones desde francés hacia español (Senellart, 2018). En el artículo original de la arquitectura Transformer, "Attention Is All You Need", logran un puntaje BLEU de 0.284 en traducciones de inglés a alemán y uno de 0.410 en traducciones de inglés a francés (Vaswani *et al.*, 2017).

En cuanto a los resultados de los algoritmos de clasificación, se puede observar que casi todos llegaron a una precisión similar, aproximadamente 58%, con la excepción del Árbol de decisiones, el cual únicamente llegó a 47% de las predicciones acertadas sobre el conjunto de datos de prueba. Algo que muestran todos lo clasificadores en común es su acierto hacia las traducciones de origen francés, lo cual se observa en las Figuras 7, 9, 11 y 13. Para todos, el idioma francés fue el que mostró la mayor cantidad de predicciones correctas. De igual manera, las mismas figuras muestran que el idioma alemán es el segundo idioma con una mayor cantidad de aciertos para todos los algoritmos de clasificación. En cuanto a inglés y español, las traducciones de origen inglés mostraron una cantidad más alta de predicciones correctas que las traducciones de origen español, en las Figuras 7, 9 y 13. El Árbol de decisiones fue el único que mostró el resultado contrario, como se observa en la Figura 11.

En las Figuras 8, 10, 12 y 14, las cuales describen la cantidad de predicciones correctas para cada par de idiomas entre los idiomas de entrada y los que se debían predecir, se observa que los mayores contribuyentes a que francés sea el idioma con mejores predicciones son las traducciones en español. Esto podría indicar que el español es un buen idioma intermedio para mantener la información de traducciones de origen francés. Pero en general, el francés pareciera ser el idioma con más facilidad de detectar, pues el segundo par de idiomas con una mayor cantidad de predicciones en todas las gráficas de barras también involucran al francés como el idioma a predecir. Las traducciones en alemán que deberían predecirse como textos originalmente en francés son las únicas que no muestran un resultado tan alto, pero las traducciones en alemán muestran resultados bajos en general. Esto se probablemente se deba a que es el idioma con los puntajes más bajos, para el modelo del motor de traducciones. Sin embargo, no se encontró ninguna relación directa entre dichos puntajes y las cantidades de predicciones alcanzadas, para cada par de idiomas. Un ejemplo de esto son las traducciones en alemán que deberían predecirse como textos originalmente en español. Estas mostraron la menor cantidad de predicciones correctas en tres de los clasificadores, hecho que muestran las Figuras 8, 10 y 14. Sin embargo, el Cuadro 10 expone que los puntajes alcanzados por el motor de traducciones para dicho par no es el más bajo, en ninguna de las direcciones. Adicionalmente, el mismo cuadro enseña que ninguno de los dos puntajes entre los idiomas español y francés sea el más alto.

En el artículo "Detecting Machine-Translated Text using Back Translation" (Nguyen-Son *et al.*, 2019), también se muestran mejores resultados para las pruebas realizadas con francés como idioma intermedio. A pesar de que el objetivo de esta investigación fue distinto, el método utilizado es esencialmente el mismo. Esto podría indicar que la eficacia de traducir entre un idioma a otro es un factor que influencia altamente la efectividad del método.

VIII. CONCLUSIONES

- Se logró desarrollar un modelo de Aprendizaje de máquina capaz de traducir textos entre los idiomas inglés, alemán, español y francés utilizando una arquitectura de tipo Transformer.

- Se entrenó dicho modelo hasta alcanzar un puntaje promedio de 0.325 en la métrica de evaluación automática de traducciones BLEU y un puntaje promedio de 0.533 en la métrica METEOR.

- Utilizar un modelo de BPE por separado para cada lenguaje puede contribuir a aumentar el rendimiento que pueda alcanzar un motor de traducciones multilingüe, basado en la arquitectura Transformer, si los lenguajes a utilizar son español, inglés, alemán y francés.

- Utilizando el motor de traducciones, se generó un conjunto de datos que contenía las diferencias de las Traducciones-Inversas de una serie de párrafos, con el cual se entrenaron y compararon cuatro algoritmos de clasificación que buscaran categorizar textos según su idioma de origen, dentro de los propuestos en la investigación.

- Con la metodología utilizada y la configuración de los algoritmos mencionados anteriormente, la Máquina de vectores de soporte muestra ser el algoritmo de clasificación con mejores resultados para predecir el idioma original de un texto traducido automáticamente entre los idiomas inglés, alemán, español y francés.

- No se encontró ninguna relación directa entre los puntajes obtenidos por el motor de traducciones y la cantidad de predicciones correctas para cada par de idiomas, utilizando la metodología descrita en esta investigación.

- El método propuesto en esta investigación logra detectar el idioma original de textos traducidos por máquina desde los idiomas español, inglés, alemán o francés hacia alguno de ellos, con una precisión máxima de 58.51%.

IX. RECOMENDACIONES

Se recomienda utilizar un conjunto de idiomas con el mismo origen para tratar de conseguir puntajes más homogéneos para el motor de traducciones. Con esto se podría eliminar la posible influencia que tiene la variación de efectividad del motor, para traducir ciertos pares de lenguajes, sobre los resultados de los clasificadores.

Otra medida que se podría tomar para alcanzar dicha homogeneidad es entrenar un modelo distinto para cada par de idiomas hasta que alcancen aproximadamente los mismos puntajes. Sin embargo, esta es una tarea bastante difícil. En primer lugar, entrenar tantos modelos por separado consume una cantidad considerablemente alta de recursos, tanto de tiempo, como de cómputo. En segundo lugar, cada modelo para los distintos pares de idiomas alcanzaría el puntaje elegido luego de un tiempo distinto de entrenamiento. Por lo que se tendría que estar monitoreando constantemente el progreso de entrenamiento de cada modelo para saber en que punto exacto detenerlo.

También se recomienda utilizar otra métrica para calcular las diferencias entre las Traducciones-Inversas producidas. Es posible que una métrica distinta a BLEU presente una noción más definida y certera las diferencias que se deben encontrar, cambiando la rapidez con la que convergen las Traducciones-Inversas. Esto podría resultar en puntuaciones totalmente distintas para los clasificadores.

X. BIBLIOGRAFÍA

Alammar, J. (2018). *The Illustrated Transformer*. http://jalammar.github.io/illustrated-transformer/

Baig, E. C. (2020). *Translate this: How real-time translation breaks down barriers when you don't speak the language*. USA TODAY. https://www.usatoday.com/story/tech/2020/02/05/translation-tech-solutions-language-barriers-google-translate-interpreter/4596091002/

Banerjee, S., & Lavie, A. (2005). *METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments*. 8.

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *ArXiv:1406.1078 [Cs, Stat]*. http://arxiv.org/abs/1406.1078

Chotard, L. (2017, junio 27). Does automatic translation mean the end for human translators? | TextMaster. *The International Expansion Blog*. https://www.textmaster.com/blog/automatic-translation-vs-human-translators/

Cowen, W. (2017). *Translation technology: Breaking down language barriers—Delta2020 | Financial and Technology Consultancy*. https://delta2020.com/blog/173-translation-technology-breaking-down-language-barriers

Gandhi, R. (2018, julio 5). *Support Vector Machine—Introduction to Machine Learning Algorithms*. Medium. https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47

Gebel, Ł. (2020, agosto 21). *Why We Need Bias in Neural Networks*. Medium. https://towardsdatascience.com/why-we-need-bias-in-neural-networks-db8f7e07cb98

Gupta, P. (2017, noviembre 12). *Decision Trees in Machine Learning*. Medium. https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052

Hao, K. (2018, noviembre 17). *What is machine learning?* MIT Technology Review. https://www.technologyreview.com/2018/11/17/103781/what-is-machine-learning-we-drew-you-another-flowchart/

Hardesty, L. (2017, abril 14). *Explained: Neural networks*. MIT News | Massachusetts Institute of Technology. https://news.mit.edu/2017/explained-neural-networks-deep-learning-0414

Heath, N. (2018, julio 14). *What is machine learning? Everything you need to know*. ZDNet. https://www.zdnet.com/article/what-is-machine-learning-everything-you-need-to-know/

Klein, G. (2019, octubre 9). *CTranslate2: Accelerate inference of OpenNMT models*. OpenNMT Forum. https://forum.opennmt.net/t/ctranslate2-accelerate-inference-of-opennmt-models/3115

Koehn, P. (2005). *Europarl: A Parallel Corpus for Statistical Machine Translation*. 8.
Kurokawa, D., Goutte, C., & Isabelle, P. (2009). *Automatic detection of translated text and its impact on machine translation*. 8.

Nguyen-Son, H.-Q., Thao, T. P., Hidano, S., & Kiyomoto, S. (2019). Detecting Machine-Translated Text using Back Translation. *ArXiv:1910.06558 [Cs]*. http://arxiv.org/abs/1910.06558

Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2001). BLEU: A method for automatic evaluation of machine translation. *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics - ACL '02*, 311. https://doi.org/10.3115/1073083.1073135

Pestov, I. (2018, marzo 12). *A history of machine translation from the Cold War to deep learning*. FreeCodeCamp.Org. https://www.freecodecamp.org/news/a-history-of-machine-translation-from-the-cold-war-to-deep-learning-f1d335ce8b5/

Poncelas, A., Shterionov, D., Way, A., Wenniger, G. M. de B., & Passban, P. (2018). Investigating Backtranslation in Neural Machine Translation. *arXiv:1804.06189 [cs]*. http://arxiv.org/abs/1804.06189

Ruder, S. (2016, enero 19). *An overview of gradient descent optimization algorithms*. Sebastian Ruder. https://ruder.io/optimizing-gradient-descent/

scikit-learn developers. (2020). *1.6. Nearest Neighbors—Scikit-learn 0.23.2 documentation*. https://scikit-learn.org/stable/modules/neighbors.html

Senellart, J. (2018, octubre 9). *Training Romance Multi-Way model*. OpenNMT Forum. https://forum.opennmt.net/t/training-romance-multi-way-model/86

The pandas development team. (2020). *pandas-dev/pandas: Pandas* (latest) [Computer software]. Zenodo. https://doi.org/10.5281/zenodo.3509134

Vashee, K. (2019, diciembre 4). *Understanding MT Quality: BLEU Scores*. SDL. https://blog.sdl.com/blog/understanding-mt-quality-bleu-scores.html

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention Is All You Need. *ArXiv:1706.03762 [Cs]*. http://arxiv.org/abs/1706.03762

Wood, T. (2020, septiembre 27). *Sigmoid Function*. DeepAI. https://deepai.org/machine-learning-glossary-and-terms/sigmoid-function

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Kaiser, Ł., Gouws, S., Kato, Y., Kudo, T., Kazawa, H., … Dean, J. (2016). Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *ArXiv:1609.08144 [Cs]*. http://arxiv.org/abs/1609.08144

Yiu, T. (2019, agosto 4). *Understanding Neural Networks*. Medium. https://towardsdatascience.com/understanding-neural-networks-19020b758230

XI. ANEXOS

Figura 15: Ejemplo del conjunto de datos antes de remover las líneas especiales

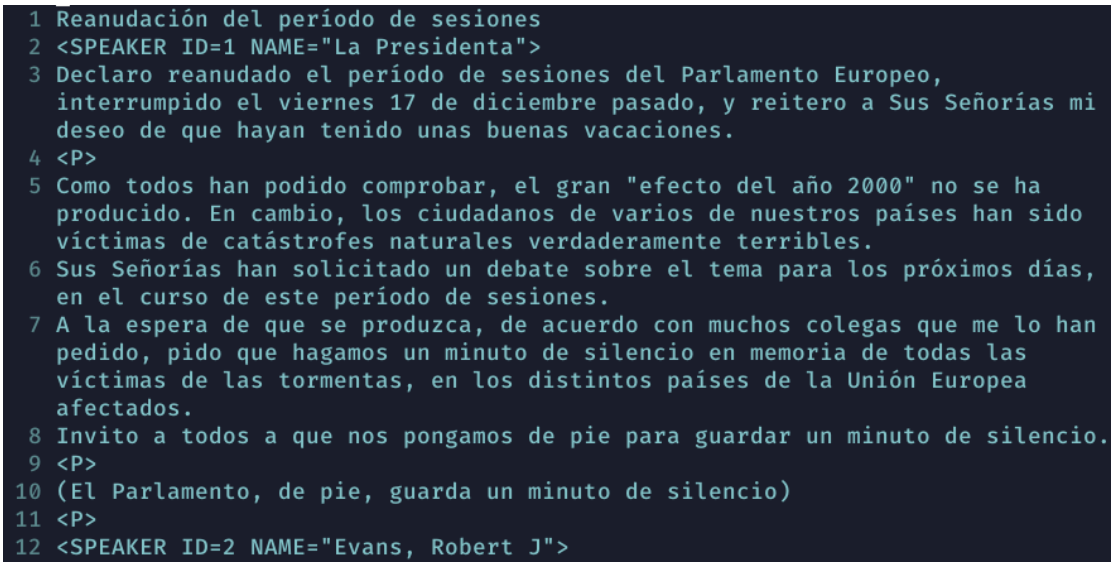


Figura 16: Ejemplo del conjunto de datos después de remover la líneas especiales

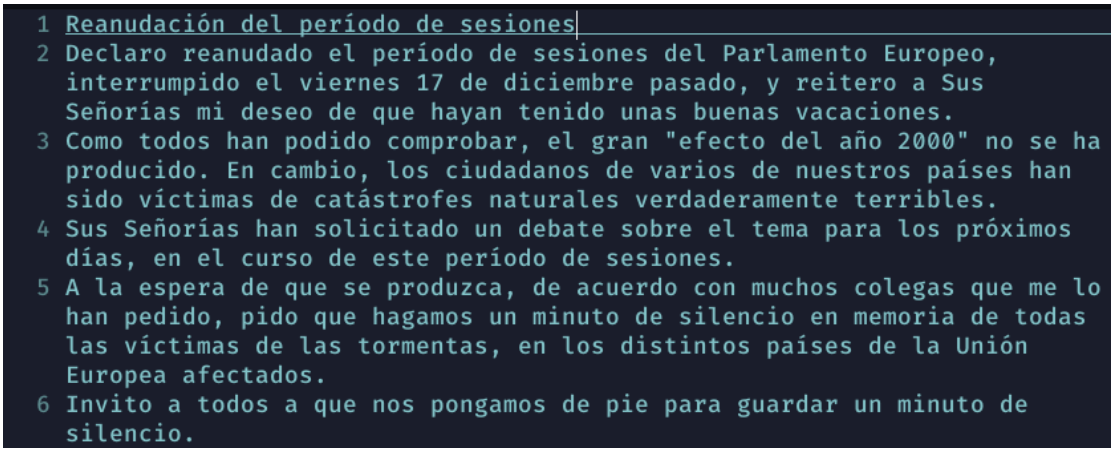
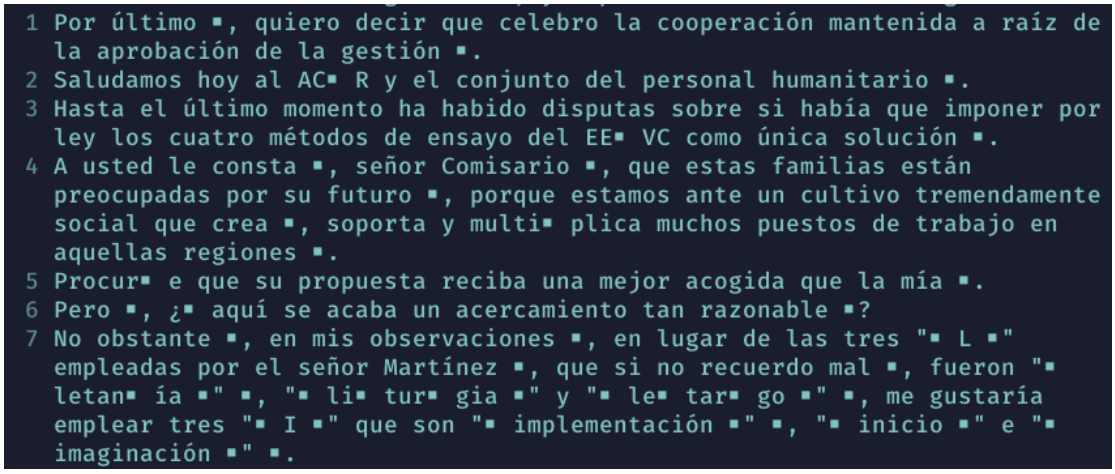
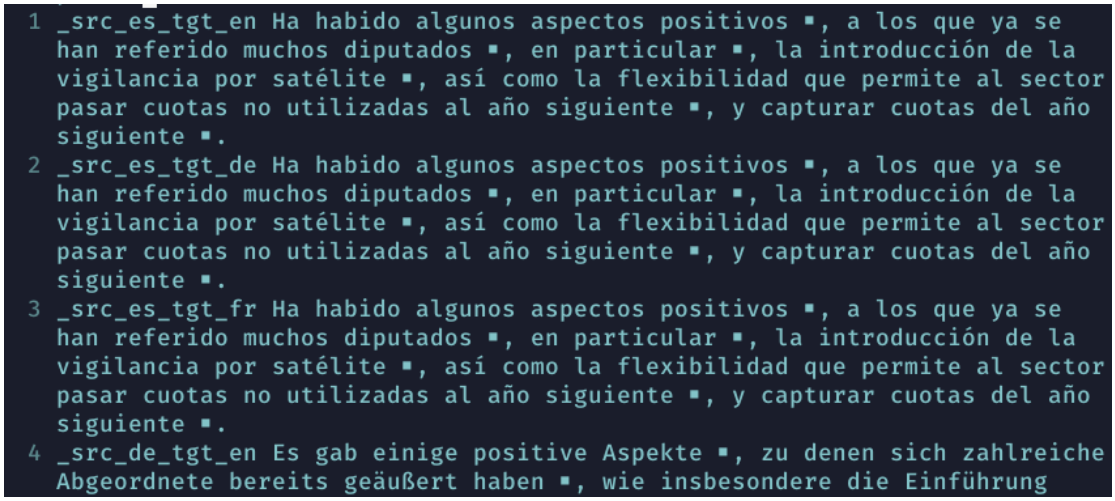


Figura 17: Ejemplo de oraciones separadas en sub-palabras por el modelo de BPE



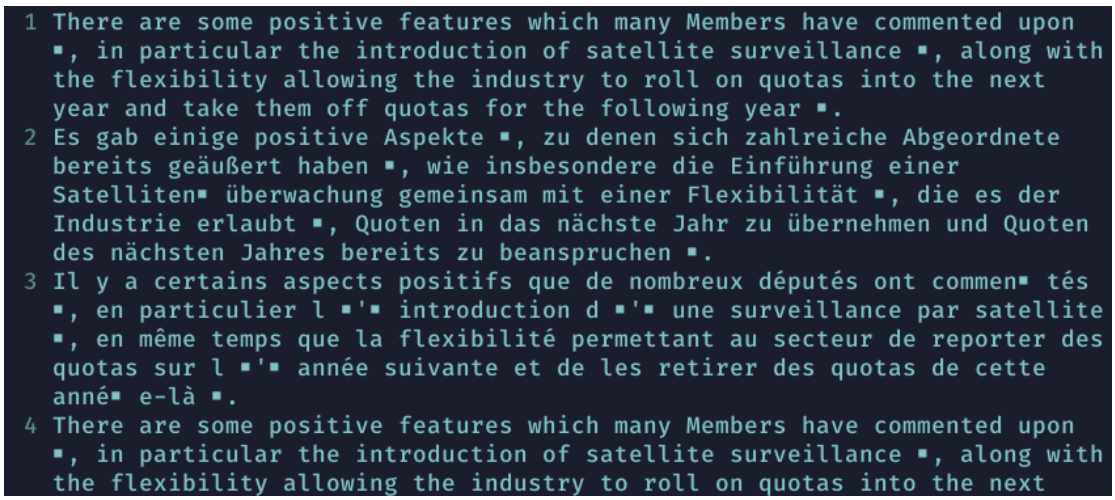
```
1 Por último ▪, quiero decir que celebro la cooperación mantenida a raíz de
  la aprobación de la gestión ▪.
2 Saludamos hoy al AC▪ R y el conjunto del personal humanitario ▪.
3 Hasta el último momento ha habido disputas sobre si había que imponer por
  ley los cuatro métodos de ensayo del EE▪ VC como única solución ▪.
4 A usted le consta ▪, señor Comisario ▪, que estas familias están
  preocupadas por su futuro ▪, porque estamos ante un cultivo tremendamente
  social que crea ▪, soporta y multi▪ plica muchos puestos de trabajo en
  aquellas regiones ▪.
5 Procur▪ e que su propuesta reciba una mejor acogida que la mía ▪.
6 Pero ▪, ¿▪ aquí se acaba un acercamiento tan razonable ▪?
7 No obstante ▪, en mis observaciones ▪, en lugar de las tres "▪ L ▪"
  empleadas por el señor Martínez ▪, que si no recuerdo mal ▪, fueron "▪
  letan▪ ía ▪" ▪, "▪ li▪ tur▪ gia ▪" y "▪ le▪ tar▪ go ▪" ▪, me gustaría
  emplear tres "▪ I ▪" que son "▪ implementación ▪" ▪, "▪ inicio ▪" e "▪
  imaginación ▪" ▪.
```

Figura 18: Ejemplo de oraciones en el archivo final del conjunto de datos fuente para el entrenamiento del motor de traducciones



```
1 _src_es_tgt_en Ha habido algunos aspectos positivos ▪, a los que ya se
  han referido muchos diputados ▪, en particular ▪, la introducción de la
  vigilancia por satélite ▪, así como la flexibilidad que permite al sector
  pasar cuotas no utilizadas al año siguiente ▪, y capturar cuotas del año
  siguiente ▪.
2 _src_es_tgt_de Ha habido algunos aspectos positivos ▪, a los que ya se
  han referido muchos diputados ▪, en particular ▪, la introducción de la
  vigilancia por satélite ▪, así como la flexibilidad que permite al sector
  pasar cuotas no utilizadas al año siguiente ▪, y capturar cuotas del año
  siguiente ▪.
3 _src_es_tgt_fr Ha habido algunos aspectos positivos ▪, a los que ya se
  han referido muchos diputados ▪, en particular ▪, la introducción de la
  vigilancia por satélite ▪, así como la flexibilidad que permite al sector
  pasar cuotas no utilizadas al año siguiente ▪, y capturar cuotas del año
  siguiente ▪.
4 _src_de_tgt_en Es gab einige positive Aspekte ▪, zu denen sich zahlreiche
  Abgeordnete bereits geäußert haben ▪, wie insbesondere die Einführung
```

Figura 19: Ejemplo de oraciones en el archivo final del conjunto de datos objetivo para el entrenamiento del motor de traducciones



```
1 There are some positive features which many Members have commented upon
  ▪, in particular the introduction of satellite surveillance ▪, along with
  the flexibility allowing the industry to roll on quotas into the next
  year and take them off quotas for the following year ▪.
2 Es gab einige positive Aspekte ▪, zu denen sich zahlreiche Abgeordnete
  bereits geäußert haben ▪, wie insbesondere die Einführung einer
  Satelliten▪ überwachung gemeinsam mit einer Flexibilität ▪, die es der
  Industrie erlaubt ▪, Quoten in das nächste Jahr zu übernehmen und Quoten
  des nächsten Jahres bereits zu beanspruchen ▪.
3 Il y a certains aspects positifs que de nombreux députés ont commen▪ tés
  ▪, en particulier l ▪'▪ introduction d ▪'▪ une surveillance par satellite
  ▪, en même temps que la flexibilité permettant au secteur de reporter des
  quotas sur l ▪'▪ année suivante et de les retirer des quotas de cette
  anné▪ e-là ▪.
4 There are some positive features which many Members have commented upon
  ▪, in particular the introduction of satellite surveillance ▪, along with
  the flexibility allowing the industry to roll on quotas into the next
```

Figura 20: Ejemplo de tabla generada durante el preprocesamiento de los datos para los clasificadores

	T0-T1_en	T1-T2_en	T0-T1_es	T1-T2_es	T0-T1_fr	T1-T2_fr	T0-T1_de	T1-T2_de	src	origin	len
0	0.830570	1.000000	0.847027	1.000000	0.0	0.0	0.753269	0.914226	fr	en	141.0
1	0.698240	0.979630	0.783569	0.908302	0.0	0.0	0.704352	0.822784	fr	en	137.0
2	0.793994	0.972784	0.832323	0.963932	0.0	0.0	0.707897	0.891895	fr	en	121.0
3	0.894837	0.969623	0.725450	0.842605	0.0	0.0	0.705540	0.873053	fr	en	117.0
4	0.828548	0.959342	0.758798	0.964595	0.0	0.0	0.750366	0.840804	fr	en	121.0