AFI ESCUELA DE FINANZAS

Máster en Data Science y Big Data

Afi Escuela de Finanzas

Extracción de información en textos usando modelos de lenguaje pre-entrenados.

Autor: David Betancur Sánchez

Tutor: Álvaro Barbero Jiménez, José Manuel Rodríguez

Madrid, Julio, 2019

España

ÍNDICE

Índice de tablas

Índice de figuras

RESUMEN

PLN (Procesado de lenguaje natural) es un campo que ha tomado mucha fuerza en los últimos años. Hoy en día existen algoritmos para extraer información relevante de periódicos, libros y cualquier fuente de texto en general. Además cada vez el alcance es mayor pues es posible trabajar sobre modelos que ya están entrenados y ahorrarse todo el costo computacional y de tiempo que trae esta tarea. Algunos de los temas que se tratan con PLN incluyen la detección de entidades y la extracción de información clave de los textos. Por entidad se refiere a términos que representen objetos de la vida real, tales como personas, lugares y organizaciones que son denotados normalmente por nombres propios. La detección de entidades permite el entendimiento de un texto en particular u optimizar otros procesos y algoritmos. Por otro lado, la extracción de información clave en textos permite la creación de plantillas que resuman el contenido del texto en categorías previamente identificadas y de esta manera poder realizar análisis completos de la información contenida en estos textos. Las representaciones de encoders bidireccionales basados en el "Transformer" (BERT por sus siglas en ingles) es un articulo recientemente publicado por "Google AI Language" que ha demostrado mejorar mucho los resultados del estado del arte sobre ciertas tareas, entre estas el reconocimiento de entidades nombradas (NER). En este trabajo se experimentará sobre esta nueva estrategia para obtener resultados del estado del arte sobre los temas tratados.

ABSTRACT

PLN (Natural Language Processing) is a field that has risen a lot in recent years. Today, there are algorithms that extract relevant information from newspapers, books and any text source in general. In addition, the scope is greater every time because it is possible to work on models that are already pre-trained and save all the computational and time costs that this task brings. Some of the topics covered by PLN include the entity recognition and the extraction of key information from texts. By entity, it refers to terms that represent real-life objects, such as people, places and organizations that are usually denoted by proper names. The entity recognition allows the understanding of a particular text or optimize other processes and algorithms. On the other hand, the extraction of key information in texts allows the creation of templates that summarize the content of the text in previously identified categories and in this way be able to perform complete analysis of the information contained in these texts. The Bidirectional Encoder Representation from Transformers (BERT) is an article recently published by "Google AI Language" which has shown the improvement of the results of the state of the art on certain tasks, among them the Named Entitie Recognition (NER). In this work, we will experiment with this new strategy to obtain results of the state of the art on the topics discussed.

INTRODUCCIÓN

Este trabajo consiste en todo el desarrollo de un sistema que dados textos en lenguaje natural sea capaz de extraer datos estructurados con la información clave de los textos. El objetivo final es la implementación del modelo de lenguaje pre-entrenado "BERT" en un dataset de noticias para extraer información clave como actores de atentados, víctimas, entre otros. A lo largo del trabajo hay varias pruebas con distintos datasets comparando modelos mas sencillos como word embeddings de GloVe con el modelo central que es BERT. Además hay un análisis exploratorio que describe un poco el dataset, viendo los personajes y organizaciones mas involucradas en las distintas noticias. También los países donde predominan estas noticias. Además se observará un poco el funcionamiento de los distintos embeddings utilizados por medio de visualizaciones en 2D utilizando técnicas de reducción de la dimensionalidad como PCA.

Parte I

PLANTEAMIENTO DE LA INVESTIGACIÓN

1.1 Descripción del problema

En los últimos años, el procesado de lenguaje natural ha demostrado ser capaz de resolver problemas que antes no se podía resolver, esto gracias al desarrollo de nuevos algoritmos acompañados de un incremento en la capacidad computacional disponible. Para tener éxito en la resolución de un problema de procesado de lenguaje natural, uno de los desafíos que hay que superar es poder trabajar con la baja cantidad de datos etiquetados disponibles para entrenar los modelos. Como solución a esto, se han creado modelos de representación del lenguaje de propósito general utilizando enormes cantidades de texto mayoritariamente en la web. Estos textos no son anotados pero con ayuda de técnicas como enmascarado o predicción de la siguiente palabra, los modelos aprenden cuales palabras suelen aparecer juntas o aprende a relacionarlas entre si. De esta manera, estos modelos pueden ser manipulados de una manera fina

(fine tunning) para que puedan especializarse en un problema en particular. Esta manipulación fina claramente requiere de poco tiempo y muchos menos datos pues ya esto fue tenido en cuenta para el modelo de propósito general.

1.2 Objetivos

- Desarrollar un sistema que dados textos en lenguaje natural sea capaz de extraer datos estructurados con la información clave de los textos.
- Explorar la utilidad del modelo de lenguaje pre-entrenado BERT en problemas de extracción de información.

1.2.1 Casos de uso

NER (named entity recognition) tiene muchos casos de uso, hasta ahora los mas implementados son los siguientes:

- **Clasificación de noticias por contenido:** permite organizar ciertos artículos y noticias para personalizar el target o cliente al que va dirigida dicha noticia de manera que sea contenido de interés para el cliente y sea una experiencia mucho mas especializada.
- **Servicio al cliente:** Recientemente los bots de servicio al cliente se hacen más y más populares. Para que sean mucho mas útiles, se está implementando el procesado de lenguaje natural para detectar los temas específicos de los que tratan los inconvenientes o dudas del usuario y así poder redirigirlo con el área adecuada de la empresa y poder brindarle un servicio mucho mas especializado.

- **Temas en artículos científicos:** A diario se publican muchos artículos científicos de muchos temas, lo que dificulta buscar artículos relacionados con lo que uno quiere. NER permite clasificar estos artículos en temas específicos aunque ya estos estén acompañados de unas keywords que dan una clasificación inicial.

Parte II

MARCO TEÓRICO

2.1 Antecedentes de estudio

2.1.1 Estudios de ner en la wikipedia

En el articulo "Transforming Wikipedia into Named Entity Training Data" [15] hice ron reconocimiento de entidades nombradas en la wikipedia y después de hacer una validación cruzada, obtuvieron un 89% de F1.

2.1.2 Estudios con el dataset de MUC3 y MUC4

"MUC" son las siglas de "Message Understanding Conference". Fue creado para esta conferencia en la que varios equipos compitieron para obtener los mejores resultados a la hora de etiquetar o rellenar unos formatos de información clave de una serie de noticias de atentados terroristas en Latinoamerica en la década de los 80-90. [8]. Aquí se reportaron los mejores equipos y los mejores resultados fueron de un 40-50 % de recall, 55-65 % de precisión y 10-20 de overgeneration. Overgeneration es una métrica que no es muy utilizada, pero en este caso

resulta útil. Esta métrica la mencionan en uno de los papers de la conferencia de MUC [7] y se define a continuación:

$$OVG = \frac{sporious}{actual}$$

En donde:

- OVG es la métrica de overgeneration.
- sporious es cuantas palabras que no deberían de tener etiqueta el algoritmo detecta como que si la tienen.
- actual es el numero de palabras que tienen la etiqueta

Algunos artículos tratan este dataset buscando detectar las distintas entidades que lo componen. Es un problema complejo por lo que los resultados en general no son muy altos. Por ejemplo, en el artículo "Learning domain-specific information extraction patterns from the Web" [17] se trataron de detectar únicamente los actores de los atentados y las víctimas y se llegó a un 0.5 de F1.

2.1.3 NER con BERT

En cuanto a tareas de reconocimiento de entidades utilizando BERT, aunque sea muy nuevo, hay ya varias implementaciones. Por ejemplo kyzhouhzau [12] tiene un repositorio que realiza esta tarea con el dataset de CoNLL-2003 y obtiene muy buenos resultados:

Otra persona que también obtiene buenos resultados con este dataset (CoNLL-2003) es Kamal Raj [16]:

```
accuracy:  98.15%; precision:  90.61%; recall:  88.85%; FB1:  89.72
              LOC: precision:  91.93%; recall:  91.79%; FB1:  91.86  1387
             MISC: precision:  83.83%; recall:  78.43%; FB1:  81.04  668
              ORG: precision:  87.83%; recall:  85.18%; FB1:  86.48  1191
              PER: precision:  95.19%; recall:  94.83%; FB1:  95.01  1311
```

Figura 1: Resultados NER en CONLL-2003 usando BERT

	precision	recall	f1-score	support
ORG	0.9152	0.9073	0.9113	464
PER	0.9767	0.9692	0.9730	260
LOC	0.9397	0.9263	0.9330	353
MISC	0.8276	0.9014	0.8629	213
avg / total	0.9198	0.9240	0.9217	1290

Figura 2: Resultados NER en CONLL-2003 usando BERT

2.1.4 Técnicas tradicionales

Como antecedentes de estudio también podemos revisar los algoritmos de procesamiento de lenguaje natural que se han utilizado tradicionalmente hasta llegar a lo que se utiliza actualmente.

- **Frecuencia de grupos de caracteres:** Esta técnica consiste en evaluar cuantas veces aparece cada carácter dentro de un documento. Posteriormente se puede utilizar algún algoritmo de machine learning o deep learning para clasificar los textos. Vale aclarar que este algoritmo no serviría para reconocimiento de entidad nombrada (NER) pues ni siquiera evalúa palabras enteras por lo que no podría reconocer si se trata de una entidad nombrada.

- **Tokenizadores:** Esta técnica consiste en la separación por tokens, los cuales no son solo

palabras sino también signos de puntuación y otras unidades básicas del lenguage. Al tokenizar se pueden aplicar distintas técnicas como contar cuantas veces se repite cada token en el documento u otras que veremos a continuación. Posteriormente se pueden aplicar los algoritmos de clasificación.

- **Vectorizadores:** Los vectorizadores lo que hacen es que convierten cada token o documento en un vector de longitud constante que pueden ser introducidos a un modelo de machine learning o deep learning para distintas tareas. Scikit-learn tiene varios vectorizadores que se mencionan a continuación.

- **CountVectorizer:** Este vectorizador convierte una colección de documentos de texto en una matriz de conteos de tokens.

- **TfidfVectorizer:** Este vectorizador lo que hace es que le da mas peso a los tokens que aparecen mucho en el documento actual pero poco en los otros documentos. De esta manera las palabras muy generales del lenguaje van a tener valores pequeños pues aparecen en todos los documentos, y las palabra propias de cada texto van a tener valores mas altos pues aparecerán mucho en el documento propio pero poco en los otros documentos.

2.2 Marco conceptual

2.2.1 Modelos del lenguaje

Según la RAE [1] el lenguaje es una facultad del ser humano de expresarse y comunicarse con los demás a través del sonido articulado o de otros sistemas de signos. Los modelos del lenguaje son entonces la representación de esta facultad en números de tal manera que se puedan realizar

tareas de distintos tipos como clasificación de textos, análisis de sentimiento, traducción, entre otras. En el caso de procesamiento de lenguaje natural, los modelos de lenguaje son una distribución de probabilidad en una secuencia de palabras. Es decir, se busca saber con qué probabilidad una palabra puede ocupar un espacio en una frase determinada. Para estimar eso, lo mas común es observar las palabras o "tokens" anteriores para poder predecir cual palabra podría continuar. A veces resulta mas adecuado utilizar no las palabras completas sino hacer particiones en letras, fonemas, silabas o pares de palabras. A estas particiones se les llama ngrams.

2.2.2 NER

Una entidad nombrada es una colección de fragmentos de texto rígidamente designados que se refieren exactamente a una o varias instancias de concepto idénticas, reales o abstractas. Estas instancias pueden tener varios alias y un nombre puede referirse a diferentes instancias.[21] El reconocimiento de estas entidades nombradas es una tarea de NLP que requiere de distintas técnicas como lo es la utilización de algoritmos de deep learning.

2.2.3 Extracción de textos

Un problema de extracción de información clave de textos se puede entender más bien como un problema de etiquetado de palabras o reconocimiento de entidades nombradas (NER), en el que el modelo lo que haga es recorrer cada palabra del texto y ponerle una etiqueta. Se debe de tener en cuenta también una categoría nula que diga que es una palabra que no es clave con respecto a la información que estamos buscando. A veces una entidad como tal esta compuesta de varias palabras, entonces lo que se hace normalmente es etiquetar de manera distinta las distintas palabras que componen la entidad, poniendo un prefijo que puede ser como a continuación:

tag	meanning
B	beginning
I	inside
L	last
O	outside
U	unit

Tabla 1: tags meanings

Este es el sistema de etiquetado "BILUO" que le da una etiqueta distinta a las partes de la entidad para poder hacer análisis mas específicos.

2.2.4 Glove

GloVe es un algoritmo de aprendizaje no supervisado para obtener representaciones vectoriales de palabras. El entrenamiento se realiza sobre estadísticas globales agregadas de co-ocurrencia palabra-palabra desde un corpus, y las representaciones resultantes muestran interesantes subestructuras lineales del espacio de los vectores que representan las palabras.[2]

En la Figura 3, podemos observar la esencia de los embeddings de glove. Este modelo logró entender la diferencia entre lo masculino y femenino. Aunque no es exacto, podemos ver que las distancias entre lo masculino y femenino se mantiene relativamente constante.

2.2.5 Embeddings contextualizados

Los embeddings contextualizados son embeddings potentes que capturan información latente sintáctica-semántica que va más allá de los word-embeddings estándar [5]. Las diferencias clave son:

- Se entrenan sin una noción explícita de palabras y, por lo tanto, modelan palabras como simplemente secuencias de caracteres. Es decir, se pueden aprender propiedades semán-

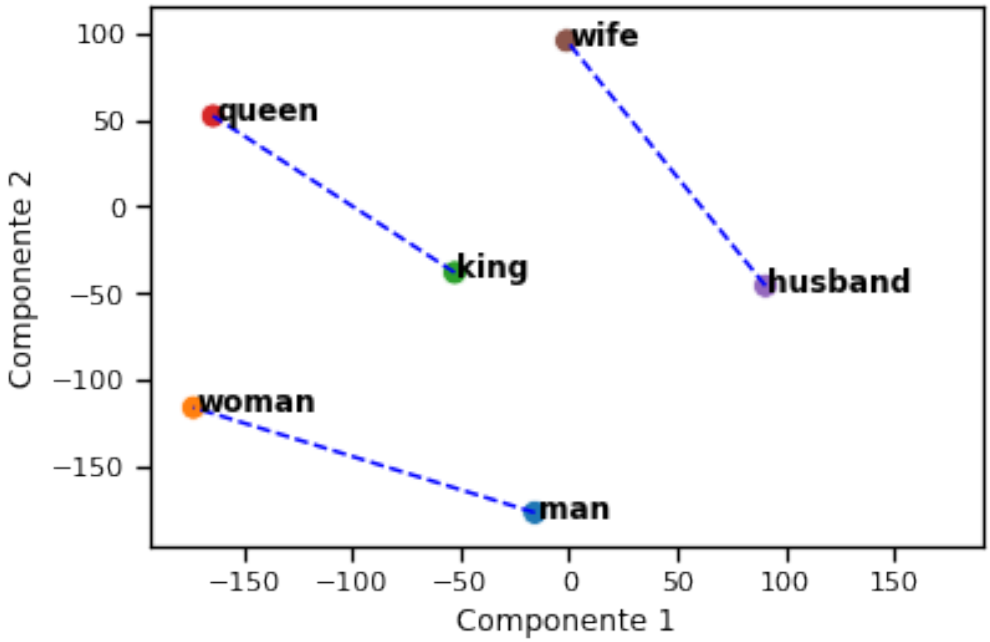

Figura 3: Ejemplo de embeddings de glove

ticas y sintácticas con sub-palabras o secuencias de caracteres en general, y así lo han demostrado algunos estudios recientes [11]

- Están contextualizados por el texto que los rodea, lo que significa que la misma palabra tendrá diferentes valores dependiendo de su uso contextual.

2.2.6 Embeddings contextualizados agregados de Flair

Estos utilizan un método en el que se agregan dinámicamente los embeddings contextualizados de cada cadena de caracteres única que se encuentra en el texto. Luego se usa una operación de agrupación para extraer una representación de palabra "global" de todas las instancias contextualizadas.[3]

2.2.7 Embeddings Forward/Backward

Los embeddings contextualizados de flair permiten la opción de hacerlos forward o backward. Para explicar esto vamos ver la imagen de la Figura 4 de uno de los artículos de zalando research [5]

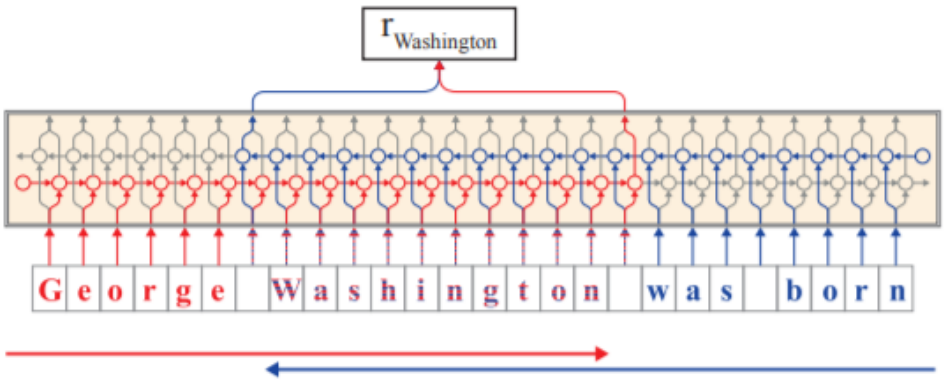

Figura 4: forward-backward embeddings

Como se puede observar en la figura 4 tenemos dos tipos de embeddings. El forward y el backward. Mientras que el forward va recopilando información de caracteres de izquierda a derecha, el backward lo hace de derecha a izquierda. Normalmente, se concatenan ambos embeddings de manera que tengan información de todos los caracteres alrededor de la palabra que se está evaluando.

2.2.8 BERT

BERT[9] fue un articulo publicado por el grupo de investigación de google el cual marcó un momento en la historia del procesamiento de lenguaje natural el cual se puede comparar con lo que es la introducción de ImageNet al mundo de la visión artificial. BERT es un modelo que desde que salió ha sobrepasado varios records en tareas de procesamiento natural y como es

open source, está ayudando a mejorar cada vez los resultados de muchos modelos presentes en la actualidad. Para poder entender el funcionamiento completo de BERT, primero tenemos que entrar a ver el concepto fundamental detrás de este modelo, que es el concepto de Embedding, aunque en las secciones anteriores ya se había hablado un poco de esto, acá se va a repasar con más detalle. Para que una palabra pueda ser procesada por un algoritmo, primero hay que convertirla en una representación numérica, la cual tenga incluidas las propiedades semánticas y sintácticas de esta. Para crear estas representaciones podemos utilizar modelos como los de GloVe (sección 2.2.4) o de Word2Vec. Word2vec es un modelo que consiste en una red neuronal que generalmente tiene 3 capas, una de entrada, una oculta y una de salida. La oculta tiene pocas neuronas. La capa de entrada y de salida pueden diferir pues se tienen 2 métodos, CBOW (Continous bag of words) y Skip-gram. Antes de definir estos dos métodos hay que ver como se obtienen unas representaciones iniciales de las palabras. En este caso se puede utilizar un "one hot encoding" que crea vectores con 1 en la posición que ocupa en el set de palabras y 0 en el resto de posiciones. Este set de palabras es un diccionario en el que aparecen todas las palabras de nuestro corpus. Ahora si, en cuanto a CBOW se quiere predecir una palabra en un punto especifico dado su contexto (palabras alrededor) entonces en este caso la entrada serian los vectores de las palabras alrededor de la que se quiere predecir y la salida sería la representación de la palabra que iría en medio de estas otras. El otro método (Skip-gram) es el opuesto: dado una palabra, se quiere predecir el contexto, entonces en este caso la capa de entrada y de salida se invierten de las de CBOW. En la Figura 5 podemos observar estas arquitecturas.

Lo interesante es que al final el embedding será la matriz de pesos de la capa oculta de la red neuronal. Estos embeddings, si todo se entrenó correctamente, contienen las propiedades sintácticas y semánticas de las palabras que representan, sirviendo entonces para tareas mas

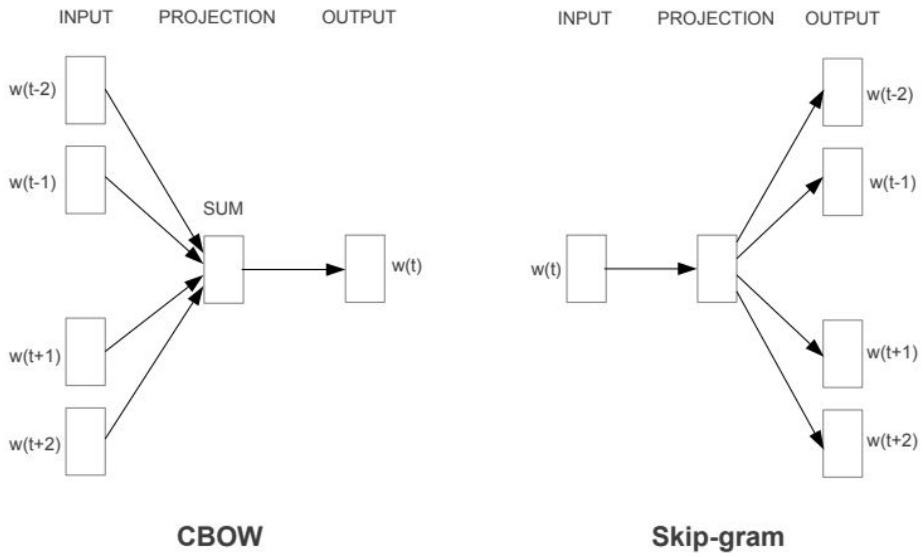

Figura 5: word2vec [19]

complejas dentro del procesamiento del lenguaje natural.

La finalidad de BERT es entregar unos embeddings con varias carácteristicas importantes, pero entre estas la mas importante es que estas representaciones tienen en cuenta el contexto, es decir, la misma palabra en distintos lugares de una frase puede significar cosas distintas, esto BERT lo tiene en cuenta.

Ahora, entrando en detalle, BERT significa "Bidirectional Encoder Representations of Transformers". Vamos a ir explicar algunos conceptos del funcionamiento de BERT:

Bidirectional: Con esto se quiere decir que el modelo evalúa el contexto tanto de las palabras o tokens desde el lado izquierdo como desde el lado derecho. Para esto, BERT utiliza un método llamado "Masked Language Model". Este consiste en reemplazar aleatoriamente palabras por una mascara y luego tratar de predecir cuál palabra iba ahí. Luego, BERT hace otra técnica que consiste en pasar una frase después de otra y tratar de predecir si la segunda frase si va ahí, si sí tiene sentido. Estas segundas frases serán formadas aleatoriamente con la

mitad siendo malas y la mitad buenas.

Transformer: el entrenamiento grueso de BERT consiste en una serie de capas de "The Transformer" que permiten que el modelo aprenda adecuadamente estas representaciones. "The Transformer" [20] fue inventado en 2017 y permite que los modelos tengan atención, es decir, no va aprendiendo representaciones palabra a palabra sino que dada una serie de palabras, el modelo aprende las relaciones entre ellas como se ve en la Figura 6.

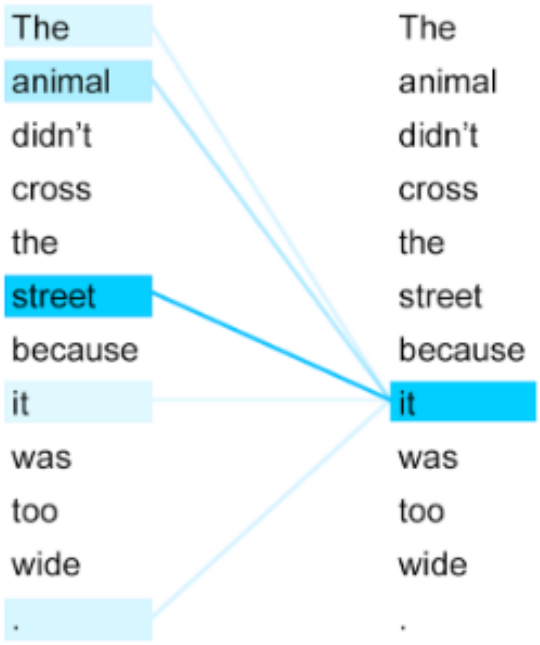

Figura 6: attention [18]

Como se ve ahí, se relaciona la palabra "it" con las palabras de la frase que puedan tener algún tipo de relación con ella, dándole pesos a cada una.

En realidad BERT solo usa la parte del Encoder del Transformer, la cual se ve en la Figura 7. Esto lo hace a partir de aprender los pesos de 3 matrices: Query, Key y Value. Iremos una por una:

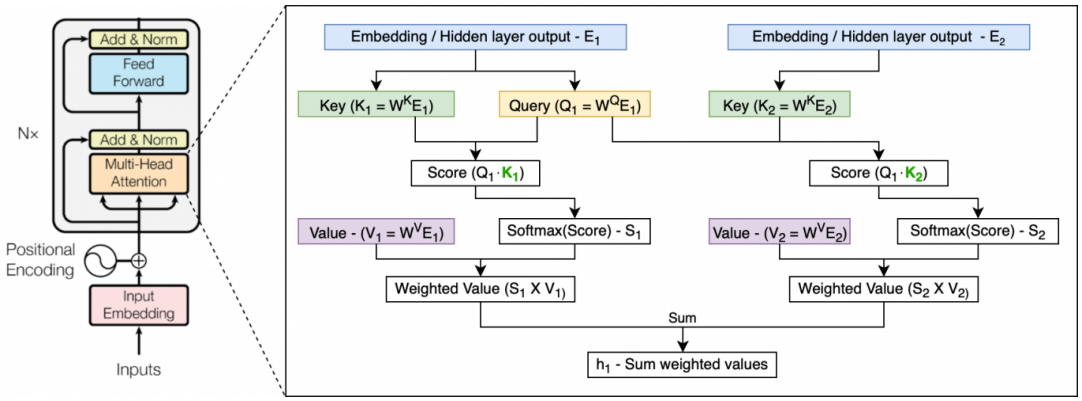

Figura 7: Encoder del Transformer

- **Query**: Lo que se está buscando, es la consulta que se está haciendo, como por ejemplo cual es el sujeto de la oración.
- **Key**: Lo que el token actual puede ofrecerle a los otros tokens.
- **Value**: El valor actual del token.

De nuevo podemos ver la Figura 7 para entender el funcionamiento de esta parte del encoder del Transformer llamada capa de atención. Tenemos entonces los embeddings de todas las palabras de una frase, en este caso hipotéticamente tenemos una frase de 2 palabras. Cada palabra tiene una key que es lo que tiene para ofrecer, y estamos buscando algo en específico, digamos que en este caso estamos buscando cual es el sujeto de la frase. Esto que estamos buscando sería el query que es la misma para ambos embeddings. Tanto la key como el query son vectores de números que salen de multiplicar los embeddings iniciales por una matriz de pesos que se va entrenando a medida que nuevos datos van llegando. Ahora se hace el producto punto entre la key y la query (Nota: esta multiplicación es muy rápida si se tiene GPU por lo que se hace muy óptimo el proceso). Se obtiene entonces un score que resulta ser otro vector al cual se le puede aplicar la función softmax para obtener probabilidades entre 0 y 1. Teniendo

estas probabilidades hacemos una multiplicación con el vector values los cuales son los valores de todos los tokens por lo que se obtiene al final un vector que representa el peso de ese token para esa consulta que estamos haciendo. Esto se hace también con el resto de palabras y así se obtiene la suma de todos estos pesos los cuales indican que tan probable es que cada token sea el sujeto en este caso.

En cuanto al positional encoding es añadir una vector extra que sigue un patrón que el modelo aprende el cual le da el carácter posicional que le hace falta a la arquitectura. Este hace uso de funciones de seno y coseno para crear los valores de los vectores.

"The Transformer" ha sido muy útil para tareas como traducción de texto. Se puede ver cómo en la siguiente animación de google: `https://davidbetancur8.github.io/data_science_tfm/extras/Transformer.gif`.

Finalmente, BERT es una red neuronal que la componen una serie de capas de "Transformers", que guarda los pesos de los datos que se ha entrenado y permite ajustarse a tareas más específicas. Ahora, como se observa en la Figura 8, BERT fue entrenado de una manera no supervisada con enormes cantidades de texto (BooksCorpus [22] y Wikipedia) y de esta manera luego puede ser usado para tareas específicas de una manera supervisada. En nuestro caso esta tarea será NER. Actualmente BERT tiene distintos modelos como cased y uncased (todo en minúsculas o combinado), también tiene en ingles y en multilenguaje y uno muy interesante llamado BIOBERT [13] que es para text minning biomédico.

2.2.9 Visualizaciones embeddings

Con ayuda de la técnica de PCA, se realizaron algunos gráficos dinámicos que permiten ver la diferencia entre el funcionamiento de los embeddings de GloVe y BERT. Mientras que en GloVe aparece cada palabra una vez sin importar el contexto, en BERT cada palabra aparece en varias

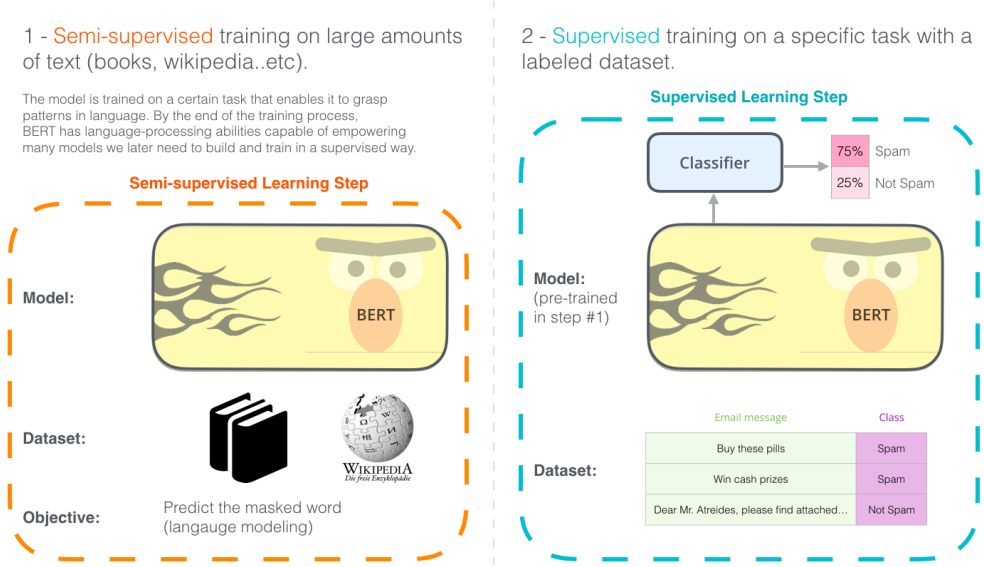

Figura 8: BERT supervisado y no supervisado [4]

ocasiones siendo cada punto una palabra con contexto distinto. Para esto, se utilizó un texto acerca de los futbolistas Lionel Messi y Cristiano Ronaldo, se inicializaron los embeddings correspondientes y se sacaron los vectores para cada palabra. Posteriormente, como estos vectores son de alta dimensionalidad, se redujo a los 3 componentes principales con ayuda de PCA de scikit-learn. Estas 3 componentes principales se gaficaron en un scatter 3D de plotly y se puede ver la versión web en: `https://davidbetancur8.github.io/embedding_viz/`

En estos gráficos dinámicos cada punto representa una palabra, aunque se puede observar que al poner el ratón sobre el punto aparecen más de una. La del medio que aparece mas grande y en negrita es la que fue representada, y las otras son simplemente para identificar donde está ubicada contextualmente esta palabra. Es interesante ver rápidamente que en los embeddings de GloVe hay menos puntos. Esto es porque precisamente estos no son contextualizados entonces si una palabra aparece en distintos lugares de la frase con distintos significados, igualmente se verán representados todos en el mismo punto del espacio n-dimensional de los embeddings. En

el caso de los embeddings de BERT, no es así. Ahora se puede observar que hay varias palabras repetidas en el espacio, esto porque el modelo decidió que estaban en contextos distintos. En las Figuras 9 y 10 podemos observar un gráfico estático muy sencillo para mostrar la diferencia entre los embeddings de GloVe y los de BERT. Este gráfico fue hecho calculando las dos primeras componentes principales y graficandolas en el eje x e y. Los colores fueron asignados dependiendo de si algún embedding se repetía. Mientras que en el caso de los embeddings de GloVe se ven muchos puntos repetidos, en BERT no se ve ninguno. Esto es precisamente porque los embeddings de GloVe son un diccionario en el que cada palabra tiene un embedding, no importa los distintos significados que esta palabra pueda tener. En el caso de BERT, cada embedding es calculado pasando por una red neuronal y depende del contexto, entonces para que el embedding salga repetido no solo tiene que estar repetida la palabra sino también todas aquellas que le rodean.

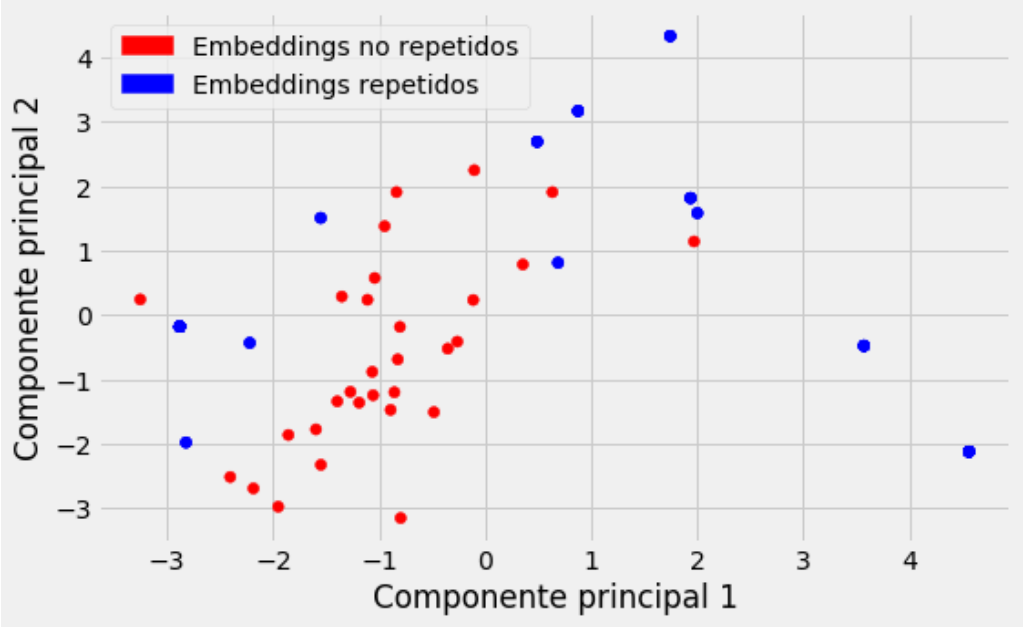

Figura 9: Repeticiones en embeddings de GloVe

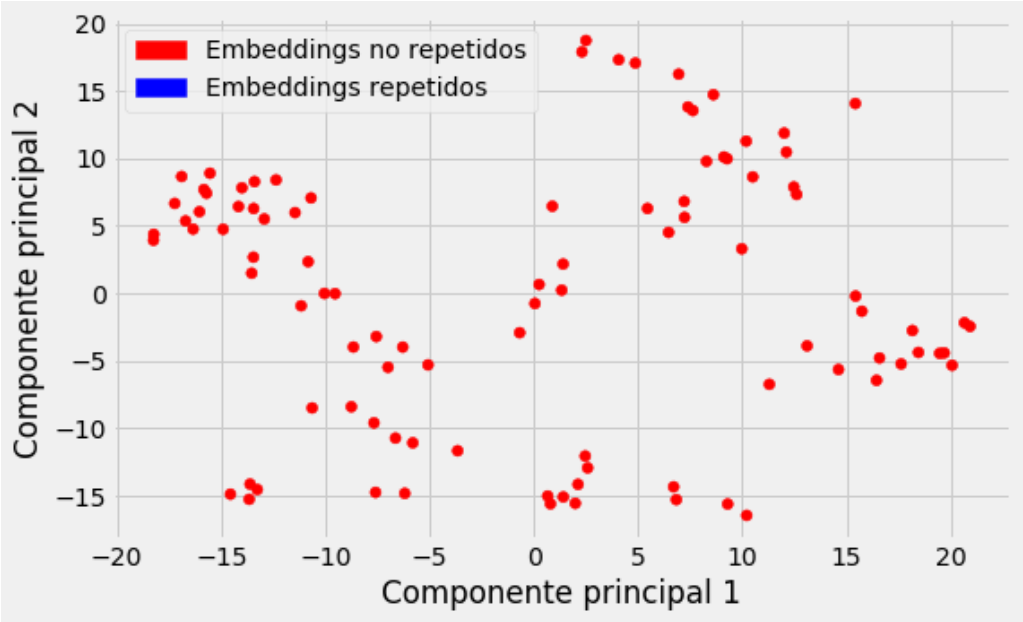

Figura 10: Repeticiones en embeddings de BERT

Parte III

ANÁLISIS EXPLORATORIO

3.1 Descripción de los datos

3.1.1 Wikiner

Se comenzó por hacer un análisis exploratorio de las palabras mas comunes en el dataset. Para esto se utilizó la librería de Wordcloud. El resultado se puede observar en la Figura 11. Las palabras mas comunes son de esperar pues son palabras que se utilizan mucho normalmente. Es interesante por ejemplo ver en la parte superior izquierda que aparece "United States" aunque ningún otro país aparezca. Eso demuestra el peso que tienen sobre internet pues se les menciona mucho.

3.1.2 MUC3 y MUC4

Una vez más se utilizó la librería de Wordcloud para ver las palabras mas comunes en el dataset de MUC3 y MUC4. El resultado se puede observar en la Figura 12. Las palabras mas comunes eran de esperarse para un dataset de noticias de atentados con fines políticos. Por esto se ven

Figura 11: Palabras mas comunes en el dataset de Wikiner

palabras como gobierno, país, fuerza armada, etc.

En la Figura 13, podemos observar las ubicaciones de las noticias en el mapa. Podemos observar que muchas de las noticias están en latinoamerica, y normalmente no hay muchas por lo que aparece casi todo en amarillo claro. Pero en El Salvador hay muchas noticias por lo que se pinta de verde oscuro. Se puede observar con más claridad el mapa en `https://davidbetancur8.github.io/data_science_tfm/exploration/mapa.html`

Figura 12: Palabras mas comunes en el dataset de MUC3 y MUC4

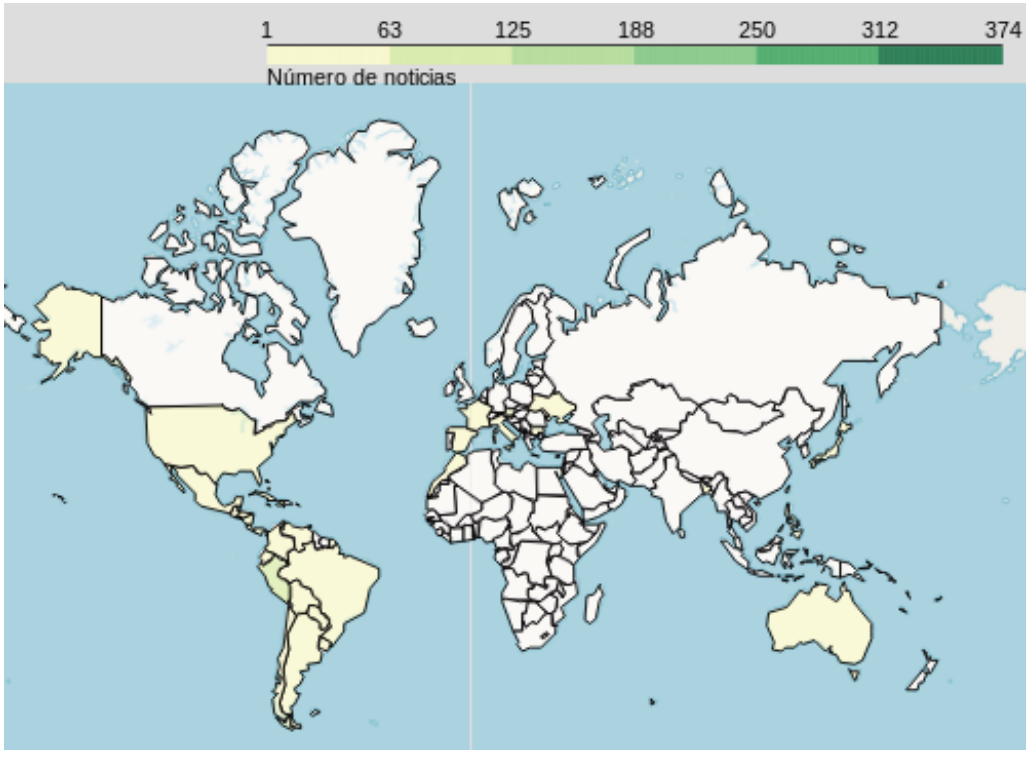

Figura 13: Número de noticias por país

En la Figura 14 se pueden observar las fechas en las que ocurrieron los sucesos de las noticias. No se logra observar muy bien pero hay un dato atípico que aparece casi en 2011 lo cual no tiene sentido pues el dataset fue creado en el 2001. Por lo tanto se procedió a eliminar esta fecha y el nuevo gráfico se observa en la Figura y 15 con más claridad. Se ve que muchas noticias ocurrieron en Diciembre de 1989. En estas fechas ocurrieron una serie de intervencionismos de EEUU en algunos países latinoamericanos, tal vez algunas noticias tengan que ver con esto. Vemos también que cada día salían menos de 20 noticias pero dentro de cada noticia hay muchas palabras por lo que con esto es suficiente para entrenar los modelos.

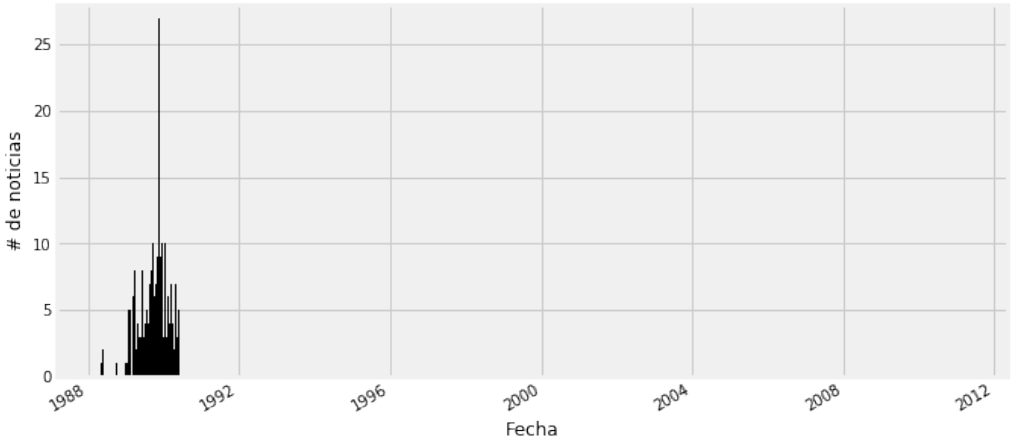

Figura 14: Fechas de las noticias

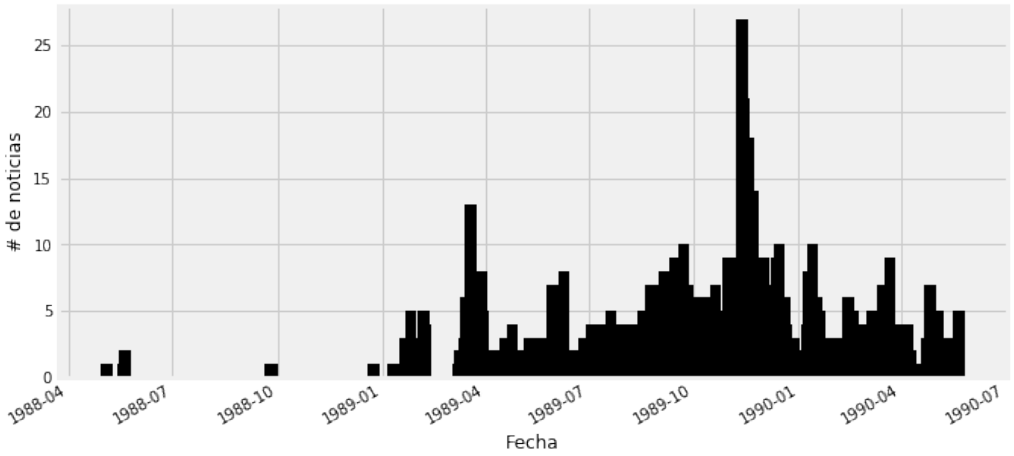

Figura 15: Fechas de las noticias sin outlier del 2011

Ahora se va a analizar un poco las distribuciones de las variables que se van a predecir.

En cuanto a los targets físicos, como se ve en la Figura 16, se puede observar que lo que mas se atacaban eran los buses y restaurantes. Por ejemplo en Colombia entre 1980 y 2000 hubo una oleada de violencia a raíz del narcotráfico hasta el punto de poner explosivos en lugares públicos para asesinar personajes importantes en su lucha contra estos carteles.

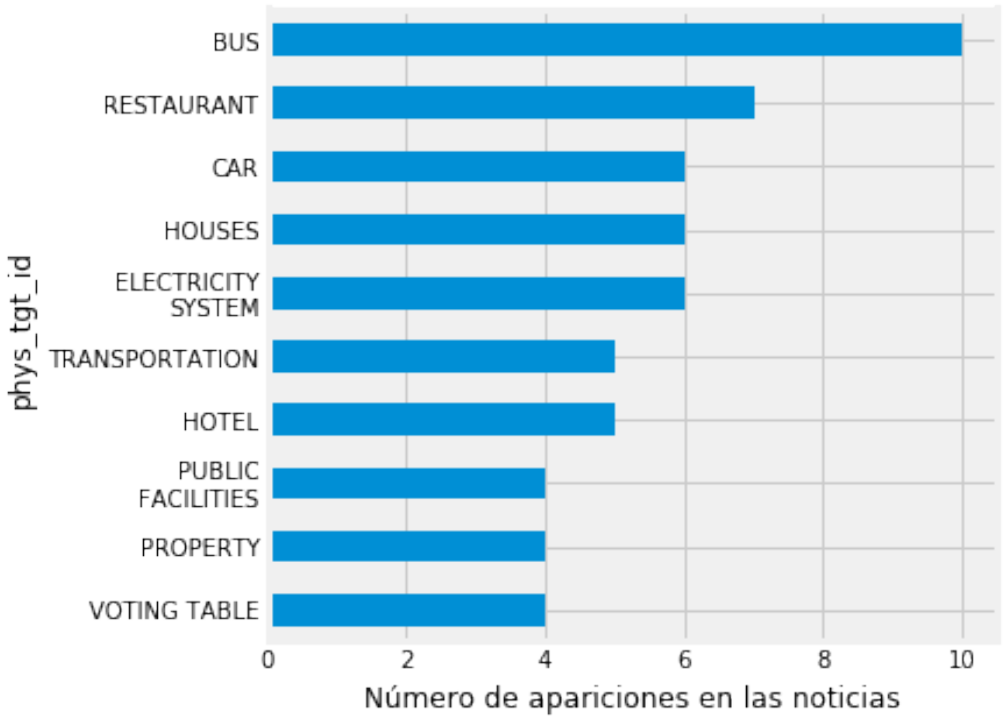

Figura 16: Objetivo físico

En cuanto a las organizaciones autoras de los atentados vemos que predomina el FMLN que es el frente de liberación nacional de Farabundo Mart, el cual operaba en EL SALVADOR. Luego sigue el "Shining path" que operaba en Perú durante muchos años y el Ejercito de Liberación Nacional (ELN) que operaba y aún opera en Colombia.

Ahora, en cuanto a los actores de los atentados a nivel mas individual, los que aparecen más en las noticias son guerrilleros, terroristas y rebeldes.

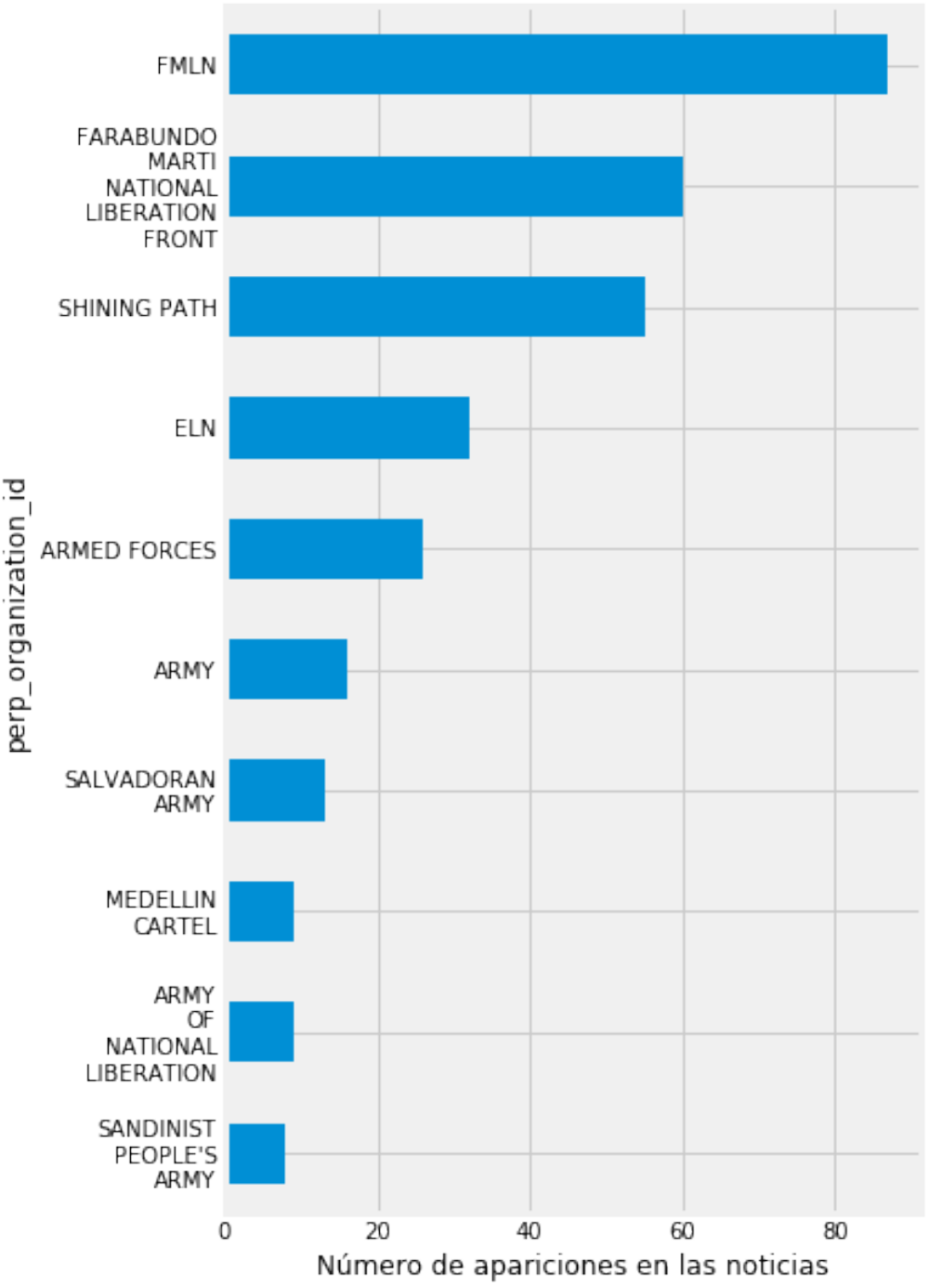

Figura 17: Organización autora del ataque

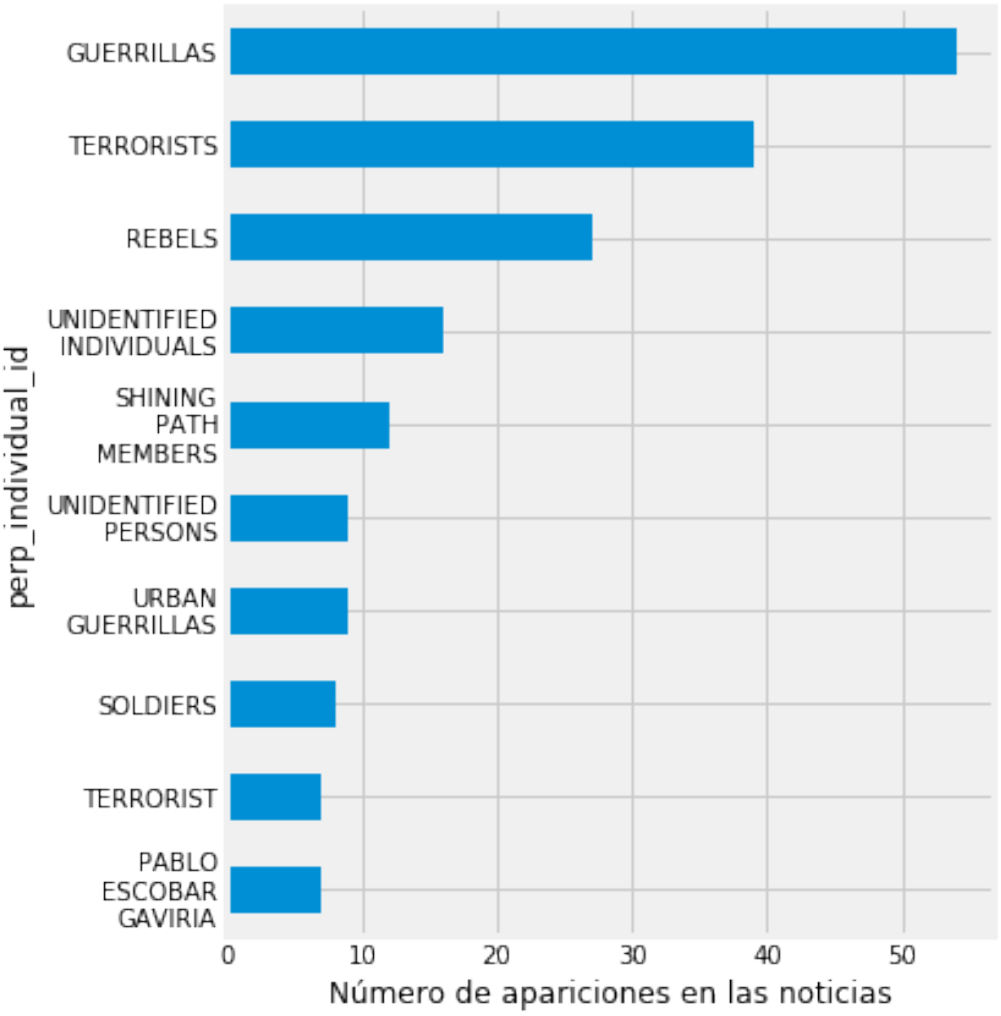

Figura 18: Autor del acto

Entre los elementos utilizados para los ataques, lo que mas se usó fueron mas que todo bombas.

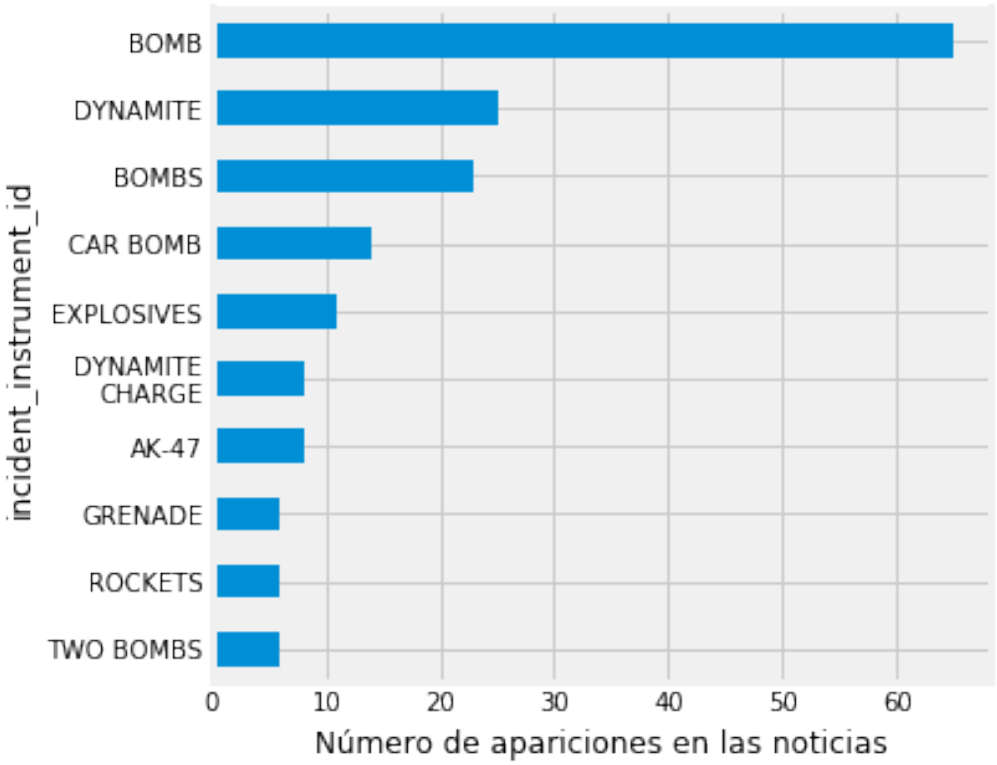

Figura 19: Instrumento

El personaje mas atacado fue Luis Carlos Galán que fue un reconocido periodista y político en Colombia. Hubo múltiples intentos de asesinato en su contra hasta que finalmente lo asesinaron En Agosto de 1989.

En cuanto a las descripciones de algunas víctimas a nivel general los mas afectados fueron los Jesuitas.

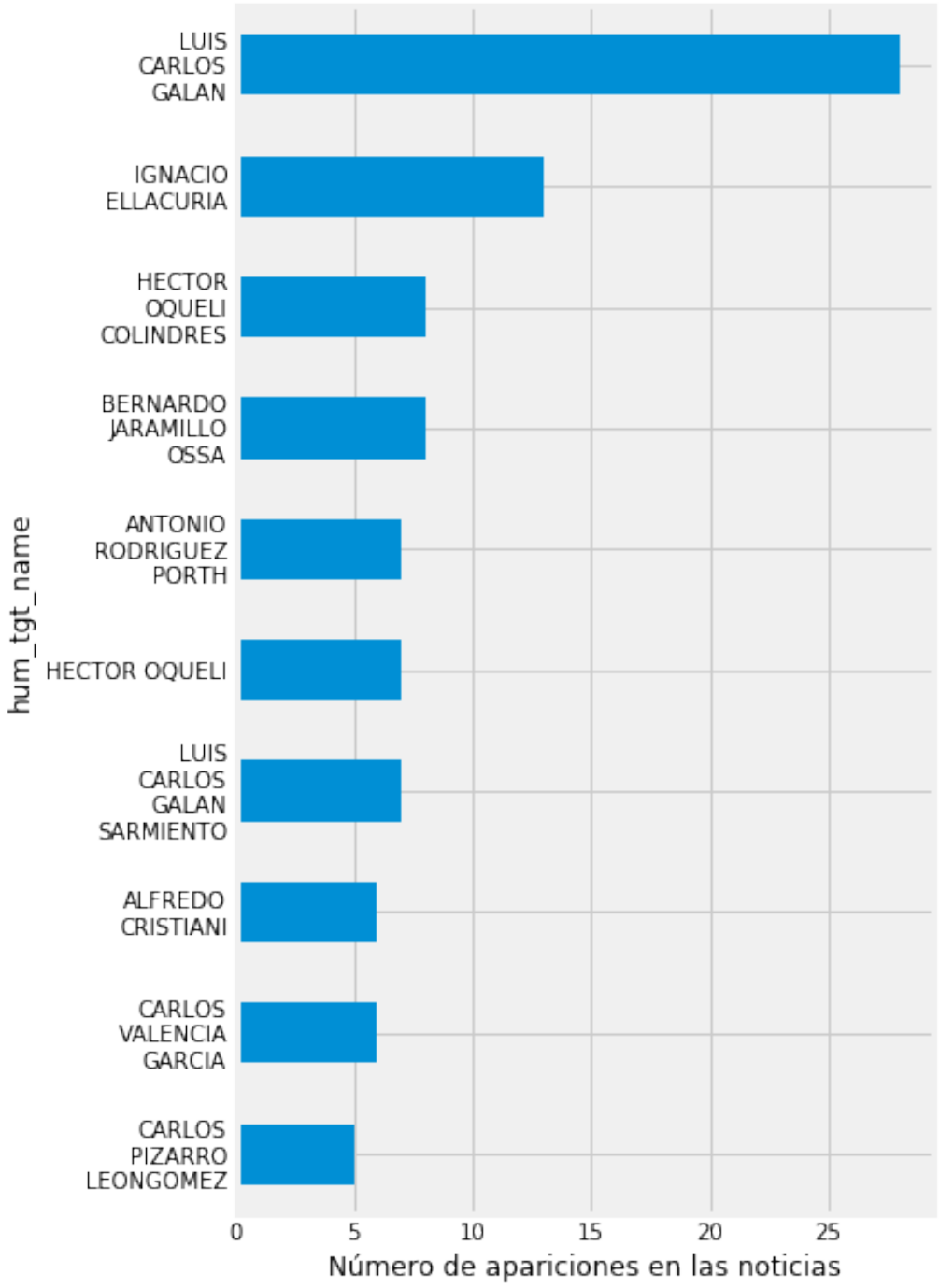

Figura 20: Nombre del humano objetivo

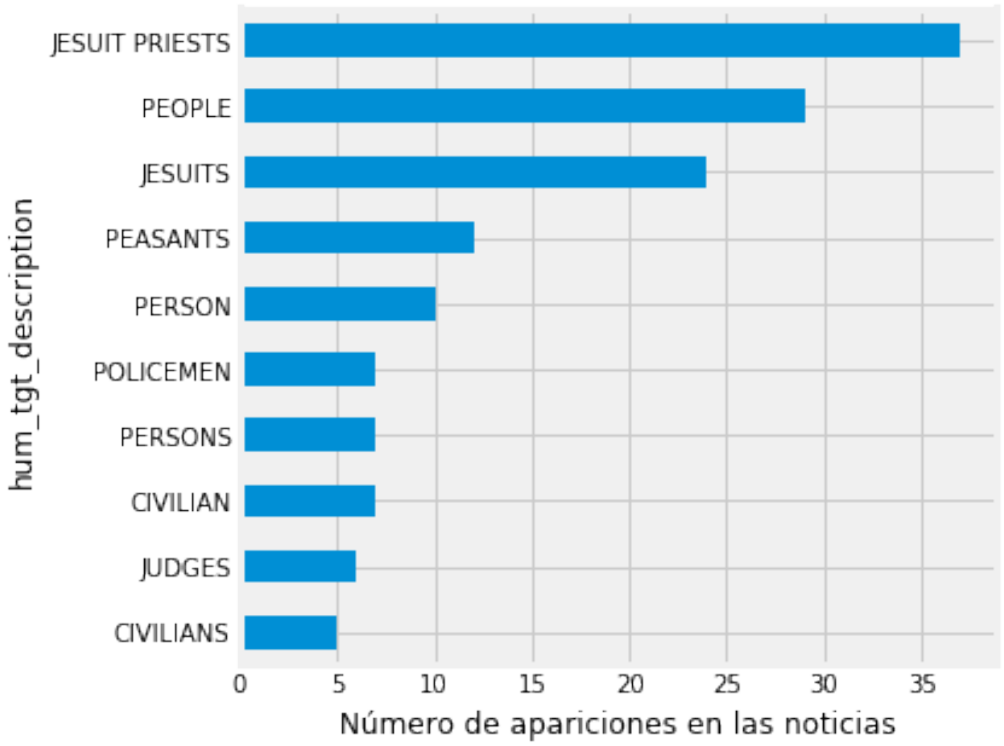

Figura 21: Descripción del objetivo

3.2 Preprocesado de la información

El dataset de wikiner y wnut ya venían incluidos en la librería flair la cual es una librería de procesamiento de lenguaje natural que tiene varias utilidades para muchas de las tareas que se realizaron en este trabajo. Estos datasets dentro de la librería ya vienen con el formato para que se puedan leer los datos correctamente por lo que no fue necesario un preprocesado en esta sección. Aún así, en el dataset de MUC3 y MUC4 si que los datos venían sin el formato que acepta flair y fue necesario realizar una ETL para organizarlos de manera que queden de acuerdo al formato que pide la librería flair. Esta ETL siguió los siguientes pasos que se mencionan a continuación:

- **Descarga de datos:** Los datos se descargaron y descomprimieron. Estos datos están disponibles en `https://www-nlpir.nist.gov/related_projects/muc/muc_data/muc_data_index.html`

- Primero se preprocesaron los textos y la información clave con la ayuda del código existente de "link". Esto devolvía los textos mas limpios y la información clave en una especie de diccionario json que va a facilitar mucho el resto del procesamiento.

- Posteriormente se creó un dataframe en el que en una columna está el texto limpio y luego una columna mas por cada entidad. En estas columnas en cada fila está el diccionario con la información clave, y si no hay información entonces se deja en None.

- A continuación se iteró por cada elemento de la tabla, cambiando las entidades nombradas por los índices de inicio y fin de donde aparecen en el texto.

- Teniendo esta tabla ya se puede utilizar la librería de Spacy que es una librería de nlp con muchas utilidades. En este caso se utilizó la utilidad "biluo tags from offsets" para taggear el corpus.

- Teniendo los datos en este formato ya se puede utilizar la utilidad ColumnCorpus de Flair que crea el corpus exactamente como debe de estar para poder entrenar el modelo.

En la Figura 22 y 23 se puede observar un ejemplo de como era el texto y el template de información clave antes de la ETL.

Luego de la ETL , quedaría algo como en la Figura 24. los puntos que hay en medio es porque son muchas palabras y todas están etiquetadas entonces se resumió para poderlo mostrar brevemente acá.

Este etiquetado es de tipo BILUO que se explicó en la sección 2.2.3.

```
DEV-MUC3-0011 (NOSC)

   LIMA, 9 JAN 90 (EFE) -- [TEXT] AUTHORITIES HAVE REPORTED THAT FORMER
PERUVIAN DEFENSE MINISTER GENERAL ENRIQUE LOPEZ ALBUJAR DIED TODAY IN LIMA
AS A CONSEQUENCE OF A TERRORIST ATTACK.

   LOPEZ ALBUJAR, FORMER ARMY COMMANDER GENERAL AND DEFENSE MINISTER UNTIL
MAY 1989, WAS RIDDLED WITH BULLETS BY THREE YOUNG INDIVIDUALS AS HE WAS
GETTING OUT OF HIS CAR IN AN OPEN PARKING LOT IN A COMMERCIAL CENTER IN THE
RESIDENTIAL NEIGHBORHOOD OF SAN ISIDRO.

   LOPEZ ALBUJAR, 63, WAS DRIVING HIS OWN CAR WITHOUT AN ESCORT.  HE WAS SHOT
EIGHT TIMES IN THE CHEST.  THE FORMER MINISTER WAS RUSHED TO THE AIR FORCE
HOSPITAL WHERE HE DIED.
```

Figura 22: Noticia antes de la ETL

```
%%%
[
  ["message_id", "DEV-MUC3-0011"],
  ["message_template", 1],
  ["incident_instrument_id", null],
  ["perp_individual_id", {"strings": ["THREE YOUNG INDIVIDUALS"], "type": "simple_strings"}],
  ["perp_organization_id", null],
  ["phys_tgt_id", null],
  ["hum_tgt_name", {"strings": ["ENRIQUE LOPEZ ALBUJAR"], "type": "simple_strings"}],
  ["hum_tgt_description", {"strings_lhs": ["FORMER ARMY COMMANDER GENERAL AND DEFENSE MINISTER"],
"strings_rhs": ["ENRIQUE LOPEZ ALBUJAR"], "type": "colon_clause"}]
]
```

Figura 23: Template información antes de la ETL

```
authorities O
have O
reported O
.
.
.
three B-perp_individual_id
young I-perp_individual_id
individuals L-perp_individual_id
.
.
.
```

Figura 24: texto anotado después de la ETL

Parte IV

METODOLOGÍA

4.1 Estructura general

Casi todos los experimentos siguieron la estructura recomendada por Flair, la cual se detalla paso a paso a continuación:

- Cargar las librerías correspondientes
- Cargar el corpus con el formato correcto que se especifica en la sección de preprocesado.
- Especificar el tipo de tarea que se busca, en este caso "ner"
- Crear un diccionario con el corpus
- Inicializar los embeddings que se quieren usar
- Como se trata de una tarea de NER se tiene que inicializar el tagger que es lo que le va a dar una etiqueta a cada palabra. En este tagger va a ir el diccionario, los embeddings

y algunas configuraciones como la arquitectura de la capa densa que va al final de la red neuronal.

- Inicializar el trainner
- Empezar el entrenamiento

Se puede ver un ejemplo mas detallado en la documentación de Flair en `https://github.com/zalandoresearch/flair/blob/master/resources/docs/TUTORIAL_7_TRAINING_A_MODEL.md`

4.2 WIKINER

4.2.1 Benchmark

Para el benchmark se utilizaron embeddings de "GloVe" en el dataset de WIKINER. Las entidades que se buscaron fueron de localización, organización, persona y misceláneo. Como métricas de evaluación se utilizaron recall, precisión, accuracy y F1.

4.2.2 BERT

Los embeddings contextualizados que se van a utilizar en este caso son los embeddings de BERT. La librería flair tiene la incluida la implementación de "huggingface" [10] que parece ser la más utilizada en este momento.

4.3 MUC3 y MUC4

4.3.1 Benchmark

El benchmark para este apartado nuevamente se realizó con los WordEmbeddings de glove.

4.3.2 BERT

Nuevamente se utilizaron los embeddings contextualizados de BERT para esta tarea.

Una de las cosas que ayuda mucho a un sistema de NER es cuando los nombres y organizaciones empiezan con una letra mayúscula, pues los algoritmos detectan esto y le dan un mayor peso para ser detectado como una entidad. Por alguna razón este dataset fue hecho todo en mayúsculas, lo que complica mucho el problema. Por suerte, BERT tiene un modelo entrenado "uncased" lo que significa que todo fue pasado a minúsculas antes de pasar por la red neuronal. Para poder acoplarse a este modelo entonces se volvió a crear el corpus pasando todo a minúsculas.

Para este caso, no fue factible utilizar el mismo procedimiento que con wikiner, pues al parecer la librería Flair aún no está optimizada para datasets nuevos y por esto el tiempo que tomaba entrenar este modelo era supremamente alto (aprox 1 hora por época) y aún así los resultados eran muy malos como se ve en la Figura 28.

Por esta razón se recurrió a otra alternativa, la cual fue utilizar la librería Flair únicamente para calcular los embeddings y guardar estos en un fichero y luego utilizar una red neuronal en keras para entrenar el modelo. De esta manera, la parte de calculo de los embeddings solo se hace una vez lo que optimiza mucho el proceso. Como se puede imaginar, este fichero es muy grande: aproximadamente 34000 registros, cada uno con embeddings de dimensión 3072. Por esto, se tuvo que separar esta información en distintos ficheros (10 de 2 GB cada uno). Se

guardaron en forma de pickle.

Al ser distintos ficheros y tan pesados, la red neuronal de Keras fue entrenada con un fit_generator, el cual, como dice su nombre, se alimenta de un generador. Este generador iteraba por cada linea de cada fichero y guardaba cada lista de features con su respectiva etiqueta. Para poder utilizar entrenamiento por batches se utilizó un decorador que va generando los batches a partir del generador [6].

Los primeros resultados parecieron buenos al principio con un f1 de 97 % después de 3 épocas. Pero al ver las predicciones en el set de test se observó que estaba prediciendo siempre la clase 'O' que es ninguna entidad. Se estudió mas a fondo y se observó que la función auxiliar para calcular la métrica de loss que era categorical crossentropy no funcionaba muy bien con datasets desbalanceados y en este caso es un dataset supremamente desbalanceado pues casi todas las palabras carecen de entidad. Y la métrica de score era f1 MICRO la cual, igualmente presenta este problema entonces arrojaba "buenos" resultados. Se procedió entonces a buscar alguna función de pérdida que se comporte bien con datasets desbalanceados y se dio con una llamada focal loss [14] la cual se puede observar a continuación:

$$FL(\rho_t) = -\alpha_t(1-\rho_t)^\gamma \log \rho_t$$

En donde

- γ: constante que hace al modelo centrarse más en ejemplos difíciles.
- α: constante que le da equilibrio al desbalance de clases.

Esta función de pérdida fue creada para problemas de clasificación de objetos pues pasa mucho que los fondos planos prevalecen mucho en la imagen lo que hace que sea un dataset desbalanceado pues hay muchas porciones de la imagen que no son ningún objeto. Se puede

extrapolar este problema al nuestro, pues en nuestro caso estas porciones sin objeto se pueden traducir en porciones de texto sin entidad.

La primera prueba sencilla haciendo uso del focal loss, fue con 2 capas densas de 10 neuronas y con un gamma de 20 en la función de pérdida (gamma es el parámetro de penalización de la clase prevalente).

Como se explicó en la sección 2.2.3 de extracción de textos, cada categoría en realidad puede tener varias etiquetas pues depende del puesto donde este en la entidad, esto hace que al final queden las categorías muy divididas y hace que la categoría nula, que es la que mas prevalece, sea mucho mas mayoritaria con respecto al resto. Para esto se resumieron estas categorías en las mas generales de manera que ya el modelo no diferencia entre las partes de la entidad sino que cada palabra que compone la entidad es reconocida como la categoría independiente. De esta manera se trata de mejorar un poco el desbalanceo entre clases y que el modelo aprenda un poco más rápido. Además, se volvió a crear el corpus pero esta vez quitando las stop words para tratar de balancear un poco el dataset. Luego de realizar esto, se volvió a correr la red neuronal pero esta vez con una arquitectura de 3 capas ocultas de 100 neuronas cada una, para ver si se podía hacer overfitting y luego empezar a disminuirlo con dropout.

4.3.2.1 Caracteristicas para cada uno de los experimentos

- Experimento 1: categorical crossentropy de loss, 1 capa de 10 neuronas.
- Experimento 2: Focal loss (gamma=20, alpha=1), 2 capa de 10 neuronas, 20 épocas. En este aún está con todas las categorías.
- Experimento 3: 100 épocas y 3 combinaciones de gamma-alpha en el focal loss (20-1, 20-1, 10-5). Este ya es con la reducción de categorías y sin stop words.

Parte V

RESULTADOS

5.1 WNUT-17

La Tabla 1 contiene los resultados obtenidos para el dataset de wnut-17 utilizando word embeddings de glove y Pooled embeddings de flair, uno forward y uno backward. Estos tipos de embeddings están explicados en la sección 2.2.6 y 2.2.7. Estado del arte reportado de 49.49% (F1) en https://github.com/zalandoresearch/flair y el obtenido fue de 46.27%

Entity	precision	recall	accuracy	f1
corporation	0.4130	0.2879	0.2043	0.3393
creative-work	0.5000	0.1620	0.1394	0.2447
group	0.5263	0.1212	0.1093	0.1970
location	0.5676	0.5600	0.3925	0.5638
person	0.7798	0.5035	0.4408	0.6119
product	0.3721	0.1260	0.1039	0.1883

Tabla 1: Métricas de los resultados del WNUT-17

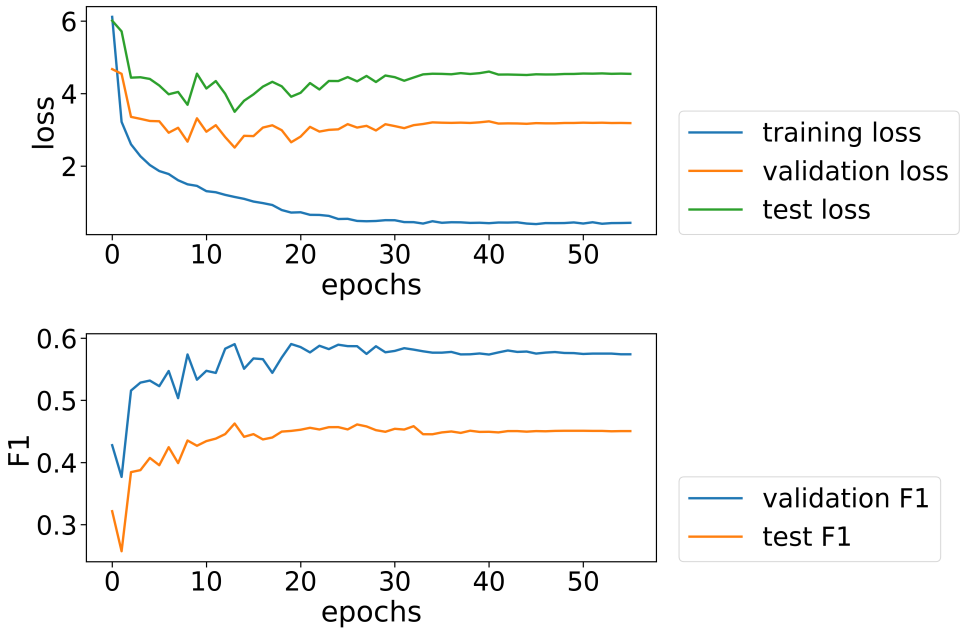

Figura 25: loss y f1 en WNUT-17 con embeddings de glove y flair

5.2 Benchmark Wikiner

La Tabla 3 contiene los resultados obtenidos para el dataset de wikiner utilizando como benchmark los embeddings de glove. Las entidades son como a continuación:

Entity	name
LOC	location
MISC	miscellaneous
ORG	organization
PER	person

Tabla 2: tags names

En la Figura 26, podemos observar el comportamiento del entrenamiento utilizando el dataset de wikiner con los embeddings de glove. Cada punto del gráfico es la métrica en una época en especifico.

Entity	precision	recall	accuracy	f1
LOC	0.8068	0.8362	0.6967	0.8212
MISC	0.7353	0.5947	0.4898	0.6576
ORG	0.7952	0.6701	0.5715	0.7273
PER	0.9089	0.9167	0.8395	0.9128

Tabla 3: Métricas de los resultados del benchmark

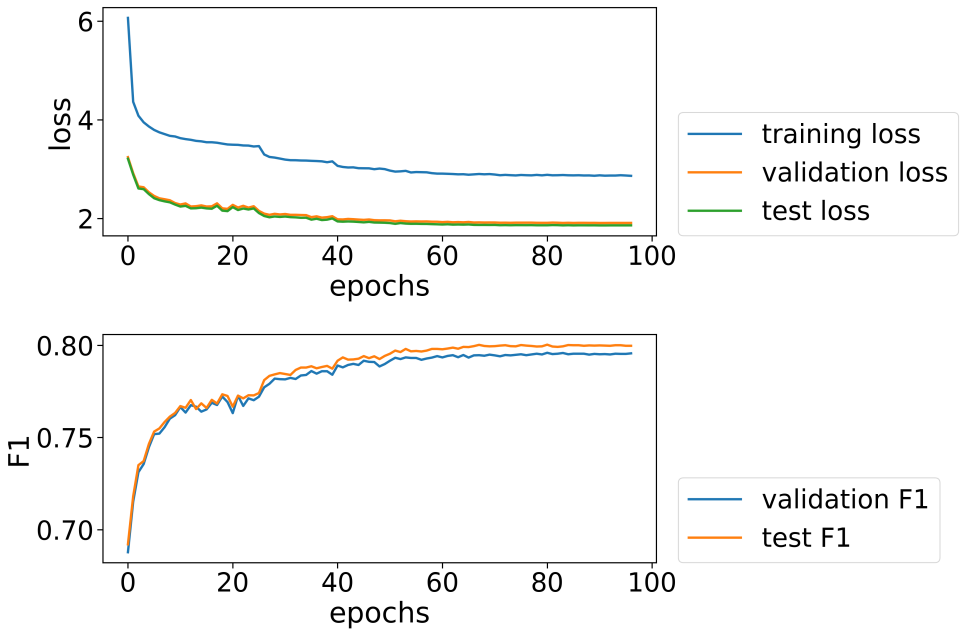

Figura 26: loss y f1 en benchmark

5.3 BERT WIKINER

La Tabla 4 contiene los resultados obtenidos para el dataset de wikiner utilizando los embeddings de bert.

En la Figura 27, podemos observar el comportamiento del entrenamiento utilizando el dataset de wikiner con los embeddings de BERT. El entrenamiento paró en 98 épocas porque el "learning rate" estaba muy pequeño. Este modelo mejoró considerablemente los resultados

Entity	precision	recall	accuracy	f1
LOC	0.8481	0.8822	0.7618	0.8648
MISC	0.8085	0.7446	0.6330	0.7752
ORG	0.8383	0.7837	0.6808	0.8101
PER	0.9473	0.9545	0.9064	0.9509

Tabla 4: Métricas de los resultados del primer experimento con BERT

obtenidos con los embeddings de glove, casi en un 10% y esto sin fine tunning de las capas de BERT. En versiones futuras de la librería Flair se podrá hacer esto y es muy probable que al afinar los parámetros del modelo aún mas se podrán obtener mejores resultados. En ambos casos (glove y bert) la entidad mejor reconocida fue la de "persona" y la peor es la miscelánea.

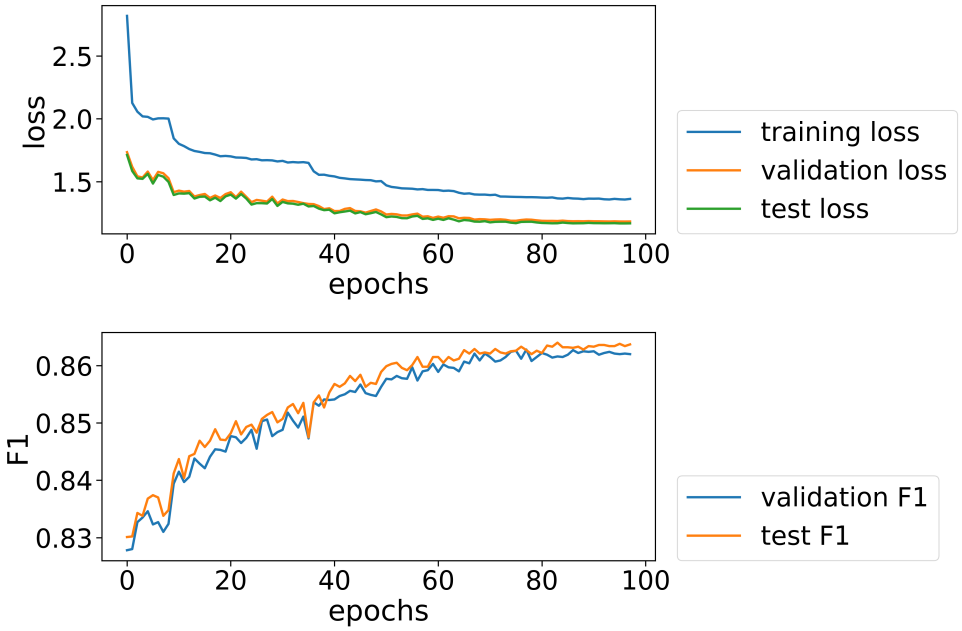

Figura 27: loss y f1 en bert

5.4 MUC3 y MUC4

5.4.1 BERT

5.4.1.1 Todo con Flair

La primera aproximación que fue con la librería Flair por completo no dio muy buenos resultados, y aún no esta muy desarrollada para poder modificar la arquitectura. En la Figura 28 se puede observar que en F1 no llegó ni a superar el 25%. Por esto se opto por la reformulación del código y arquitectura del problema.

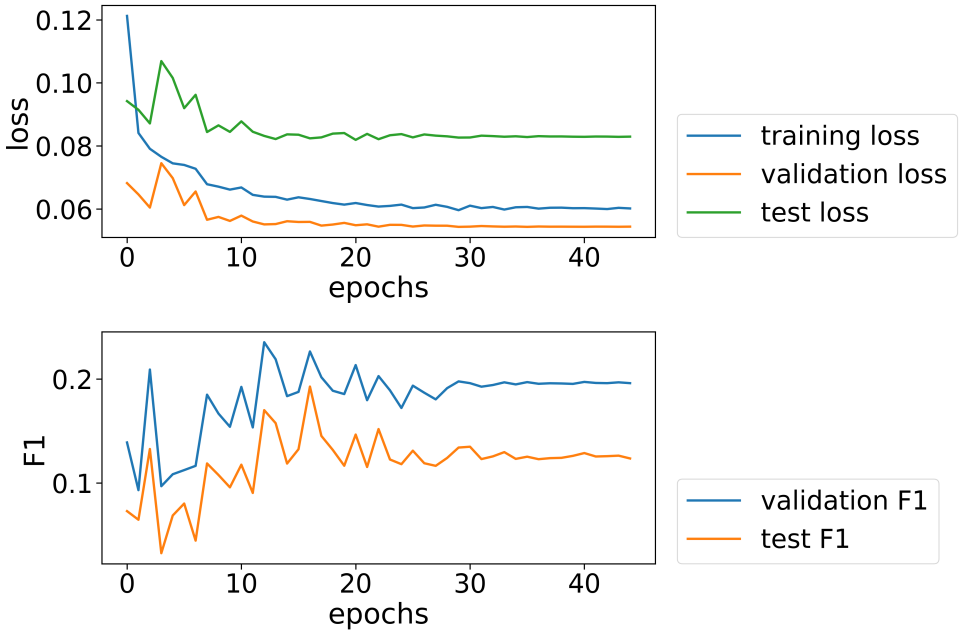

Figura 28: loss y f1 en la primera versión de BERT en el dataset de MUC

5.4.1.2 Embeddings con Flair y red neuronal con Keras

El **Experimento 1** clasificó todo como la clase nula.

En el **Experimento 2** utilizando focal loss ya en algunos casos empezó a predecir un poco mejor que cuando se tenía el categorical crossentropy como se puede ver en la matriz de confusión de la Figura 29. Se ve que casi toda la matriz esta con valores de 0 menos algunas casillas, en particular la columna 19 tiene casi todo en 1. Esta columna precisamente es la clase nula que clasifica siempre. Antes de utilizar el focal loss aparecían todos los valores en esta columna, acá al menos ya hay algunos casos como por ejemplo la clase 5 que ya logra clasificar algunos casos bien.

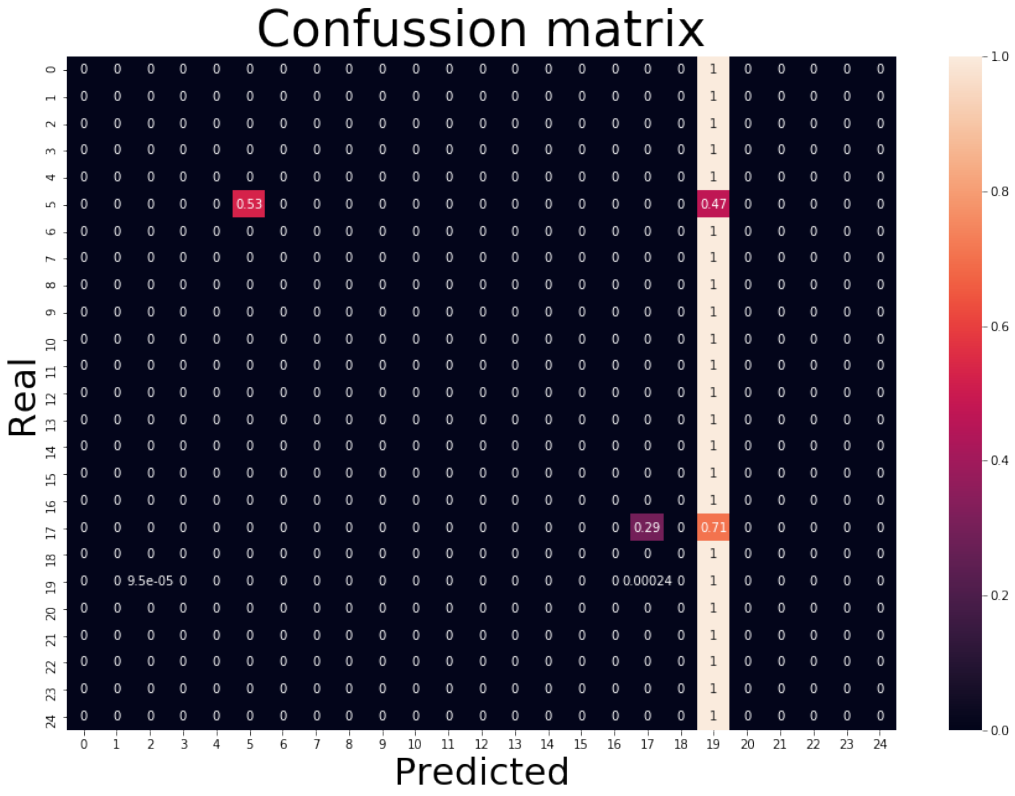

Figura 29: Matriz de confusión normalizada con gamma=20

En el **Experimento 3** tenemos 3 matrices de confusión, así como en el Experimento 2 se utilizará las matrices de confusión normalizadas pues como la clase mayoritaria es tan grande, hace que el gráfico sea "binario" pues la escala será demasiado amplia.

Las categorías son las siguientes:

Index	Entity
0	-
1	O
2	human target description
3	human target name
4	incident instrument id
5	perp individual id
6	perp organization id
7	physical target id

Tabla 5: Inices y entidades de la matriz de confusión

- **gamma 10 y alpha 5**: Se puede observar en la Figura 30. Fue la que peores resultado tuvo de las 3. Su mejor predicción fue en los instrumentos utilizados que predijo bien en 40% de los casos.

- **gamma 20 y alpha 2**: Se puede observar en la Figura 31. Obtuvo leves mejoras, llegando a 41% de predicciones correctas en las organizaciones autoras de los ataques.

- **gamma 20 y alpha 1**: Se puede observar en la Figura 32. Fue la que mejores resultados obtuvo, llegando a un 58% en las organizaciones autoras y a un 46% en los instrumentos usados para los atentados.

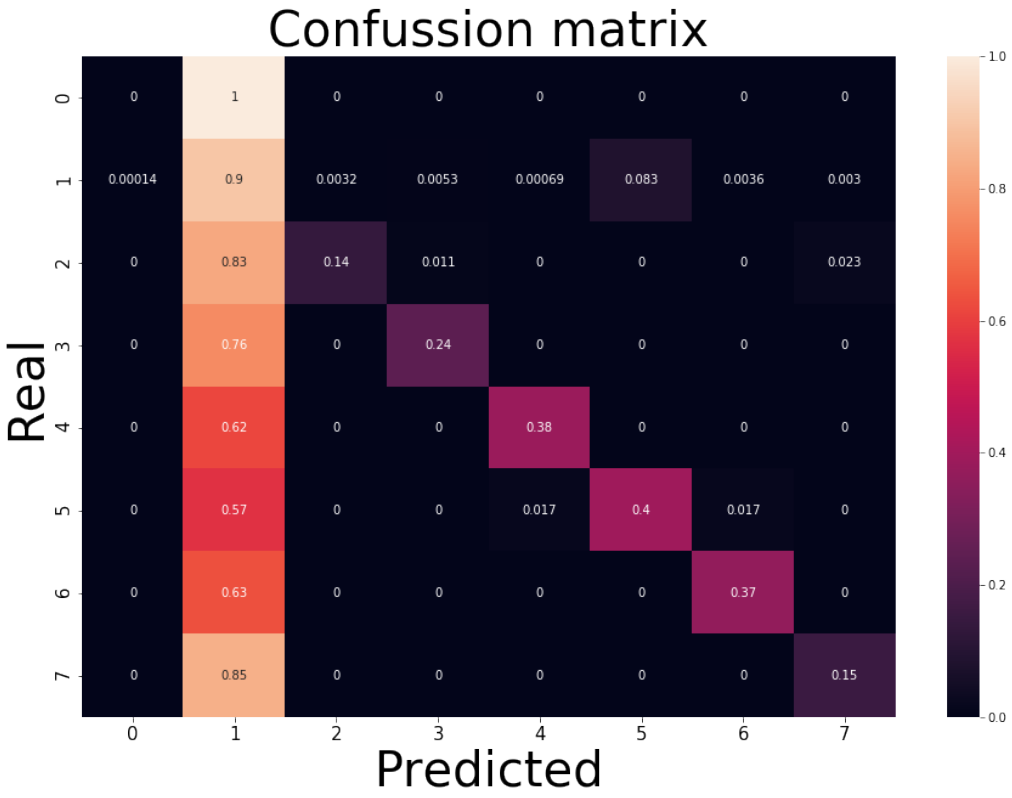

Figura 30: Matriz de confusión normalizada con gamma=10, alpha=5

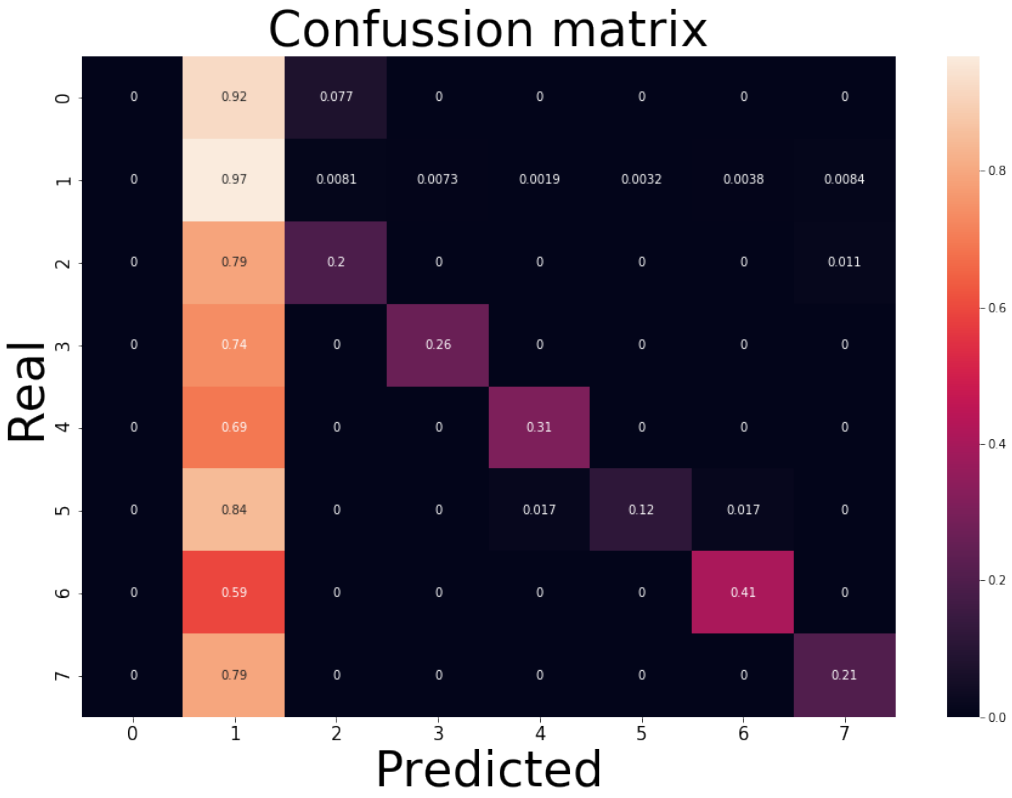

Figura 31: Matriz de confusión normalizada con gamma=20, alpha=2

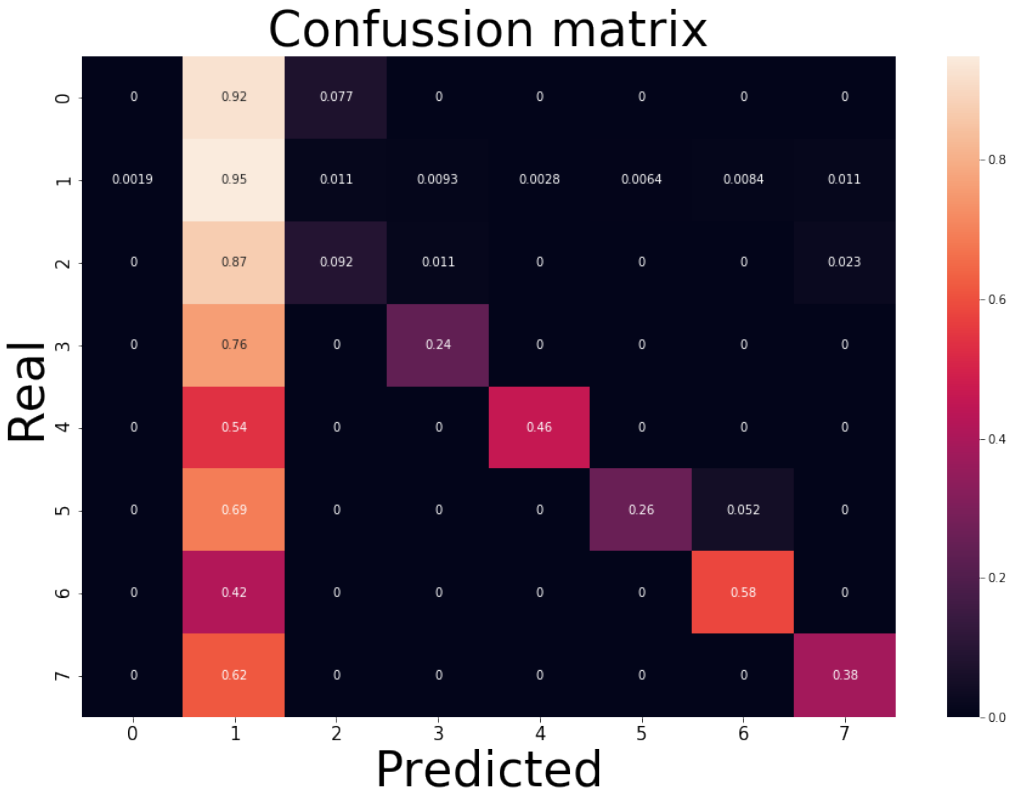

Figura 32: Matriz de confusión normalizada con gamma=20, alpha=1

Parte VI

DISCUSIÓN DE RESULTADOS

6.1 Contrastación de resultados con otros estudios similares

6.1.1 Wnut-17

Estado del arte: 49.49%

Resultados obtenidos: 46.27%

Se obtuvieron resultados un poco por debajo del estado del arte, aunque con arquitecturas muy sencillas, hace falta utilizar modelos un poco más completos para tratar de llegar a igualar estos resultados.

6.1.2 Wikiner

Estado del arte: 89%

Resultados obtenidos: 87%

Una vez mas, se obtuvieron resultados cercanos al estado del arte, hace falta ajustar un poco mas los parámetros para lograr igualarlo o sobrepasarlo. Tal vez cuando Flair permita hacer un

fine tunning de las capas de BERT, se logre ajustar mucho mejor el modelo.

6.1.3 MUC3 y MUC4

Estado del arte: 50% F1

Resultados obtenidos: - Alrededor de 30% F1

No son resultados muy satisfactorios, pero en general algunas categorías las clasifica muy bien. Sería interesante mirar en detalle porqué obtiene resultados tan bajos en esas otras categorías. Aún así se pudo observar que los parámetros del Focal-loss influyen mucho en los resultados, por lo que una optimización de hiperparámetros podría ayudar mucho.

Parte VII

CONCLUSIONES

7.1 Conclusiones

- En el presente trabajo se utilizaron técnicas modernas para el reconocimiento de entidades nombradas y extracción de información clave de documentos. Específicamente se utilizaron los embeddings de BERT los cuales tienen en cuenta el contexto y han demostrado mejorar el estado del arte en distintas tareas de procesamiento de lenguaje.

- Se utilizó la librería Flair para aprovechar sus utilidades en términos de preprocesamiento y más importante aún la implementación de los distintos embeddings incluyendo BERT, pero como es muy reciente dicha implementación, todavía no está muy optimizado y fue necesario partir el problema y utilizar elementos de mas bajo nivel como lo es Keras.

- Se logró hacer reconocimiento de entidades con un F1 de 89% en el dataset de wikiner y con un 30% en el dataset de MUC3 y MUC4 utilizando embeddings contextualizados.

Parte VIII

RECOMENDACIONES Y TRABAJO FUTURO

8.1 Recomendaciones

- A la hora de hacer detección de entidades es de suma importancia la parte de preprocesamiento del texto pues muchas veces los modelos fueron preentrenados con textos con ciertas características como todo en mayúsculas o minúsculas, etc.
- Para problemas de mucho texto se hace necesario contar con buen hardware y mucho mejor si se cuenta con una GPU pues son problemas complejos y con muchos datos que a veces solo con una CPU tardan mucho.
- Es importante entender bien toda la estructura y funcionamiento de los embeddings contextualizados para así poder utilizar estrategias adecuadas y obtener buenos resultados.

8.2 Trabajo Futuro

- Probar con los datasets mas recientes de MUC (el último es el #7) para observar si se obtienen buenos resultados.
- Hacer un fine tunning de las capas de bert pues por ahora se dejaron quietas y solo se añadieron capas al final a la tarea de clasificación. Esto depende de las versiones futuras de la librería Flair o la implementación de otras librerías.
- Probar más arquitecturas mas complejas que aunque se demoren más depronto obtengan mejores resultados.
- Hacer optimización de hiperparámetros con gridsearch o BayesSearch para mejorar los resultados en el dataset de MUC.
- Utilizar otros datasets como de clasificación de artículos cientificos u otro de los casos de uso de este problema.
- Utilizar BioBert en problemas biomédicos como por ejemplo un doctor virtual que haga recomendaciones de acuerdo a síntomas.

REFERENCIAS

[1] *Diccionario de la Real Academia Española (22.a ed)*. 2019.

[2] JEFFREY PENNINGTON, RICHARD SOCHER, CHRISTOPHER D. MANNING . Glove: Global vectors for word representation. `https://nlp.stanford.edu/projects/glove/`, 2014. [Online; accessed 28-May-2019].

[3] AKBIK, A., BERGMANN, T., AND VOLLGRAF, R. Pooled contextualized embeddings for named entity recognition. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)* (Minneapolis, Minnesota, June 2019), Association for Computational Linguistics, pp. 724–728.

[4] ALAMMAR, J. The illustrated bert, elmo, and co. (how nlp cracked transfer learning).

[5] ALAN AKBIK, DUNCAN BLYTHE, R. V. Contextual string embeddings for sequence labeling.

[6] BARBERO, A. neurowriter. `https://github.com/albarji/neurowriter/blob/master/neurowriter/genutils.py`, 2017.

[7] CHINCHOR, N. Muc-3 evaluation metrics.

[8] CHINCHOR, N., LEWIS, D. D., AND HIRSCHMAN, L. Evaluating message understanding systems: An analysis of the third message understanding conference (muc-3). *Comput. Linguist. 19*, 3 (Sept. 1993), 409–449.

[9] DEVLIN, J., CHANG, M.-W., LEE, K., AND TOUTANOVA, K. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018.

[10] HUGGINGFACE. pytorch-pretrained-bert. `https://github.com/huggingface/pytorch-pretrained-BERT`, 2018.

[11] ILYA SUTSKEVER, O. V., AND LE, Q. Sequence to sequence learning with neural networks.

[12] KYZHOUHZAU. Bert-ner. `https://github.com/kyzhouhzau/BERT-NER`, 2019.

[13] LEE, J., YOON, W., KIM, S., KIM, D., KIM, S., SO, C. H., AND KANG, J. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *CoRR abs/1901.08746* (2019).

[14] LIN, T., GOYAL, P., GIRSHICK, R. B., HE, K., AND DOLLÁR, P. Focal loss for dense object detection. *CoRR abs/1708.02002* (2017).

[15] NOTHMAN, J., CURRAN, J. R., AND MURPHY, T. Transforming wikipedia into named entity training data.

[16] RAJ, K. Bert-ner. `https://github.com/kamalkraj/BERT-NER`, 2019.

[17] RILOFF, E., AND PATWARDHAN, S. Learning domain-specific information extraction patterns from the web.

[18] SKYMIND. How do transformers work in nlp? a guide to the latest state-of-the-art models. [Online; accessed June 27, 2019].

[19] SKYMIND. A beginner's guide to word2vec and neural word embeddings, 2018. [Online; accessed June 27, 2019].

[20] VASWANI, A., SHAZEER, N., PARMAR, N., USZKOREIT, J., JONES, L., GOMEZ, A. N., KAISER, L., AND POLOSUKHIN, I. Attention is all you need. *CoRR abs/1706.03762* (2017).

[21] WEBKNOX. Named-entity-definition. `http://webknox.com/p/named-entity-definition`, 2019. [Online; accessed 27-May-2019].

[22] ZHU, Y., KIROS, R., ZEMEL, R. S., SALAKHUTDINOV, R., URTASUN, R., TORRALBA, A., AND FIDLER, S. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. *CoRR abs/1506.06724* (2015).

Anexos

Anexos A

Bitácora

Dia	Horas trabajadas	Trabajos realizados
1	2	Familiarización librería Flair
2	2	Familiarizacion Bert y NER
3	5	Familiarizacion servidores del IIC y primeros programas de prueba
4	2	Primera prueba dataset WNUT17
5	2	Revisión primera prueba, correcciones y volver a correr
6	3	Revision segunda prueba y primeros pasos con wikiner
7	3	Documentacion resultados parciales
8	2	Lectura articulos sobre NER en BERT
9	2	3d embeddings usando plotly, primera aproximacion
10	2	terminar plots de embeddings y avanzar en informe
11	3	Aprender a utilizar instancias de ec2 de aws y arreglos experimentos
12	3	GloVe y Flair en el dataset de 20 newsgroup
13	5	ETL Dataset MUC3 y MUC4
14	5	ETL Dataset MUC3 y MUC4
15	5	ETL Dataset MUC3 y MUC4
16	5	ETL Dataset MUC3 y MUC4
17	5	condensacion codigos, analisis exploratorio y primera prueba con BERT en MUC3 y MUC4
18	4	Modificacion ETL para separar por frases e informe
19	6	Revision primera prueba BERT-MUC34 y ajustes
20	5	Reformulación estrategia BERT
21	5	Reformulación estrategia BERT y trabajo escrito
22	5	Construccion red neuronal keras con embeddings de BERT
23	5	Familiarizacion Decoradores y generadores para implementar en el fit_generator de keras
24	6	Pruebas red neuronal keras
25	5	Pruebas red neuronal keras y presentacion final
26	5	Pruebas red neuronal
27	5	Pruebas red neuronal
28	5	Pruebas red neuronal
29	5	Pruebas red neuronal
30	5	Memoria y presentacion final
Total	122	

Figura 33: bitácora

Anexos B

Software

Se utilizó un entorno virtual de anaconda con los siguientes requerimientos:

- - cudatoolkit==9.0
- - cudnn==7.1.2
- - keras-gpu=2.2.2
- - jupyter=1.0.0
- - nb_conda=2.2.1
- - matplotlib=2.2.2
- - scikit-learn=0.19.1
- - pandas=0.23.0
- - scikit-image=0.14
- - pip:

- - scikit-optimize==0.5.
- - flair==0.4.2

Anexos C

Hardware

instancia EC2 de AWS con GPU y 16 cores