UNIVERSIDAD DE LA REPÚBLICA URUGUAY

FACULTAD DE INGENIERÍA

Extracción de fuentes de opinión a partir de textos de prensa uruguaya

Informe de Trabajo Final de Licenciatura presentado por

Cindy Espíndola

en cumplimiento parcial de los requerimientos para la graduación de la carrera de Licenciatura en Computación de Facultad de Ingeniería de la Universidad de la República

Supervisores

Aiala Rosá
Juan José Prada

Montevideo, 03 de diciembre de 2024

Agradecimientos

Quisiera agradecer a mi familia y amigos por siempre brindar su gran apoyo y motivación a lo largo de toda mi carrera. En especial a mis padres por su paciencia y ayuda en todo momento a lo largo de este trayecto.

Además agradezco enormemente a mis tutores, Aiala y Juan José, por su continua disposición y guía en la elaboración de este trabajo.

Resumen

En documentos de prensa escrita frecuentemente se incluyen opiniones emitidas por los principales involucrados, ya sean personas u organizaciones. Es por esto que resulta de utilidad poder realizar un procesamiento automático de documentos de prensa para detectar estas opiniones y sus fuentes.

En este trabajo se tiene como foco una parte del proceso de extracción de esta información: la identificación de las fuentes de opinión. Se modela la tarea como una clasificación de secuencias, procesando cada oración y etiquetando las palabras conforme pertenecen o no a la(s) fuente(s) de opinión en caso de existir.

Se hizo uso de la arquitectura Transformer y modelos existentes basados en lenguaje español, realizando un *fine tuning* con nuevos ejemplos para ajustar estos modelos a la tarea mencionada.

Se realizó la comparación entre el uso del modelo de lenguaje para el español habitualmente utilizado, BETO, y un modelo generado específicamente a partir de documentos de prensa uruguaya, ROUBERTA. Como parte de la evaluación, además se consideraron varias alternativas cambiando algunos de sus hiperparámetros.

Por otro lado, se utilizó un modelo de lenguaje generativo *open source* para realizar algunas pruebas con el fin de evaluar si es posible prescindir del entrenamiento con un conjunto anotado en forma manual, para en su lugar utilizar *prompts* con un conjunto muy limitado de ejemplos. Esto dio como resultado respuestas poco predecibles y dificultades por parte del modelo para cumplir con la tarea y formato establecidos.

Se pudo comprobar que, de las opciones vistas con ambos modelos a los cuales se aplicó *fine tuning*, se obtienen resultados relativamente similares.

Los resultados evaluados sobre el conjunto de testeo presentaron una medida F exacta de 81.1 % con los modelos basados en ROUBERTA y 75.4 % con los modelos basados en BETO. Esto demuestra que, a pesar de haber tenido un corpus de datos significativamente menor durante su preentrenamiento, el utilizar un modelo preentrenado con el mismo dominio específico de la tarea a realizar permite obtener muy buenos resultados.

Palabras clave: transformers, extracción de opiniones, sequence labelling, procesamiento de lenguaje natural

Índice general

Capítulo 1

Introducción

Los textos de prensa uruguaya, como todo documento periodístico, contienen información relevante de la situación del país. Esto, entre otras cosas, incluye en muchas ocasiones información que puede ser subjetiva, normalmente, opiniones de distintos agentes.

Es necesario poder identificar dentro de un texto, cuando existe una opinión, quién es la fuente que genera esta opinión, tanto para comprender que la información allí incluida podría ser subjetiva o no, como para entender las posturas de las diferentes entidades, ya sean figuras públicas u organizaciones, empresas, grupos, etc.

Existe un antecedente de esta tarea que había sido analizada y desarrollada mediante el uso de métodos de *Conditional Random Fields* y reglas contextuales por Rosá (2011). Sin embargo, cada vez más se está observando la eficacia al hacer uso de modelos de lenguaje basados en arquitectura Transformer para un sinnúmero de tareas de procesamiento del lenguaje natural, lo que plantea la interrogante de si se podrían emplear estos modelos para obtener buenos resultados en la extracción de fuentes de opinión.

En este trabajo se realizan diferentes pruebas con el fin de determinar si se obtiene un buen desempeño utilizando modelos del lenguaje en español, con particular interés en los resultados obtenidos al utilizar un modelo de lenguaje conocido como ROUBERTA(Filevich, Marco, Castro, Chiruzzo, y Rosa, 2024), el cual fue preentrenado exclusivamente con documentos de prensa uruguaya. Con el uso de este modelo se busca a su vez evaluar si un modelo entrenado con un corpus de dominio específico obtiene buenos resultados al compararlo con un modelo general, entrenado con un corpus mucho mayor, también en el mismo idioma.

1.1. Motivación

Con el uso de los portales digitales, los medios de prensa con el tiempo cada vez han ido incrementando la cantidad de noticias que se publican a diario, no

limitándose a las entregas periódicas en formatos tradicionales como había sido históricamente, lo cual genera un gran volumen de información periodística, en el formato de artículos de prensa escritos.

Esto motiva la necesidad de obtener herramientas para la extracción de la información sobre este tipo de textos. Los documentos en el ámbito de la comunicación y periodismo habitualmente contienen opiniones realizadas por los principales actores en las noticias presentadas, los cuales pueden ser tanto figuras públicas como colectivos o agrupaciones u organizaciones, entre otros.

Es de importancia, por lo tanto, poder identificar dentro de las opiniones incluidas en estos documentos quién dijo qué cosa, cuáles fueron las posturas tomadas sobre un determinado asunto, etc. Como parte de este proceso se necesita identificar las fuentes de opinión, lo cual es una motivación para evaluar posibles herramientas a utilizar para la realización de esta tarea.

Si bien, como se verá en la sección 2.8 existen otros trabajos anteriores que se han realizado con similares objetivos, pocos son aquellos que trabajan sobre el lenguaje español. En los modelos de lenguaje basados en transformers se ha evidenciado que se obtienen mejores resultados sobre modelos en un único idioma que en modelos multilingües, es por esto que si la tarea solamente se aplica a un determinado idioma, es habitual trabajar con sistemas creados para este.

El uso de los modelos que emplean la arquitectura Transformer ha probado ser de utilidad para diversidad de tareas. Esto ha motivado la necesidad de evaluar su eficacia para el problema presentado, en especial, en este trabajo, hacemos uso del modelo ROUBERTA antes mencionado, el cual demostró en distintas pruebas que además permite un mejor entendimiento del contexto cultural que involucra al país (por ejemplo, prediciendo palabras como “murga”, “caif”, etc., donde modelos más generales del español no tienen esta información contextual). A su vez, comparamos esto con algunas otras técnicas y modelos para investigar posibles soluciones al problema de extracción de fuentes de opinión.

1.2. Objetivos

El presente trabajo tiene como objetivo central la aplicación de los modelos de lenguaje creados con arquitectura Transformer para la tarea de extraer de manera automática fuentes de opinión en oraciones pertenecientes a documentos de prensa, en particular, documentos en lenguaje español pertenecientes a los medios de prensa uruguaya.

Para esto se plantearon los siguientes objetivos:

- Investigar el estado del arte en el uso de estas herramientas y otros trabajos con el objetivo de realizar esta tarea.

- Realizar pruebas utilizando dos de las técnicas para trabajar con este tipo de modelos: *fine tuning* y *prompting*.

- Analizar y comparar los resultados obtenidos con las diferentes opciones consideradas. En particular, comparar el uso de un modelo preentrenado sobre el mismo dominio particular de esta tarea contra un modelo preentrenado sobre un corpus mayor pero general de textos en español.

1.3. Cronograma

Tareas	Duración	Fechas
Estudio de trabajos recientes y otros trabajos anteriores relacionados con la tarea u otras tareas aplicadas sobre el mismo dominio de información	4 semanas	Abril-Mayo
Investigación de herramientas a utilizar para la tarea propuesta	3 semanas	Mayo-Junio
Implementación inicial de la solución	6 semanas	Junio-Agosto
Experimentación y comparación entre diferentes opciones de modelos y prompts. Ajustes a la solución original	6 semanas	Julio-Setiembre
Realización del informe final	8 semanas	Setiembre-Noviembre

1.4. Organización del documento

En esta sección se introducen los distintos capítulos que componen el cuerpo del presente documento, así como una breve descripción de cada uno de ellos y su contenido.

- **Introducción**: En este primer capítulo se introduce el lector a los objetivos y motivaciones que llevaron a la realización de este trabajo.
- **Marco teórico y estado del arte**: Se presenta el contexto teórico necesario para comprender el modelado de la solución, así como las diferentes tareas y evaluaciones realizadas. A su vez, en este capítulo se incluyen descripciones de algunos trabajos relacionados con la tarea a desarrollar.
- **Definición del problema**: Aquí se describen las herramientas y recursos utilizados, además de las diferentes etapas que componen la solución al problema introducido anteriormente.
- **Implementación y Experimentación**: En este capítulo se presenta la implementación final realizada para las distintas etapas del proceso de obtención de fuentes de información introducidas en el capítulo previo. Además se realiza un análisis sobre los resultados individuales y comparativos de las diferentes alternativas implementadas.

- **Conclusiones y trabajo futuro**: Finalmente, en este capítulo se abordan las conclusiones de la experimentación realizada, así como posibles trabajos futuros.

Capítulo 2

Marco teórico y estado del arte

En el presente capítulo se introducen los conceptos teóricos a tener en consideración detrás de los experimentos realizados. A su vez, se presenta el estado del arte de la temática involucrada y se mencionan otros trabajos actuales asociados a tareas similares.

2.1. Procesamiento del Lenguaje Natural

El Procesamiento del Lenguaje Natural (usualmente conocido por sus siglas PLN) es un campo de la inteligencia artificial cuyo objetivo es *"[..] to get computers to perform useful tasks involving human language, tasks like enabling human-machine communication, improving human-human communication, or simply doing useful processing of text or speech.[..]"* (Jurafsky y Martin, 2009). Es decir, desarrollar sistemas que puedan realizar tareas que involucren el lenguaje humano.

En la actualidad, cada vez son más los sistemas que, en mayor o menor medida, utilizan métodos y/o técnicas de PLN. Si bien estos sistemas son comúnmente de gran utilidad para facilitar la interacción entre las personas y las aplicaciones, como es en el caso de los asistentes virtuales o los *chatbots*, otra área de investigación dentro del procesamiento del lenguaje natural es el análisis y la extracción automatizada de información.

El beneficio del uso de estrategias de PLN para este tipo de análisis surge del gran volumen de datos sobre el que se busca extraer y analizar información, por lo tanto, realizar estas tareas de forma manual conllevaría un esfuerzo muchas veces inviable.

Para esto se utilizan diferentes estrategias de acuerdo con el tipo de información que se busca obtener.

En algunas de las aplicaciones de este campo lo que se busca es categorizar cada texto de acuerdo a una colección definida, esto se conoce como clasifica-

ción de textos o *text classification* (Manning, Raghavan, y Schütze, 2008). Esto también se realiza, por ejemplo, para tareas de análisis de sentimientos.

Entre otro tipo de tareas que forman parte del PLN, también existen aplicaciones donde, como se verá en este trabajo, lo que se busca es analizar a nivel de cada oración para poder extraer determinada información. Este tipo de procesamientos se definen como tareas de clasificación de secuencias.

2.2. Tarea de clasificación de secuencias y etiquetado BIO

Cuando se trata de analizar textos en lenguaje natural, en muchas ocasiones es necesario poder identificar partes de cada oración que contienen información relevante para la tarea a realizar.

Este es el caso por ejemplo, de las tareas NER (*Named Entity Recognition*). En ellas se busca obtener determinadas entidades como pueden ser personas, organizaciones, ubicaciones, etc. Otros ejemplos incluyen *POS-Tagging*, donde se identifican las distintas partes de la oración (como pueden ser adjetivos, verbos, entre otros), y *Semantic Role Labelling*, donde se identifican roles semánticos como "quién", "dónde", etc. Todas estas tareas pueden ser modeladas como tareas de clasificación de secuencias.

En la clasificación de secuencias o *Sequence Labelling*, como se expresa en (Jurafsky y Martin, 2024): "*[...] the network's task is to assign a label chosen from a small fixed set of labels to each element of a sequence[...]*". Es decir, el objetivo general de estas tareas es asignar a cada elemento de la secuencia una de las posibles etiquetas definidas.

En esta tarea, se recibe como entrada una secuencia de palabras, luego de procesado se devuelve una secuencia de igual longitud a la de entrada, donde cada elemento en la salida es una etiqueta asociada a la clasificación de la palabra en su misma posición. Las clases posibles son un conjunto finito predefinido.

Las etiquetas a asignar pueden estar en diferentes formatos, uno de los formatos habitualmente utilizados son las etiquetas BIO**(Beginning - Inside - Outside)**. Definido por primera vez en (Ramshaw y Marcus, 1995), en este formato para cada tipo de entidad se hace la distinción entre la primera palabra y el resto de palabras que la conformen. La etiqueta *Outside* se utiliza solamente para aquellas palabras que no forman parte de ninguna de las clases definidas.

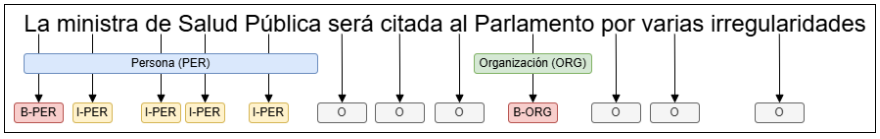

Figura 2.1: Ejemplo de *Named Entity Recognition* haciendo uso de etiquetas BIO

En la Figura 2.1 se aprecia un ejemplo de etiquetado BIO aplicado a la

tarea NER antes mencionada. En este caso se indican las categorías "Persona" y "Organización" por lo que el conjunto de etiquetas posibles si solo se consideran estos dos tipos de entidades sería el formado por las siguientes: B-PER, I-PER, B-ORG, I-ORG y O.

En el presente trabajo, en particular, hacemos uso del esquema BIO para modelar el problema planteado. En este sentido, consideramos únicamente el tipo de entidad "Fuente de opinión (FU)", y el problema consiste en la correcta clasificación de secuencias de acuerdo a este esquema.

2.3. Large Language Models

Los modelos de lenguaje se presentan en (Jurafsky y Martin, 2024) como: *"[..] models that assign a probability to each possible next word"*, en otras palabras, dada una secuencia de palabras de entrada, utilizando el modelo se puede obtener aquella palabra (o secuencia de palabras) incluida en su vocabulario que es más probable como continuación del texto proporcionado.

Si bien, a simple vista, los modelos de lenguaje pueden parecer herramientas para una aplicación específica (predecir la palabra siguiente en una secuencia), en la actualidad, son la base de numerosas tareas del procesamiento del lenguaje natural, debido a que cuando el modelo aprende estas probabilidades, en realidad de manera implícita está aprendiendo mucha más información acerca del lenguaje y el tipo de textos sobre el cual está siendo entrenado. Se aprenden reglas semánticas y sintácticas, entre otra información contextual, lo que permite que sea utilizado para un vasto conjunto de tareas.

Para poder crear este tipo de modelos han existido diferentes estrategias a lo largo de los años. En los modelos actuales se realiza un proceso de preentrenamiento, donde el modelo aprende tanto el vocabulario como las probabilidades con base en un amplio corpus de textos. El resultado de realizar un preentrenamiento sobre un modelo de lenguaje es lo que se conoce como Large Language Models o LLM (Jurafsky y Martin, 2024)

Estos conjuntos de datos pueden ser generales o particulares. Los modelos generales utilizan conjuntos de datos de gran volumen sobre contenido sumamente variado, como ocurre con los grandes modelos más conocidos, donde se utilizó gran parte del contenido de internet para el entrenamiento. Por otra parte, los modelos particulares generalmente son entrenados con conjuntos de menor tamaño, donde lo que se busca es que el modelo aprenda de un subdominio específico, a fin de obtener buenos resultados para tareas específicas a este subdominio utilizando una menor cantidad de recursos.

2.4. Transformers

El Transformer: es una arquitectura de redes neuronales, la cual se convirtió en la arquitectura estándar para los LLM. Fue definida por primera vez en 2017, en el artículo: *"Atention is all you need"* creado por Vaswani y cols. (2017).

Anterior a esto, lo que era considerado el estado del arte para tareas de PLN consistía en el uso de **Redes Neuronales Recurrentes** (**RNN** por sus siglas en inglés), en particular los modelos **LSTM** o de larga memoria a corto plazo, presentados por Hochreiter y Schmidhuber (1997), los cuales permitían mejorar el comportamiento de este tipo de modelos sobre secuencias largas de palabras.

Los modelos basados en RNN presentan un problema conocido como "problema de desvanecimiento de gradientes". Como se explica en (Jurafsky y Martin, 2024): *"[...] during the backward pass of training, the hidden layers are subject to repeated multiplications, as determined by the length of the sequence. A frequent result of this process is that the gradients are eventually driven to zero[...]"*.

Esto quiere decir que dado el proceso de RNN, las capas ocultas en estos modelos implican una cantidad de multiplicaciones (de números pequeños) que crece a medida que el tamaño de la secuencia es mayor, lo que lleva a que los gradientes se aproximen a cero.

Este problema era mitigado por los modelos LSTM, dado que incluyen un mecanismo para hasta cierto punto discernir entre lo que puede ser olvidado y lo que se debería mantener del contexto. Sin embargo, continuaba siendo una limitante para los modelos de lenguaje natural, ya que muchas veces el contexto necesario para correctamente responder a una tarea de este tipo suele representar una cantidad de palabras con la que la mayoría de este tipo de modelos tenía dificultades.

Además de esto, los modelos basados en estas estructuras tenían la desventaja de que eran secuenciales, lo que significaba que para procesar una secuencia de entrada, cada elemento debía ser procesado por todas las capas de la red antes de poder procesar el siguiente elemento de la secuencia. Esto no permitía la ejecución de operaciones en paralelo para mejorar el tiempo de ejecución, lo cual en la gran cantidad de cálculos necesarios para estos entrenamientos es crucial.

Por su parte, los modelos transformers representan una mejoría respecto de los anteriores en ambas limitantes. Esto se debe a que el transformer es *"[...] a model architecture eschewing recurrence and instead relying entirely on an attention mechanism to draw global dependencies between input and output.[...]"*(Vaswani y cols., 2017). En otras palabras, es una arquitectura que se basa en mecanismos de atención para obtener dependencias globales entre entradas y salidas, evitando así la utilización de recurrencias.

A su vez, estos modelos no trabajan de forma directa con palabras como los modelos tradicionales, sino que lo hacen con unidades conocidas como token (salidas resultantes de pasar por un algoritmo de tokenización[1] (Jurafsky y Martin, 2024)). Estos *tokens* pueden ser palabras completas o unidades menores a la palabra, de esta forma se evitan problemas ocasionados por palabras recibidas en la entrada que no se encuentren dentro del vocabulario finito ya establecido.

[1]Uno de los algoritmos de tokenización más utilizados es el conocido como **Byte-Pair Encoding**, el cual comienza considerando como tokens a todos los caracteres presentes en el conjunto de entrenamiento, más un caracter especial para el "fin de palabra". En cada iteración se agregan al conjunto de tokens las combinaciones más frecuentes en el corpus (la cantidad de iteraciones se encuentra predefinida).

Para comprender a qué hace referencia y cómo se comportan este tipo de modelos es que a continuación se presentan sus diferentes componentes o bloques básicos.

2.4.1. Bloques básicos

Los transformers cuentan con una arquitectura modular basándose en lo que se conoce como **bloque transformer**. Habitualmente, los grandes modelos de lenguaje como GPT o BERT utilizan un numeroso conjunto de capas compuestas por estos bloques (por ejemplo, en el caso del primer modelo BERT presentado por Devlin, Chang, Lee, y Toutanova (2019), se contaba con 12 capas de bloques transformers). Cada salida de un bloque transformer alimenta la entrada del siguiente. Esto les permite "comprender" múltiples relaciones complejas entre las secuencias de palabras de entrada.

Como se aprecia en la Figura 2.2, estos bloques están compuestos por capas de diferentes tipos:

- Capas de atención (*Multi-Head Attention* y *Masked Multi-Head Attention*).
- Capas de propagación hacia adelante (*Feed Forward*)
- Conexiones residuales y capas normalizadoras (*Add and Norm* en la figura)

Además de un procesamiento inicial para obtener sus *embeddings*(vectores que representan a las palabras (Jurafsky y Martin, 2024)) e información posicional, así como una etapa final donde se realiza una transformación lineal y *softmax* para obtener las probabilidades finales.

Softmax o también llamado regresión logística multinomial es un tipo de clasificador, el cual recibe un vector de k valores y devuelve un nuevo vector donde para cada índice tiene un valor entre 0 y 1 asociado a la probabilidad de la clase de ese índice de acuerdo a la siguiente ecuación (Jurafsky y Martin, 2024):

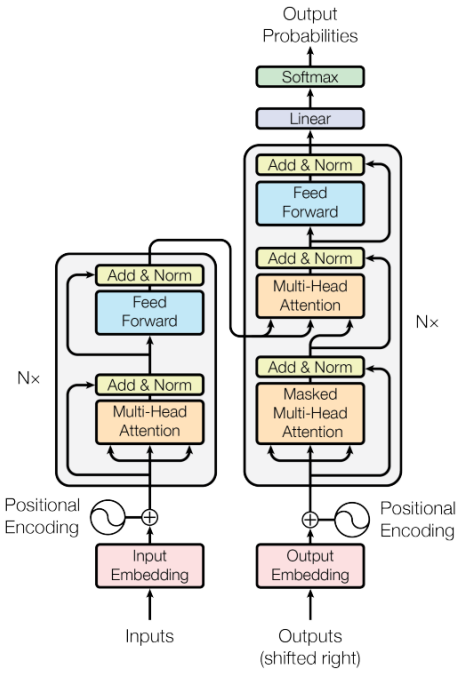

Figura 2.2: Arquitectura del modelo transformer presentada en el artículo original

$$softmax(z_i) = \frac{exp(z_i)}{\sum_{j=1}^{K} exp(z_j)} 1 \leq i \leq K \tag{2.1}$$

El bloque de capas que se observa a la izquierda se conoce como el "Encoder" o codificador, mientras que el derecho es el "Decoder" o decodificador.

En primer lugar, las entradas (o salidas de bloques intermedios, ya que también deberán pasar por estas etapas al convertirse a su vez en entradas del siguiente bloque) deben pasar a ser un conjunto de *embeddings*, lo que a su vez representa a los significados de estas entradas, donde palabras cuyo significado tiene alguna relación se encuentran a menor distancia que palabras no relacionadas. Estas representaciones, a medida que se avanza en las capas, van resultando mejores para el contexto específico que se tenía de entrada a la red, dado que los vectores que se van generando incorporan información más contextual. Se utiliza la codificación posicional para representar la información del orden que presentan los tokens de entrada.

Como definen Jurafsky y Martin (2024): *"Attention is the mechanism in the transformer that weighs and combines the representations from appropriate other tokens in the context from layer k-1 to build the representation for tokens in layer k"*, es decir que es un mecanismo el cual asigna pesos y combina las representaciones de otros *tokens* apropiados en el contexto de la capa inmediata anterior para construir las representaciones de la capa actual. Lo que se conoce como *attention head* recibe como entrada un vector x_i y devuelve otro vector a_i del mismo tamaño, calculado a partir de las siguientes ecuaciones:

$$score(x_i, x_j) = \frac{q_i.k_j}{\sqrt{d_k}} \tag{2.2}$$

$$\alpha_{ij} = softmax(score(x_i, x_j)) \forall j \leq i \tag{2.3}$$

$$a_i = \sum_{j \leq i} \alpha_{ij} v_j \tag{2.4}$$

Donde q_i, k_i y v_i son el resultado de multiplicar el vector x_i de entrada con las matrices de pesos correspondientes, cada una haciendo referencia a un distinto rol que ocupa el vector x_i en el proceso: como elemento actual siendo comparado (*query*), como elemento anterior siendo comparado con el actual (*key*) y como valor del elemento anterior (*value*).

Los mecanismos **Multi-head Attention**, también conocidos como **Self-attention** cuentan con un conjunto de *attention heads* paralelos. Cada uno de ellos provee distinta información de las relaciones entre los tokens de entrada. En el caso de los mecanismos **Masked Multi-head Attention** además se asegura de solamente considerar las salidas en posiciones menores o iguales a la actual, son estos tipos de mecanismos los que permiten entrenar modelos para generar texto como los modelos GPT (*Generative Pre-trained Transformers* o Transformers Generativos Preentrenados).

El último gran paso tanto en el bloque codificador como decodificador es pasar por una capa **Feed-forward**, esta capa es "*[...] a fully-connected 2-layer network, i.e., one hidden layer, two weight matrices[...]*" (Jurafsky y Martin, 2024) (una red formada por una capa oculta completamente conectada con dos matrices de pesos). Esta es aplicada a cada posición de manera idéntica y aislada, mediante dos transformaciones lineales con una activación ReLU entre ellas, como se define en la siguiente ecuación: $\text{FFN}(x) = \max(0,\, xW_1 + b_1)W_2 + b_2$ (Vaswani y cols., 2017)

Al igual que ocurre con otras arquitecturas de este tipo, la salida final además debe pasar por una transformación lineal y *softmax* para obtener una distribución de probabilidad sobre el vocabulario, esto permite determinar cuál es la palabra más probable como continuación de la secuencia de entrada.

2.4.2. Modelos BERT

Estos modelos fueron introducidos por el equipo de Google AI Language en el artículo: *"BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding"* (Devlin y cols., 2019). Desde la introducción de estos modelos hasta muy recientemente (en la actualidad en parte desplazados por el uso de los modelos generativos), han sido de los LLM mayormente utilizados gracias a su gran desempeño en las distintas tareas que conlleva el procesamiento del lenguaje natural. Su nombre en español significa "Representaciones bidireccionales de codificadores a partir de transformers". Esto hace referencia a la manera en la cual son preentrenados, condicionando desde el contexto izquierdo y derecho en conjunto dentro de todas sus capas, a partir de texto no etiquetado.

Su arquitectura se encuentra basada en la implementación original de Transformer:. Como fue descrito en (Devlin y cols., 2019) "*BERT's model architecture is a multi-layer bidirectional Transformer encoder based on the original implementation described in* Vaswani y cols. (2017)".

Inicialmente, se presenta con dos modelos base:

$$BERT_{BASE}(L = 12, H = 768, A = 12, \text{parámetros} = 110M) \tag{2.5}$$

$$BERT_{LARGE}(L = 24, H = 1024, A = 16, \text{parámetros} = 340M). \tag{2.6}$$

En donde L es la cantidad de bloques Transformer, H es la dimensión de la capa oculta y A es la cantidad de *attention heads*.

Esta familia de modelos es fácilmente adaptable a multitud de tareas mediante el agregado de una única capa de salida en el proceso de fine tuning. También son utilizados en la obtención de representaciones vectoriales de oraciones o textos, así como de *Word Embeddings* contextuales. Además de esto, si se comparan con los modelos de tipo GPT, los modelos BERT poseen un tamaño considerablemente menor. Todas estas características los convierten en candidatos ideales a la hora de elegir modelos base para realizar el proceso de *fine tuning* y su uso en general.

ROBERTA

Partiendo de la base de los modelos BERT presentados originalmente es que surge una optimización conocida como ROBERTA, presentada por Liu y cols. (2019).

El proceso ROBERTA (cuyo nombre en español significa "Enfoque de BERT Robustamente Optimizado") para el preentrenamiento de modelos de tipo BERT incluyó un conjunto de cambios, siendo los principales:

- Enmascarado dinámico: cada secuencia fue enmascarada de 10 formas distintas en lugar de ser enmascarada de una sola forma en todo el entrenamiento.
- Se elimina el objetivo "Predicción de la Siguiente Oración": en la presentación original del modelo BERT este es preentrenado haciendo uso de dos objetivos: modelado de lenguaje enmascarado y predicción de la siguiente oración. Durante las pruebas realizadas con distintos formatos de entrenamiento determinaron que quitar este segundo objetivo mantiene e incluso mejora levemente el rendimiento de las tareas posteriores.
- Grandes mini lotes: se amplía el tamaño de los minilotes utilizados, lo cual reduce la cantidad de pasos que realiza el preentrenamiento y su costo computacional.
- *Byte-Pair Encoding* a nivel de byte: se utilizan *bytes* en lugar de caracteres Unicode como las unidades que componen las sub palabras.

2.4.3. Modelos Llama

Son una familia de LLM de código abierto y de tipo generativos (es decir, a diferencia de los modelos bidireccionales como los mencionados anteriormente, estos modelos solamente condicionan de acuerdo al contexto izquierdo), creados por Meta AI y también basados en la arquitectura Transformer:. Inicialmente presentados en el artículo: *"LLaMA: Open and Efficient Foundation Language Models"* (Touvron y cols., 2023), estos modelos cuentan con dos versiones principales habilitadas al uso comercial: Llama 2 y Llama 3.

Dentro de cada versión además existen distintas alternativas según las cantidades de parámetros. En el artículo original se presentan modelos desde 6 mil millones hasta 65 mil millones.

Si bien los modelos de mayor cantidad de parámetros presentan un mejor desempeño, el costo computacional y de memoria necesaria es mucho mayor, por lo que, dependiendo de la complejidad de la tarea y los recursos disponibles, se pueden utilizar las alternativas con menos parámetros, ya que de todas formas permiten alcanzar buenos resultados.

En el desarrollo de la experimentación del presente trabajo se utilizará la versión de Llama 2 con 7B parámetros[2]

[2]https://huggingface.co/meta-llama/Llama-2-7b-chat-hf

Este modelo en particular corresponde al más pequeño del rango de modelos Llama 2 (los cuales van de 7 mil millones de parámetros hasta 70 mil millones). Estos modelos comparten en gran medida la arquitectura utilizada para los modelos Llama 1, a diferencia de un mayor tamaño de largo de contexto y el uso de *grouped-query attention* (una generalización de la atención multi consulta utilizada en los modelos tradicionales).

Inicialmente fue preentrenado utilizando un conjunto de 4 millones de tokens y posteriormente se le ha realizado fine tuning supervisado en conjunto con aprendizaje por refuerzos con retroalimentación humana (*Reinforcement Learning with Human Feedback* o RLHF) para la optimización de casos de uso de diálogos.

2.5. Fine tuning y prompting

Además de la creación de nuevos modelos desde cero con técnicas de preentrenamiento, como fue mencionado anteriormente, otras formas en las cuales se puede obtener los resultados deseados por parte de un modelo son mediante las técnicas de *fine tuning* o mediante *prompting*.

2.5.1. Fine tuning

Es habitual que, para realizar una determinada tarea, se desee partir de una base de conocimiento ya aprendido (mediante un preentrenamiento sobre un amplio corpus general) en lugar de realizar todo el aprendizaje desde cero. Para esto se comienza con un modelo base que ya ha sido preentrenado (también llamado *checkpoint* dentro del proceso) y se le continúa entrenando con nuevos datos relacionados con una tarea o dominio específico en el que se busca mejorar los resultados obtenidos por el modelo inicial. Esta técnica es la que se conoce como fine tuning.

Si bien siempre consiste en utilizar los nuevos datos para ajustar los parámetros en un paso posterior al *pretraining*, existen múltiples formas de realizarlo, algunos ejemplos de esto son: **continued pretraining**, donde se vuelve a entrenar todos los parámetros del modelo con los nuevos datos; **parameter-efficient fine tuning** o **PEFT**, donde se define solamente un subconjunto de parámetros para re entrenar mientras que los demás parámetros mantienen sus valores; y **supervised fine tuning (SFT)** o también llamado **instruction tuning**.(Jurafsky y Martin, 2024)

Esta última técnica es comúnmente empleada por los modelos LLM y consiste en entrenar sobre un dataset formado por preguntas y respuestas supervisadas, con la intención de que el modelo aprenda a devolver respuestas del formato que se desea. En particular será utilizada en este trabajo como se verá posteriormente.

2.5.2. Estrategias de prompting

En el ámbito del PLN, la tarea que se realiza habitualmente se conoce como *conditional generation* o generación condicional, esto refiere a que se toma un texto como entrada para el LLM (estos textos se conocen como prompt) para que luego el texto de salida generado por el modelo sea condicionado a esta entrada. (Jurafsky y Martin, 2024)

Esto permite adaptar un modelo ya preentrenado en otra tarea de similar dominio para realizar una nueva tarea de procesamiento del lenguaje natural sin la necesidad de realizar un proceso de *fine tuning*, ya que la tarea se debe realizar de forma implícita al predecir la salida más probable para el prompt ingresado; esto se conoce como *in-context learning* o "aprendizaje en contexto"(Brown y cols., 2020). En particular, como habitualmente los LLM se basan en arquitectura transformer, esto permite utilizar textos de entrada con suficiente cantidad de palabras para poder definir un contexto adecuado para la tarea a realizar.

Utilizar la estrategia correcta en los prompts que se le pasa al modelo marca la diferencia entre resultados satisfactorios o resultados no utilizables para la tarea que se quiere ejecutar. Sin embargo, según la complejidad de la tarea puede ser suficiente con una breve descripción para que el modelo realice las predicciones correctas que llevan al resultado esperado.

Las tres principales estrategias de aprendizaje en contexto son Few-shots, One-shot y Zero-shot basadas en la cantidad de ejemplos que se incluyen (Brown y cols., 2020).

Few-shots

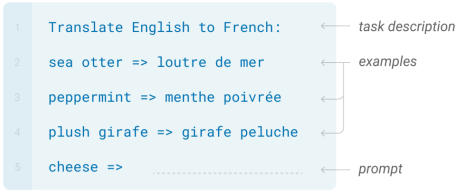

Figura 2.3: Ejemplo de prompt Few-shots (Brown y cols., 2020)

En el caso de la Figura 2.3, al momento de la inferencia se incluye como parte de la entrada una breve descripción en lenguaje natural de la tarea más un conjunto de ejemplos, donde para cada ejemplo se distingue la entrada y la salida. Finalmente, como última parte se incluye el caso del que se desea obtener una salida basada en la tarea. Comúnmente la cantidad de ejemplos que se proporcionan antes de la consulta se encuentra entre 10 y 100 (Brown y cols., 2020).

Figura 2.4: Ejemplo de prompt One-shot (Brown y cols., 2020)

One-shot

El caso que se observa en la Figura 2.4 es casi idéntico al anterior, a excepción de que solo se permite agregar un ejemplo previo. Si bien no contiene un conjunto de ejemplos como el caso anterior, un ejemplo en conjunto con la descripción en ocasiones puede ser suficiente, ya que provee una muestra de cómo se debería ver una salida para el tipo de entrada presentado.

Zero-shot

Figura 2.5: Ejemplo de prompt Zero-shot (Brown y cols., 2020)

Por último, en el caso de la Figura 2.5, solamente se incluye la descripción en lenguaje natural y luego la entrada a consultar. Esta estrategia si bien es muy limitada, puede ser suficiente para algunas tareas, principalmente si se trata de algo ya conocido que el modelo puede haber aprendido de forma implícita en su preentrenamiento (el ejemplo habitual, como se ve en la Figura 2.5 es el de traducir una oración, en tal caso el modelo "entiende" que lo que se busca es una oración que represente el mismo significado en el idioma especificado). En estos casos, además es crucial que la descripción en lenguaje natural provista sea lo menos ambigua posible, ya que no se provee de información adicional relativa a la salida esperada.

2.6. Modelos en Español

Los modelos del lenguaje pueden ser de un único lenguaje o multilingües, dependiendo del corpus sobre el cual han sido preentrenados. Si bien los resultados de los modelos multilingües en general han probado obtener buenos resultados, para tareas que solamente se realizan en un idioma en particular es habitual utilizar modelos específicos entrenados sobre estos, además de también acotar el dominio del corpus al que sea de interés para la tarea.

Dado que, en este trabajo el foco se encuentra en trabajar sobre textos de prensa uruguaya, se consideran dos modelos preentrenados en corpus de textos

en español. El modelo español más habitualmente utilizado, conocido como *BETO*[3] y el modelo creado específicamente para este tipo de textos, denominado *ROUBERTA* [4].

Ambos modelos son transformers de la familia de modelos BERT y entrenados con la técnica de *whole word masking* o "enmascarado de la palabra completa", donde la manera de entrenar al modelo es mediante el reemplazo de algunas palabras de cada entrada por un token especial (habitualmente *"<mask>"*) con el objetivo de entrenar el modelo para reconstruir la oración original (Cañete y cols., 2020).

2.6.1. BETO

Este modelo, presentado por (Cañete y cols., 2020), es considerado el principal modelo de BERT basado únicamente en textos en español. El tamaño del modelo es similar al modelo $\text{BERT}_{\text{BASE}}$ original tanto en dimensiones del modelo mismo como en tamaño del corpus de entrenamiento. Consiste en 12 capas *self-attention* y cada una de ellas consta de 16 *attention heads*, en total consta de 116 M parámetros.

BERT es un modelo *Open Source* entrenado únicamente utilizando textos en idioma español, extraídos de todo el contenido en español de *Wikipedia* y de un proyecto conocido como *OPUS*, el cual incluye charlas TED, publicaciones de las Naciones Unidas entre otros, alcanzando un corpus de aproximadamente 3 mil millones de palabras (un tamaño de aproximadamente 20 GiB de datos).

2.6.2. ROUBERTA

El modelo ROUBERTA, presentado por (Filevich y cols., 2024), ha sido entrenado con un conjunto de datos compuesto exclusivamente por textos de prensa uruguaya. Este corpus llamado UY22[5], contiene artículos de los principales medios del país (en el período entre 2000 y 2022) y cuenta con dos versiones: una versión inicial(6 GiB de datos) y una versión a la que se realizó un proceso de curado para mejorar la calidad (4 GiB de datos). En estos conjuntos de datos se incluye no solo el cuerpo de los artículos, sino además metadata asociada al mismo como puede ser la categoría, palabras clave o el título entre otros. ROUBERTA fue preentrenado haciendo uso de la versión de 4 GiB, con la arquitectura de RoBERTa-base (Liu y cols., 2019).

Si bien este modelo permite solamente entradas de un tamaño menor que BETO (128 frente a 512 palabras) y el corpus con el que ha sido entrenado es significativamente menor, es por este corpus de entrenamiento que el modelo se encuentra especializado en el tipo de documentos con los que se busca realizar la tarea planteada en el presente trabajo, tanto en la estructura de los textos pertenecientes a artículos de prensa, como en entidades o terminología propia de la región, lo que puede ayudar al análisis.

[3] https://huggingface.co/dccuchile/bert-base-spanish-wwm-cased
[4] https://huggingface.co/filevich/robertita-cased
[5] https://huggingface.co/datasets/pln-udelar/uy22

2.7. Métricas de evaluación

En el ámbito del PLN es habitual utilizar una serie de métricas que inicialmente se definen con base en la clasificación binaria y cuya definición se extiende a múltiples clases. En este sentido se comparan las etiquetas que devuelven los sistemas con las etiquetas definidas por humanos, también conocidas como *gold labels* (Jurafsky y Martin, 2024).

En clasificación binaria los elementos se evalúan como positivos y negativos, por lo tanto se definen las etiquetas como "*True Positives/Negatives*" (TP, TN) si estas etiquetas coinciden con las *gold labels*, y "*False Positives/Negatives*" (FP, FN) en caso contrario.

En base a esto es que se definen las siguientes métricas que utilizamos en el presente trabajo:

- accuracy: Porcentaje de evaluaciones correctas sobre el total de evaluaciones realizadas (Jurafsky y Martin, 2024)

$$\frac{TP + TN}{TP + FP + TN + FN} \tag{2.7}$$

- precision: Porcentaje de resultados evaluados por el sistema como positivos que realmente son positivos de acuerdo a las *gold labels*) (Jurafsky y Martin, 2024)

$$\frac{TP}{TP + FP} \tag{2.8}$$

- recall: Porcentaje de resultados positivos de acuerdo a las *gold labels* que fueron correctamente evaluados como positivos por el sistema (Jurafsky y Martin, 2024)

$$\frac{TP}{TP + FN} \tag{2.9}$$

- F1: Combinación de las métricas *precision* y *recall*. Definida como (Jurafsky y Martin, 2024):

$$\frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{2.10}$$

Estas métricas en la tarea particular de *Named Entity Recognition* son tradicionalmente calculadas considerando las entidades de forma exacta, comparando a nivel de *token*.

Para el análisis sobre el conjunto de testeo a mayor profundidad en este trabajo se hace uso de la librería *nerevaluate*[6]. Ha sido desarrollada particularmente para la evaluación de modelos que realizan la tarea de NER, ya que permite distinguir diferentes categorías de resultados en las predicciones haciendo uso de las categorías conocidas como MU-5 definidas por (Chinchor y Sundheim, 1993):

[6] https://pypi.org/project/nervaluate/

- Correcto (COR): Predicción coincide con la anotación de referencia
- Incorrecto (INC): Predicción no coincide en absoluto con la anotación de referencia
- Parcial (PAR): Predicción coincide pero no completamente con la anotación de referencia
- Faltante (MIS): Predicción no captura a la entidad en la anotación de referencia
- Espurio (SPU): Predicción no existe en la anotación de referencia

En base a estas categorías luego se separan distintos esquemas de evaluación de acuerdo a lo definido por SemEval'13(Jurgens y Klapaftis, 2013):

- Estricto : Coincidencia completa entre la cadena superficial de la entidad y el tipo
- Exacto : Concidencia completa entre la cadena superficial sin importar el tipo
- Parcial : Coincidencia parcial entre la cadena superficial sin importar el tipo
- Tipo : Se requiere alguna sobreposición entre la anotación de referencia y la entidad etiquetada en la predicción.

En el caso de este trabajo no es necesario tener en consideración el tipo de entidad, dado que solamente se cuenta con un único tipo de entidad: las fuentes de opinión. Sin embargo, se hace uso de esta librería para permitir comparar las métricas *precision*, *recall* y *F1* en un esquema exacto y parcial.

Para computar la precisión y *recall* en primer lugar se define lo siguiente:

$$POSSIBLE(POS) = COR + INC + PAR + MIS = TP + FN \tag{2.11}$$

$$ACTUAL(ACT) = COR + INC + PAR + SPU = TP + FP \tag{2.12}$$

A partir de estos valores es que se diferencian los cálculos de ambas métricas, en los casos donde se evalúan coincidencias completas se definen de la siguiente manera:

$$Precision = \frac{COR}{ACT} = \frac{TP}{(TP + FP)} \tag{2.13}$$

$$Recall = \frac{COR}{POS} = \frac{TP}{TP + FN} \tag{2.14}$$

Mientras que en los casos donde evalúan coincidencias parciales el cálculo se realiza como se muestra a continuación:

$$Precision = \frac{COR + 0{,}5 \times PAR}{ACT} = \frac{TP}{TP + FP} \tag{2.15}$$

$$Recall = \frac{COR + 0{,}5 \times PAR}{POS} = \frac{COR}{ACT} = \frac{TP}{TP + FN} \tag{2.16}$$

Luego el cálculo de la medida F1 se hace de forma habitual utilizando las métricas anteriores correspondientes.

2.8. Trabajos relacionados

En esta sección se resumen estudios previos relacionados con la presente tarea y los datos utilizados, así como estudios sobre otros dominios pero relacionados en objetivos y estrategias utilizadas para su resolución.

2.8.1. Identificación de opiniones de diferentes fuentes en textos en español

En esta tesis de doctorado realizada por Aiala Rosá (2011), se tiene como objetivo el reconocimiento de manera automática de opiniones en textos de prensa uruguaya, incluyendo además del reconocimiento de opinión completa, la identificación de las distintas partes que las componen (predicado, fuente, asunto y mensaje).

Con este fin se creó un modelo que permite obtener las distintas partes que conforman una opinión, un léxico de predicados de opinión asociado al modelo anterior, dos corpus de datos de 13 mil y 40 mil palabras (uno anotado con opiniones y sus componentes, y otro con predicados de opinión y sus fuentes respectivamente) y tres sistemas para el reconocimiento.

Un primer sistema utilizado para las distintas partes de la opinión (haciendo uso de la técnica conocida como *reglas contextuales*), un segundo sistema realizado únicamente para reconocimientos de fuentes de opinión empleando la técnica de *conditional random fields (CRF)* y un tercer sistema también realizado para el reconocimiento de fuentes de opinión que utiliza ambas técnicas mencionadas.

Los resultados generados por el primer sistema, analizando el valor de la métrica F1, que solamente hace uso de reglas contextuales y fue realizado para el reconocimiento de las distintas partes de la opinión (considerando en la evaluación a reconocimientos parciales como correctos) fueron de 92 % predicado de opinión, 81 % fuente, 75 % asunto, 89 % para el mensaje y 85 % para la opinión completa.

En lo relativo a las fuentes en particular, considerando una medida exacta (es decir, solo se toma correcto si las palabras consideradas fuente de opinión son exactamente las mismas que en la respuesta esperada) para el caso del primer sistema se obtuvo un resultado de F1 de 79 %, mientras que en el sistema que solamente utiliza CRF se obtuvo 76 % y en el sistema que combinó ambas técnicas se obtuvo una medida exacta de 83 %. Con lo que se observa que la combinación de aplicar CRF sobre la salida que se obtiene del sistema de reglas resulta en un potenciamiento de las ventajas que poseen ambos enfoques.

2.8.2. A Language Model Trained on Uruguayan Spanish News Text

Realizado por Filevich y cols. (2024), en este trabajo se presenta un nuevo modelo de lenguaje entrenado desde cero utilizando únicamente textos de prensa uruguaya, para lo cual también se debió crear el corpus obtenido mediante *web scraping* (práctica para recolectar datos existentes en internet sin ser directamente a través de una API provista del sitio, que involucra, entre otros, el análisis de datos, parseo del lenguaje natural y seguridad de la información (Herrera, 2019)) de los sitios de varios de los principales medios de prensa del país (a excepción del diario "La Diaria" quien proporcionó sus propios artículos para este fin).

Para crear el modelo de lenguaje se utilizó una arquitectura basada en ROBERTA entrenado solamente con la versión limpia del corpus antes mencionado, resultando en el modelo ROUBERTA comentado en la sección 2.6.2. Este LLM fue luego comparado con BETO (modelo BERT específicamente entrenado para el lenguaje español), BERT multilenguaje y XML-RoBERTa en las tareas de PLN conocidas como "respuesta a preguntas" y "análisis de sentimientos" sobre nuevos artículos provenientes de prensa uruguaya.

Como resultado se obtuvo un valor de 75 % *accuracy* en la tarea de análisis de sentimientos, donde BETO obtuvo 74.6 % y XML-RoBERTa 73.6 %. Para la tarea de respuesta a preguntas, ROUBERTA obtuvo un valor de 32.3 % en la medida F1 y 28.1 % *exact match*, donde BETO obtuvo 29.4 % y 24.6 % , y XML-RoBERTa 36.4 % y 26.8 % respectivamente. Donde el nuevo modelo obtuvo un mejor desempeño en ambas tareas (a excepción de la métrica F1 para respuesta a preguntas) a pesar de contar con recursos significativamente menores en su creación.

2.8.3. Dataset Construction and Opinion Holder Detection Using Pre-trained Models

En este artículo presentado en 2022, Al-Mahmud y Shimada presentan la tarea de obtener de manera automática fuentes de opinión, para lo cual se la considera dividida en dos grandes pasos: detección e identificación. Esto se realiza sobre un dataset construido a partir de ASTE-DATA-V2 (xuuuluuu, 2019), el mismo fue preparado tomando como base la preparación realizada para la construcción del dataset SemEval 2014-2016(Webber, Cohn, He, y Liu, 2020).

El dominio particular sobre el cual se realizó el trabajo fue el de reseñas de laptops, constituido por 1453 oraciones de opinión, cada una de ellas incluye una lista de triplas de la forma "[(posición objetivo, posición de la opinión, sentimiento ('POS' o 'NEG'))]". Para conformar el corpus con la información necesaria por cada oración, se agregó una etiqueta binaria para la detección de fuentes de opinión "INSIDE/OUTSIDE" y un etiquetado de formato BIO para la identificación de la fuente dentro de la oración. Esto resultó en 37 % de las oraciones de tipo INSIDE y 63 % de tipo outside, con la presencia de un total

de 971 fuentes de opinión.

El paso de la detección realiza el análisis de la existencia o ausencia de una fuente de opinión en la oración, utilizando la clasificación binaria "INSIDE/OUTSIDE". Para esto utilizan dos versiones del modelo BERT: BERT con regresión logística y DistilBERT(Sanh, Debut, Chaumond, y Wolf, 2020) también con regresión logística.

Dentro del segundo paso el objetivo es identificar dentro de las oraciones su etiquetado BIO, para realizar esta tarea utilizan dos tipos de arquitecturas: *Feature-based* (basado en características) y *Fine-tuning based*. En ambos casos se consideran para los *embeddings* combinaciones entre BERT, DistilBERT y CSE (*Contextual String Embeddings*), mientras que luego se utiliza CRF (Conditional Random Fields) para la tarea de etiquetado de secuencias en todos los casos menos uno donde solamente se utiliza el modelo BERT para todo el proceso.

Los resultados reportados en este caso indican los mejores desempeños en medida F1 para los casos (DistilBERT & CSE) + CRF con 94.53 % (tanto *feature based* como *fine-tuning based* produjeron el mismo valor, aunque el modelo *feature based* tuvo menor tiempo de procesamiento con 2 minutos 2 segundos). Sin embargo, en términos del modelo con el menor tiempo de procesamiento de todos los evaluados, el mejor performante fue el modelo *feature based* CSE + CRF (con un tiempo de 49 segundos) el cual obtuvo una medida F1 de 94.47 %.

2.8.4. Improved Opinion Role Labelling in Parliamentary Debates

El objetivo de este trabajo realizado por Bamberg, Rehbein, y Ponzetto (2022), consiste en obtener un modelo del lenguaje sobre el idioma alemán, que permita realizar la tarea de *Etiquetado de Opiniones* u *Opinion Role Labelling (ORL)*. Esta tarea trata de identificar fuentes (*holders*) y objetos (*targets*) de las opiniones.

Para esto se utiliza la arquitectura transformer, entrenando un modelo a partir de un dataset basado en los creados para IGGSA-STEPS 2014 (Ruppenhofer y cols., 2014) y IGGSA-STEPS 2016(Ruppenhofer, Struß, y Wiegand, 2016). La tarea es modelada como *token classification* y hace uso de las etiquetas BIO para la identificación de ambos componentes de las opiniones.

Como parte del trabajo, además se evalúa si es posible mejorar los resultados obtenidos mediante la transferencia de conocimiento, haciendo uso de la técnica de *fine tuning* sobre un modelo entrenado para la tarea de *Semantic Role Labelling*, para lo cual se entrena un modelo basado en BERT para SRL con datos del subconjunto en idioma alemán de CoNLL-2009(Hajič y cols., 2009). Se compara realizar el *fine tuning* luego de completar el entrenamiento inicial, con realizarlo luego de que la curva de la medida exacta F1 se comienza a aplanar.

Los mejores resultados fueron obtenidos al realizar el preentrenado para SRL y posterior *fine tuning* cuando la curva de F1 comenzó a aplanarse, donde se obtuvo en promedio una medida F1 de 67.2 % para fuentes y 54.6 % para objetos de las opiniones.

Capítulo 3

Definición del problema

Los artículos de prensa escrita habitualmente incluyen citas directas o paráfrasis de opiniones dadas por los principales agentes de la noticia. Este trabajo consiste en la detección e identificación de las fuentes de opinión.

Partiendo de un corpus generado originalmente por Rosá como parte de la tesis de doctorado mencionada en la sección 2.8.1, se utiliza este recurso para realizar *fine tuning* de instrucciones sobre modelos BETO y ROUBERTA con el fin de realizar la tarea de clasificar cada token en una oración de entrada con una etiqueta de formato BIO relativa a las fuentes de opinión.

Se comparan instancias cambiando parte de sus hiperparámetros y de los dos mejores performantes en la etapa de entrenamiento, luego se comparan sus resultados sobre el conjunto de testeo.

Además se evalúan los resultados de utilizar la técnica de *prompting* sobre el modelo existente Llama 2, utilizando estrategias *Zero-shot*, *One-shot* y *Few-shots*.

3.1. Herramientas empleadas

Se trabajó utilizando lenguaje *Python* en la herramienta Google Colaboratory, donde el entorno de ejecución empleado fue el de GPU del back-end de Google Compute Engine en Python 3, así como un conjunto de librerías provistas por la plataforma *Hugging Face*[1] para la carga y trabajo con los modelos de lenguaje. Esta plataforma permite el acceso a una extensa colección de modelos de lenguaje, corpus de datos y aplicaciones de demostración, además de permitir la publicación de datos propios y el acceso a contenido de aprendizaje sobre distintas partes del procesamiento del lenguaje natural.

Además se utilizó la librería *pandas*[2] para la manipulación de datos, así como

[1] https://huggingface.co/
[2] https://pandas.pydata.org/

sklearn[3] y *nerevaluate*[4] para la obtención de las diferentes métricas evaluadas entre otras.

3.2. Corpus utilizado

Este conjunto de datos presentado en (Rosá, 2011) se divide originalmente en entrenamiento y testeo, con 30.000 palabras y 10.000 palabras respectivamente (dentro del conjunto de entrenamiento además luego se separa un 10 % para validación de forma pseudoaleatoria). Se cuenta con 425, 65 y 162 fuentes de opinión para las particiones de entrenamiento, validación y testeo respectivamente.

Cada artículo de prensa se encuentra identificado por una línea inicial con el número de documento procesado, luego se encuentra separado a nivel de oraciones, cada una separada por una línea en blanco. Los bloques de líneas que componen una oración cuentan con una fila por palabra, donde en el corpus original se incluyen además de la palabra misma, un grupo de atributos separados por espacios. Para la experimentación realizada a lo largo de este trabajo se considera únicamente la palabra y la salida esperada (es decir, el etiquetado en formato BIO para las fuentes de opinión).

A modo ilustrativo, en la Figura 3.1 se presentan varios ejemplos de oraciones extraídas del conjunto de testeo.

Dado que es habitual, para las tareas cuyo enfoque se basa en redes neuronales profundas, trabajar directamente sobre texto en lenguaje natural como entrada sin otras consideraciones adicionales, se optó por realizar las pruebas sobre los modelos solamente haciendo uso de las palabras originales y sus etiquetas de salida verdaderas.

3.3. Etapas del proceso

En esta sección se incluyen los diferentes bloques que componen la solución al problema descrito anteriormente.

- **Preprocesamiento del corpus de datos**: Los archivos que contienen el corpus presentado en la sección 3.2, tanto de testeo como de entrenamiento, se procesaron para mantener únicamente la información relevante y se pasaron a formato *csv* de forma de obtener los recursos necesarios para la posterior creación de los conjuntos de datos.

- **Obtención de modelos**: Mediante la utilización de los métodos provistos por la librería de *Hugging Face* se cargaron los modelos preentrenados en español BETO, ROUBERTA y Llama 2 vistos anteriormente, incluyendo sus tokenizadores. Se optó por utilizar las versiones *case sensitive* (es decir que distinguen mayúsculas de minúsculas) en el caso de BETO y

[3] https://scikit-learn.org/stable/
[4] https://pypi.org/project/nervaluate/

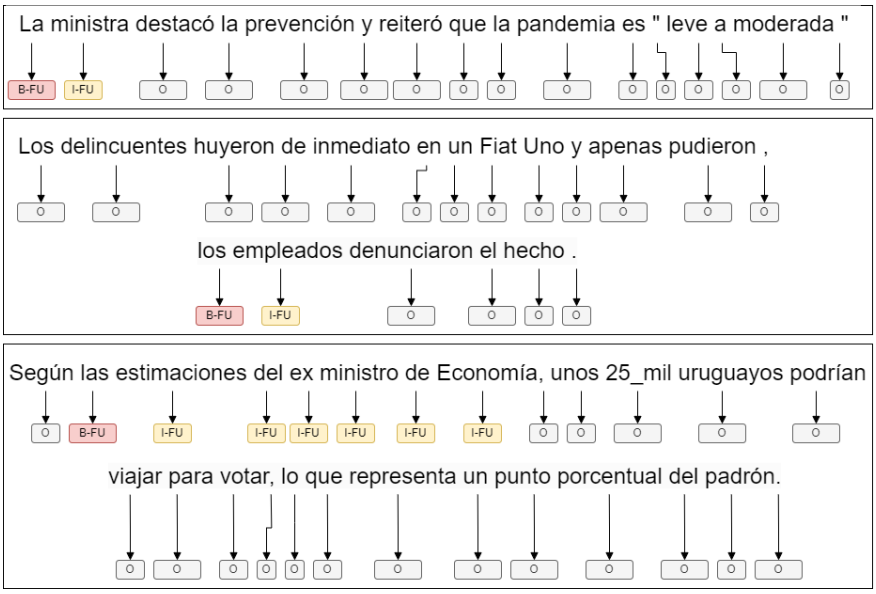

Figura 3.1: Ejemplos de oraciones y etiquetas de salida extraídas del corpus de entrenamiento

ROUBERTA dado que esta distinción es una característica particular que diferencia a los nombres propios, y en el caso de las fuentes de opinión en muchas ocasiones son nombres propios.

- **Creación de datasets y *fine tuning* sobre modelos BETO y ROUBERTA**: Utilizando las librerías antes descritas se ajustan los conjuntos de datos y etiquetas de salida de acuerdo a los distintos tokenizadores para la creación de los datasets a utilizar en la realización de *fine tuning*. Esto implica tokenizar las secuencias, adaptar las etiquetas a estos nuevos arreglos y entrenar a los modelos con los datasets modificados. Se realiza este proceso con distintas combinaciones de parte de sus hiperparámetros para ambos modelos preentrenados y se registran los valores obtenidos por las métricas tradicionales (*accuracy*, *precision*, *recall* y *F1*) cada un número de pasos fijo durante estos procesos.

- **Comparación de métricas durante el entrenamiento**: Los resultados registrados en la etapa anterior se comparan para las distintas métricas obtenidas por las diferentes combinaciones de hiperparámetros con sus resultados finales. Además para los cuatro modelos con mejor desempeño (dos basados en BETO y dos basados en ROUBERTA) se evalúa la evolución de esta métrica a lo largo del tiempo de entrenamiento.

- **Evaluación detallada de los principales modelos de fine tuning**: Los dos modelos de mejores resultados en la etapa anterior (un modelo

basado en BETO y otro modelo basado en ROUBERTA) se comparan utilizando las métricas obtenidas sobre el conjunto de testeo, evaluando tanto de manera estricta como evaluando coincidencias parciales. Además se analizan las diferencias en los errores más usualmente cometidos por ambos modelos, analizando manualmente las oraciones donde se produjeron errores.

- ***Prompting* sobre modelo Llama 2**: Se realizan inferencias con base en un conjunto de *prompts* definidos siguiendo las distintas estrategias introducidas en la sección 2.5.2, empleando un formato de *chat* como lo requiere el modelo.

- **Evaluación de los resultados obtenidos durante el prompting**: Se analizan los resultados obtenidos comparando las respuestas con las soluciones esperadas. Esto se realiza tanto a nivel de comprensión del formato de respuesta que se espera obtener, como a nivel de correctitud de los elementos que componen a la fuente de opinión para las oraciones pertenecientes al conjunto de testeo proporcionadas en los *prompts*.

Capítulo 4

Implementación y Experimentación

En este capítulo se presentan las distintas tareas que conformaron la implementación de los experimentos realizados en el presente trabajo. Se consideran tres bloques principales: preprocesamiento, *fine tuning* y *prompting*.

4.1. Preparación de los datos

```
La el D 0 0 S F 0 O B-FU
organización organización N C 0 S F 0 O I-FU
de de S 0 0 0 0 0 O I-FU
defensa defensa N C 0 S F 0 O I-FU
de de S 0 0 0 0 0 O I-FU
los el D 0 0 P M 0 O I-FU
derechos_humanos derecho_humano N C 0 P M 0 O I-FU
Amnistía_Internacional amnistía_internacional N P 0 0 0 0 B-FU-R I-FU
afirmó afirmar V 0 Op S 0 0 O O
que que C 0 0 0 0 0 O O
esta este D 0 0 S F 0 O O
condena condena N C 0 S F 0 O O
" " F 0 0 0 0 0 O O
vergonzosa vergonzoso A 0 0 S F 0 O O
" " F 0 0 0 0 0 O O
es ser V 0 S S 0 0 O O
una uno D 0 0 S F 0 O O
" " F 0 0 0 0 0 O O
mascarada mascarada N C 0 S F 0 O O
política político A 0 0 S F 0 O O
" " F 0 0 0 0 0 O O
y y C 0 0 0 0 0 O O
Human_Right_Watch human_right_watch N P 0 0 0 0 0 O B-FU
( ( F 0 0 0 0 0 O I-FU
HWR hwr N P 0 0 0 0 0 O I-FU
) ) F 0 0 0 0 0 O I-FU
```

```
criticó criticar V 0 Op S 0 0 O O
el el D 0 0 S M 0 O O
" " F 0 0 0 0 0 O O
abuso abuso N C 0 S M 0 O O
de de S 0 0 0 0 0 O O
poder poder N C 0 S M 0 O O
" " F 0 0 0 0 0 O O
del del S 0 0 0 0 0 O O
gobierno_militar gobierno_militar N C 0 S M 0 O O
de de S 0 0 0 0 0 O O
Birmania birmania N P 0 0 0 0 0 O O
. . F 0 0 0 0 0 O O
```

Figura 4.1: Texto en formato original del archivo *trainFUR*

Como se ha mencionado anteriormente en la sección 3.2, los datos que se utilizaron cuentan con un conjunto adicional de atributos por cada palabra que no serán utilizados en el presente trabajo. Se puede observar el formato original

en el ejemplo de la Figura 4.1. Antes de la creación del *dateset* es necesario tanto filtrar únicamente la información de interés para la tarea, como adaptarla a un formato comprensible por las siguientes etapas del proceso. En este caso se buscó obtener un archivo *csv* compuesto por dos columnas: la primera contiene cada oración y la segunda contiene los valores de las etiquetas reales. De esta forma se evitan los conflictos relativos a posibles caracteres separadores siendo utilizados dentro de las oraciones.

En esta etapa se utiliza la librería *re*[1] para el manejo de expresiones regulares, con la cual en primer lugar se separan los documentos enteros en oraciones, aprovechando que se encuentran divididas por líneas donde solamente existen espacios vacíos. También se descartan aquellas líneas que corresponden al cabezal de cada documento, que solo contienen el número de textos a los que pertenecen las oraciones.

Luego por cada oración, donde cada palabra es una línea, se obtiene de cada una de estas líneas solamente la primera columna (es decir, la palabra real que contenía el texto) y la última columna (asociada a la salida esperada). Para las salidas se realiza un mapeo entre O, I-FU y B-FU con los valores 0, 1 y 2 respectivamente.

4.1.1. Creación del dataset

Una vez se tuvieron los datos preparados, se utilizó la librería *sklearn*[2] para separar dentro del conjunto de datos de entrenamiento un subconjunto asociado a la validación. Se definió un tamaño del 10 % del conjunto de entrenamiento destinado para este fin, haciendo uso de una semilla para mantener la reproductibilidad.

Se utilizaron los módulos *csv* y *os* de la librería estándar para agregar en un archivo *train.csv*, *validation.csv* o *test.csv*, según corresponda, cada oración y sus valores de etiquetas separados por una coma.

Luego haciendo uso de la librería *datasets*[3] se cargó en una única variable los 3 conjuntos de datos a partir de estos archivos, donde primero se dividen las oraciones y las etiquetas en arreglos de elementos, ya que como se mencionó, fueron almacenados como una única cadena de caracteres cada uno.

Completada esta etapa se tiene una estructura con los datos separados en subconjuntos, cada uno contiene únicamente las palabras y etiquetas asociadas. La cantidad de secuencias que contienen los subconjuntos de entrenamiento, validación y testeo es de 1224, 136 y 436 respectivamente. La estructura constituye los datos "crudos" (es decir, contiene las oraciones separadas a nivel de palabras y las etiquetas esperadas), por lo que no es directamente utilizable por los distintos modelos. Para poder ser utilizados por cada modelo deben pasar sus correspondientes tokenizadores y volver a alinear las etiquetas, que aquí se encuentran a nivel de palabras, con los tokens generados.

[1] https://github.com/python/cpython/tree/3.13/Lib/re/
[2] https://pypi.org/project/scikit-learn/
[3] https://pypi.org/project/datasets/

4.2. Entrenamiento de modelos con fine tuning

Durante toda esta etapa se trabaja en simultáneo con los modelos que parten del modelo preentrenado ROUBERTA y aquellos que parten de BETO, vistos en la sección 2.6. Utilizamos la librería *transformers* tanto para cargar los modelos existentes desde la plataforma de *Hugging Face* como para realizar el entrenamiento con los nuevos datasets y sus posteriores inferencias.

4.2.1. Modelos y tokenizadores

```
ORIGINAL
['La', 'organización', 'de', 'defensa', 'de', 'los',
'derechos_humanos', 'Amnistía_Internacional', 'afirmó', 'que',
'esta', 'condena', '"', 'vergonzosa', '"', 'es', 'una', '"',
'mascarada', 'política', '"', 'y', 'Human_Right_Watch', '(', 'HWR',
')', 'criticó', 'el', '"', 'abuso', 'de', 'poder', '"', 'del',
'gobierno_militar', 'de', 'Birmania', '.']

TOKENS ROUBERTA:
 ['<s>', 'ĠLa', 'ĠorganizaciÃ³n', 'Ġde', 'Ġdefensa', 'Ġde', 'Ġlos',
'Ġderechos', '_', 'hu', 'manos', 'ĠAmnistÃĺa', '_', 'Inter',
'nacional', 'ĠafirmÃ³', 'Ġque', 'Ġesta', 'Ġcondena', 'Ġ"',
'Ġvergonz', 'osa', 'Ġ"', 'Ġes', 'Ġuna', 'Ġ"', 'Ġmas', 'car', 'ada',
'ĠpolÃĺtica', 'Ġ"', 'Ġy', 'ĠHuman', '_', 'R', 'ight', '_', 'W',
'atch', 'Ġ(', 'ĠH', 'W', 'R', 'Ġ)', 'ĠcriticÃ³', 'Ġel', 'Ġ"',
'Ġabuso', 'Ġde', 'Ġpoder', 'Ġ"', 'Ġdel', 'Ġgobierno', '_',
'militar', 'Ġde', 'ĠBir', 'mania', 'Ġ.', '</s>']

TOKENS BETO:
 ['[CLS]', 'La', 'organización', 'de', 'defensa', 'de', 'los',
'derechos', '_', 'humanos', 'Am', '##nistía', '_', 'Internacional',
'afirmó', 'que', 'esta', 'condena', '"', 'vergon', '##zosa', '"',
'es', 'una', '"', 'mas', '##cara', '##da', 'política', '"', 'y',
'Human', '_', 'Rig', '##ht', '_', 'W', '##atch', '(', 'H', '##W',
'##R', ')', 'critic', '##ó', 'el', '"', 'abuso', 'de', 'poder', '"',
'del', 'gobierno', '_', 'militar', 'de', 'Birmania', '.', '[SEP]']
```

Figura 4.2: Ejemplo del resultado obtenido por ambos algoritmos de tokenización sobre una oración perteneciente al conjunto de entrenamiento

Para cargar los modelos preentrenados se hizo uso de los métodos en las clases automáticas (*Auto Classes* de *Hugging Face*, hacen uso del nombre o de la ruta definida para decidir el tipo de modelo/ *tokenizer* necesario (*Auto Classes*, s.f.)).

Como se necesita trabajar con modelos de clasificación de secuencias, se invocó el método *AutoModelForTokenClassification*, el cual necesitó la ubicación del modelo en la plataforma, cantidad de etiquetas y los mapeos entre etiquetas y valores numéricos en ambas direcciones.

De forma similar se obtuvieron los tokenizadores mediante el método *AutoTokenizer*, el cual requiere especificar el tamaño de secuencia máximo que permite el modelo, si corresponde truncado y *padding* (es decir, recortar las oraciones que exceden el tamaño máximo y completar las oraciones de tamaño inferior con tokens que serán no considerados por los entrenamientos, para de esta forma permitir tener un tamaño uniforme de entradas, ambos verdaderos en este trabajo), así como si corresponde agregar un espacio de prefijo. Este último parámetro también se definió como verdadero, dado que permite evitar que los modelos interpreten de manera distinta a un token que se encuentra en el inicio de una oración en lugar de cualquier otra posición.

En la Figura 4.3 se observa el caso de una oración perteneciente al conjunto de entrenamiento y cómo se descompone en *tokens* con ambos tokenizadores[4]. Luego de pasar por estos algoritmos de tokenización como se puede observar el tamaño de las secuencias cambia y las etiquetas que se encontraban a nivel de palabra ya no coinciden, es por esto que se implementa un algoritmo para volver a alinear las etiquetas (ahora a nivel de *token*) con aquellas que existían a nivel de palabra.

4.2.2. Alineado de etiquetas

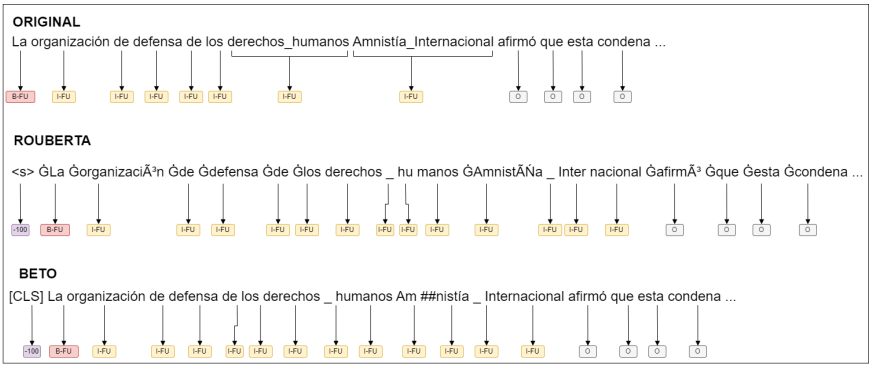

Figura 4.3: Ejemplo del resultado obtenido al alinear las etiquetas originales a los nuevos *tokens* para un fragmento de oración perteneciente al conjunto de entrenamiento

Se implementó un método que recibe las etiquetas como se encuentran en el dataset, y los *word identites* devueltos por el tokenizador (es decir, los identificadores de las palabras a las que pertenece cada token dentro de la oración) y con esta información los *tokens* correspondientes a una misma palabra heredan la etiqueta que tenía la palabra originalmente, teniendo en cuenta la salvedad de las palabras que originalmente se correspondían a la etiqueta B-FU, dado

[4]Se aprecia el uso de caracteres especiales para identificar si el *token* es el primero de una palabra, así como *tokens* adicionales que denotan el inicio y fin de las oraciones.

que la misma ahora solo le corresponde al primer token asociado a esa palabra, mientras que los demás pasan a estar asociados a la etiqueta I-FU en su lugar. Los *tokens* agregados para delimitar inicio y fin de las oraciones (así como *paddings* que fueron incluidos como se mencionó anteriormente) tendrán etiqueta de valor -100, con lo que el modelo las ignorará.

Luego de realizado este alineado, se guarda en un nuevo dataset (por modelo, debido a que ROUBERTA y BETO poseen sus propios tokenizadores) por cada oración la secuencia de *tokens* como entrada, y como salida esperada la secuencia de valores asociados a las etiquetas de cada uno de ellos.

4.2.3. Fine tuning

Una vez cargados los modelos y sus tokenizadores, así como versiones del conjunto de datos adaptados a cada tokenizador, se debió instanciar los argumentos y los entrenadores necesarios para realizar la tarea de *fine tuning*. Nuevamente se hizo uso de la librería *transformers*.

Para esto, por cada versión se debió definir los argumentos de entrenamiento, donde se especifican los mismos hiperparámetros para todos los modelos utilizados, los cuales fueron los siguientes:

- per_device_train_batch_size = 8
- per_device_eval_batch_size = 8
- num_train_epochs = 10
- eval_strategy = "steps"
- logging_strategy = "steps"
- logging_steps = 100

Los valores de *batch size* (o tamaño de lote) por dispositivo, se definieron relativamente pequeños para evitar problemas de sobrecarga en la memoria de la GPU durante el entrenamiento y evaluación. La cantidad de *epochs* en principio se evaluó como parámetro variable, sin embargo, los resultados no eran consistentes, por lo que se optó por un valor intermedio. Los hiperparámetros *eval strategy* y *logging strategy* se utilizaron para registrar los valores obtenidos por las métricas de evaluación cada una cantidad fija de pasos, definida por *logging steps*. Por último *save strategy* define cada cuánto se deben guardar checkpoints del modelo durante el entrenamiento, en este caso fue definido como "no" dado que solamente se deseaba almacenar el resultado final una vez completado el aprendizaje.

Además de estos hiperparámetros que se mantuvieron para todos los modelos entrenados, se decidió evaluar si modificando los valores de otros que también ofrece la librería se podría obtener un mejor desempeño. Con el fin de comparar posibles soluciones se consideraron variantes de los siguientes tres argumentos:

- ***Weight Decay*** (Caída de peso): Caída de pesos a aplicar. Es una técnica de regularización utilizada para evitar el sobre ajuste en redes neuronales. Se consideraron los valores 0, 0.02 y 0.05.
- ***Learning Rate*** (tasa de aprendizaj): Tasa de aprendizaje para Adam. Se consideraron los valores 2e-5, 5e-5 y 10e-5.

- ***Warmup Steps*** (Pasos de calentamiento): Número de pasos de calentamiento a ser aplicados de forma lineal de 0 al valor de *leaning rate*[5]. Se consideraron los valores 0, 50 y 100.

Debido a que se comparan tres opciones por cada parámetro mencionado, esto resulta en un total de 27 modelos (por cada modelo base). Las diferencias entre los entrenadores para modelos con base en ROUBERTA solamente radican en los modelos basados en BETO por sus tokenizadores, lo que implica distintos *datasets* de la sección anterior así como distintos *data collators* (métodos mediante los cuales se convierten listas de elementos de los *datasets* de entrenamiento y evaluación a lotes(Hugging Face, 2024)).

En el entrenador es donde además se define cómo computar las métricas y qué hacer con la información obtenida. En el caso de este trabajo para el cómputo se definió un método que obtiene las predicciones y *golden labels*, el que luego obtiene los valores de precision, recall, F1, y accuracy haciendo uso de la librería *sklearn*. Una vez obtenidas las métricas de ese paso se registran los valores en un archivo *csv*.

Se registró también el tamaño de ambos modelos así como el tiempo que tomaron en entrenamiento.

4.2.4. Comparación con métricas estrictas durante el entrenamiento

Luego del entrenamiento comparamos los valores de las métricas antes mencionadas obtenidos en el último paso de cada entrenamiento. En la Figura 4.4 se observan los valores de la medida F1. Solamente se incluyeron aquellos resultados cuyos valores fueron mayores a cero, en los otros casos lo que ocurrió fue que no hubo predicciones de entidades en los ejemplos predichos por el modelo con lo que el valor de la métrica es cero, esto se puede deber a que el modelo no haya llegado a la convergencia por los pesos iniciales o por los valores de los hiperparámetros en específico, entre otros motivos.

Salvo algunas excepciones que se ubican por debajo, los resultados se encuentran en su mayoría ubicados dentro del rango definido por 70 % y 85 %. En particular se puede apreciar que los cuatro modelos con mejor desempeño en esta etapa son BETO 17, BETO 4, ROUBERTA 3 y ROUBERTA 21. A continuación se presentan los valores de los hiperparámetros específicos a cada una de estas versiones:

Identificador	Weight Decay	Learning Rate	Warmup Steps
BETO 17	0.02	10e-5	100
BETO 4	0	5e-5	50
ROUBERTA 3	0	5e-5	0
ROUBERTA 21	0.05	5e-5	0

[5]https://huggingface.co/transformers/v3.0.2/main_classes/trainer.html#transformers.TrainingArguments

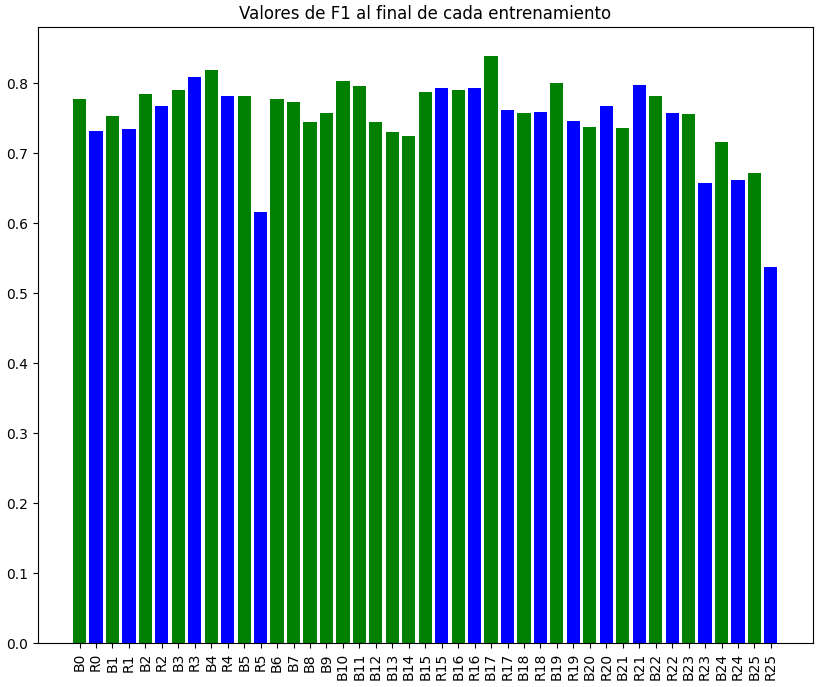

Figura 4.4: Comparación de los valores de F1 en el último paso de los preentrenamientos (base BETO en verde y base ROUBERTA en azul), solo se incluyen aquellos con valor mayor a 0

Considerando ahora estos cuatro modelos es que se decidió analizar la evolución de todos los valores de las métricas obtenidas a lo largo del proceso de entrenamiento, esto se aprecia en la Figura 4.5.

Como se pudo observar, la evolución de estos valores no se realiza de manera gradual ya que en ocasiones incluye una fluctuación importante. Sin embargo, a medida que se realiza una mayor cantidad de pasos los valores obtenidos tienden a subir.

Se observa además que, si bien considerando en promedio ambos pares de modelos, los modelos basados en BETO presentaron un mejor valor final de las métricas estrictas observadas, el comportamiento general se encuentra bastante parejo entre los cuatro modelos. El modelo BETO 17 en particular tuvo una mejor evolución de la precisión lo que llevó a mejores resultados de F1.

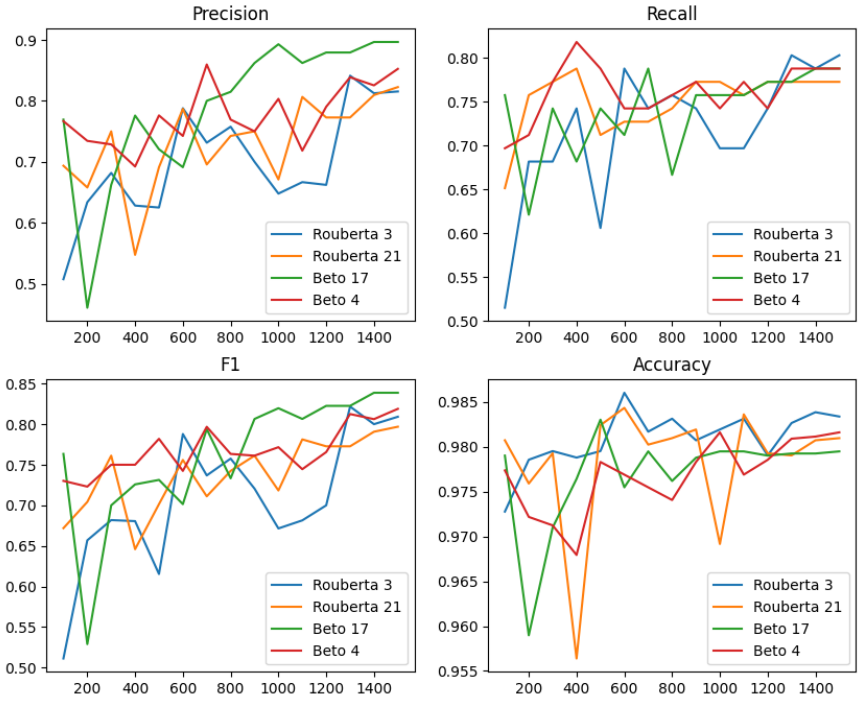

Figura 4.5: Comparación de la evolución de los valores para las métricas obtenidas en el entrenamiento, evaluadas cada 100 pasos.

En resumen, si comparamos los valores finales para cada métrica sobre el conjunto de evaluación, los resultados fueron los siguientes:

Identificador	Precision	Recall	F1	Accuracy
BETO 17	**0.896**	0.789	**0.839**	0.979
BETO 4	0.852	0.789	0.819	0.982
ROUBERTA 3	0.815	**0.803**	0.809	**0.983**
ROUBERTA 21	0.822	0.773	0.797	0.981

Tabla 4.2.4: Métricas exactas evaluadas sobre datos de evaluación para los cuatro mejores modelos.

Donde se aprecia que en el caso de *Recall* y *Accuracy* el mejor resultado fue obtenido por ROUBERTA 3, mientras que en *Precision* y *F1* BETO 17 obtuvo los mejores resultados.

4.2.5. Obtención y evaluación de inferencias en conjunto de testeo

Una vez analizados los resultados obtenidos durante el entrenamiento se decidió limitar las pruebas a dos modelos. Un modelo resultante de hacer *fine*

tuning sobre el modelo preentrenado BETO y otro sobre la base del modelo preentrenado ROUBERTA. Para esto se optó por los modelos que en la etapa anterior obtuvieron los mejores resultados: BETO 17 y ROUBERTA 3.

Para poder realizar las inferencias se definieron métodos para obtener la predicción y alinearlas a etiquetas a nivel de palabras (debido a que las predicciones realizadas por los modelos fueron obtenidas a nivel de *token*).

El primer método utiliza la librería *PyTorch*[6] para convertir a tensores los *input_ids* y *attention mask* (este último se obtiene del tokenizador). Esto es lo que utiliza el modelo para realizar las predicciones de todo el conjunto de testeo.

Las predicciones generadas luego se alinean a nivel de palabras de forma similar e inversa al método descrito en la sección 4.2.2, dado que en esta ocasión en lugar de obtener las etiquetas para cada *token* a partir de la etiqueta que posee la palabra, se tienen las etiquetas a nivel de *token* y se desea obtener las etiquetas a nivel de palabra.

Finalmente se creó un método para realizar la comparación entre las etiquetas reales y aquellas predichas por estos modelos, almacenando los resultados en un archivo *csv* por cada modelo. Además de hacer uso de la librería *nerevaluate*, presentada en la sección 2.7, para obtener un mayor nivel de detalle sobre los resultados obtenidos en el conjunto de testeo.

4.2.6. Comparación de métricas estrictas y parciales de los dos modelos mejor performantes sobre conjunto de testeo

```
Utilizando ROUBERTA
           correct  incorrect  partial  missed  spurious  possible  actual  \
ent_type       141          0        0      21        30       162     171
partial        135          0        6      21        30       162     171
strict         135          6        0      21        30       162     171
exact          135          6        0      21        30       162     171

           Precisión  Recuperación  F1-score
ent_type    0.824561      0.870370  0.846847
partial     0.807018      0.851852  0.828829
strict      0.789474      0.833333  0.810811
exact       0.789474      0.833333  0.810811
--------------------------------------------------------------------------------
Utilizando BETO
           correct  incorrect  partial  missed  spurious  possible  actual  \
ent_type       130          0        0      32        29       162     159
partial        121          0        9      32        29       162     159
strict         121          9        0      32        29       162     159
exact          121          9        0      32        29       162     159

           Precisión  Recuperación  F1-score
ent_type    0.817610      0.802469  0.809969
partial     0.789308      0.774691  0.781931
strict      0.761006      0.746914  0.753894
exact       0.761006      0.746914  0.753894
```

Figura 4.6: Resultados de la evaluación sobre el conjunto de testeo haciendo uso de la librería *nerevaluate*

Por el contrario de lo que se pudo observar con el conjunto de evaluación,

[6] https://pypi.org/project/torch/

en el caso del conjunto de testeo los mejores resultados, tanto si se consideran las métricas exactas como parciales, corresponden al modelo ROUBERTA.

En particular, este modelo obtuvo resultados incorrectos en las respuestas al comparar con respecto a las *gold labels* en 47 de las 436 oraciones que conforman el *dataset* de testeo, mientras que BETO 17 tuvo diferencias en 60 oraciones.

Esto se ve reflejado en las métricas obtenidas, las cuales se observan a en la Figura 4.6. En particular vemos que el valor de F1 exacta para el modelo ROUBERTA fue de 81.1 %, mientras que para el modelo BETO fue de 75.4 %. Por otro lado, si se consideran además las entidades parcialmente correctas se obtuvo una medida F1 de 82.9 % y 78.2 %, finalmente en el caso más permisivo donde se toma una entidad como válida solo con existir alguna superposición entre la fuente definida por la respuesta esperada y aquella predicha, los valores son 85 % para ROUBERTA y 81 % para BETO.

Además, si analizamos la información a nivel de entidades que también se encuentra en la Figura 4.6, se puede apreciar que el conjunto de testeo contaba con 162 fuentes de opinión, donde el modelo ROUBERTA falló en identificar menos cantidad de entidades que el modelo BETO (21 y 32 respectivamente). Dando como resultado 121 fuentes correctamente identificadas por el modelo BETO y 135 por el modelo ROUBERTA.

Si comparamos entre los dos modelos las oraciones en las cuales las respuestas fueron incorrectas, vemos que en gran parte de los casos ambos modelos fallaron en las mismas oraciones. En particular de los 47 y 60 casos con errores reportados por los modelos, 32 ocurrieron sobre las mismas entradas.

Esto no implica que en esos casos el error haya sido siempre el mismo, como podemos apreciar en la Figura 4.7.

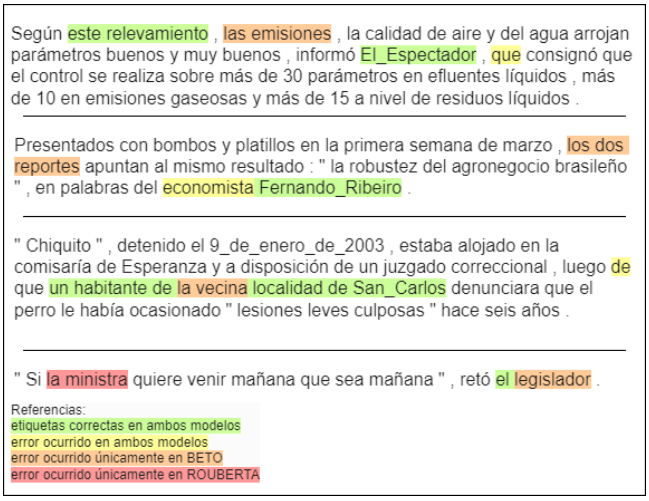

Figura 4.7: Ejemplo comparación de errores obtenidos por ambos modelos sobre oración de testeo

En términos generales se observó que para el modelo ROUBERTA los errores

producidos habitualmente son errores donde no se encuentran fuentes (es decir se etiquetan como *Outside*) mientras que el modelo BETO en los resultados obtenidos fue más propenso a considerar fuentes que en realidad no lo son, como en el caso de los ejemplos antes vistos.

4.3. Obtención modelo Llama y pipeline de inferencias

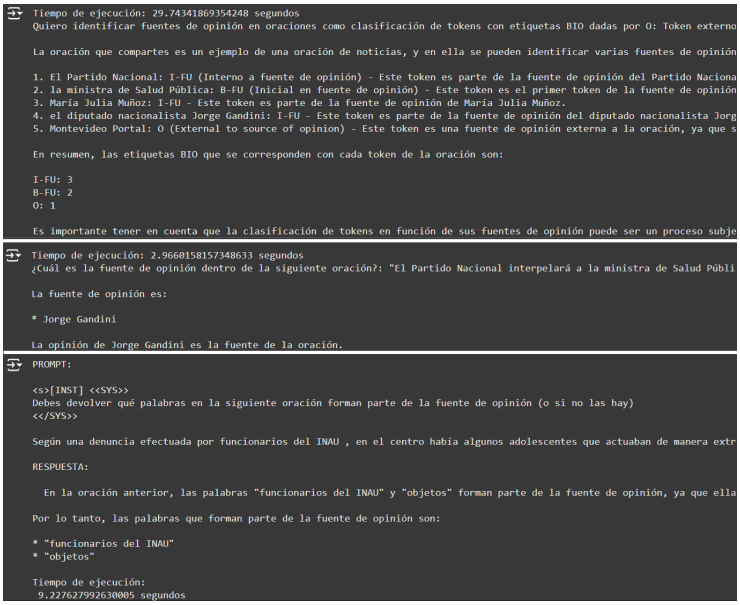

Figura 4.8: Ejemplo salidas obtenidas con la estrategia *Zero shot*

El modelo utilizado en esta instancia es el modelo Llama 2 con 7B parámetros, introducido en la sección 2.4.3. Para poder hacer uso de este modelo se utiliza nuevamente la librería *transformer*, en particular los métodos *AutoTokenizer* y *pipeline*.

Para la instancia del tokenizador fue necesario definir el modelo y el tamaño del contexto[7] (se utilizó un tamaño de 2048 *tokens*). Luego el *pipeline* se definió de tipo *text-generation* y se incluyó un método que solamente recibe un prompt e imprime el texto generado por este pipeline. En el anexo .1 se incluyen las salidas completas obtenidas en estos ejemplos.

Siguiendo la estrategia Zero-shot se trata de explicar en términos de lenguaje natural la tarea a realizar y el formato esperado, sin proveer de ningún ejemplo previo al que se debe responder. Los resultados mostraron que con los *prompts*

[7]además se debió incluir una clave de *Hugging Face* dado que el modelo requiere solicitar permiso de uso con un usuario en esta plataforma

utilizados el modelo no llegaba a comprender la tarea a realizar. Además se decidió realizar una prueba utilizando un formato específico para los modelos de chat basado en código utilizado por uno de los laboratorios del curso Análisis de Textos (InCo - FIng), edición 2024. Nuevamente, se utilizó una estrategia *Zero-shot*, resultando en una respuesta similar. En la Figura 4.8 se aprecian los resultados de algunos ejemplos.

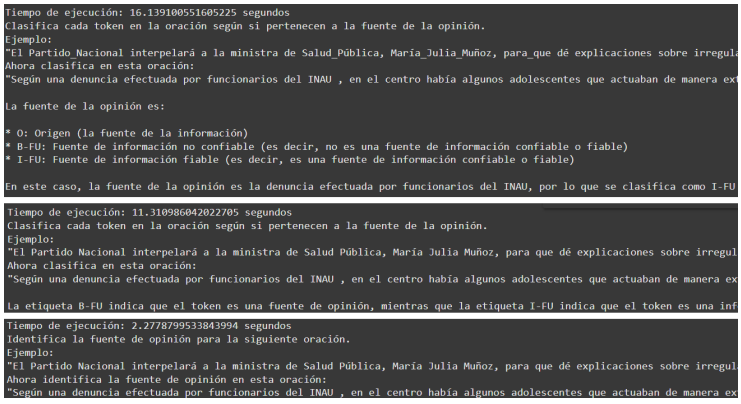

Figura 4.9: Ejemplo salidas obtenidas con la estrategia *One-shot*

Se pudo observar en las respuestas que el modelo en ocasiones alcanza a comprender lo que corresponde a la fuente de opinión, cuando se le consulta directamente, sin embargo cuando se solicita un formato este no lo respeta.

Luego, con los casos de One-shot y Few-shots se utiliza una definición más breve de la tarea para dar paso a algunos ejemplos pertenecientes al conjunto de entrenamiento, a modo de presentar el formato de respuesta utilizado.

Durante las pruebas realizadas no se observó una clara diferencia entre utilizar uno o más ejemplos, dado que los resultados con ambas estrategias fueron muy similares. Sí fue posible observar que fue suficiente con la estrategia *One-shot* para que el modelo se enfocara en las fuentes de opinión (a diferencia del caso *Zero-shot* donde ocurrió, por ejemplo, que se considerara la tarea a realizar como generación de un cuestionario con preguntas similares).

En estos casos de uno o varios ejemplos, si bien las respuestas en algunas de las pruebas incluyeron una salida con el formato especificado, la cantidad de elementos que contenían las secuencias de etiquetas obtenidas no se correspondía con la cantidad de palabras incluidas en la oración de entrada. A modo de ejemplo si tomamos la respuesta obtenida del último caso observable en la captura 4.10, la parte de esta respuesta que fue generada por el modelo (es decir, sin incluir que en la respuesta primero se repite el prompt de entrada) es la siguiente:

-> [O,O,O,B-FU,I-FU,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O]
La fuente de opinión es "una denuncia efectuada por funcionarios del INAU".

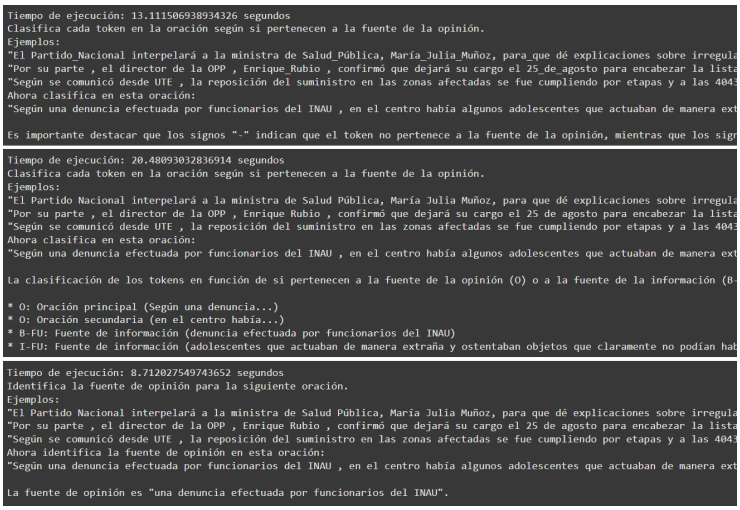

Figura 4.10: Ejemplo salidas obtenidas con la estrategia *few shots*

En este caso es correcta la respuesta en lenguaje natural, ya que la fuente de opinión es realmente "una denuncia efectuada por funcionarios del INAU". Sin embargo, las etiquetas provistas no se corresponden con esta respuesta, dado que el arreglo devuelto no contiene la misma cantidad de etiquetas que de palabras que se incluyen en la oración de entrada y que, además, en este caso particular la fuente de opinión consiste en siete palabras, pero las etiquetas en la respuesta solo son una B-FU y una I-FU. Es decir que no se encuentra correctamente representado.

Esto demuestra que el modelo si bien posee un mayor entendimiento del formato de la tarea planteada a base de los ejemplos, no es suficiente para que realice correctamente la asociación entre las etiquetas y las palabras.

Finalmente en términos del tiempo de ejecución que tomó realizar las inferencias, sin importar la estrategia utilizada, fue inconsistente y heterogéneo. En ocasiones el sistema tardaba varios minutos en responder antes de fallar por haber hecho uso de toda la memoria disponible. En los casos cuando sí se obtuvo respuesta, los tiempos se ubicaron en el rango entre 2.3 y 29.7 segundos. Cuando respondió en una cantidad menor a 5 segundos en general solo respondía el mismo *prompt* más una pregunta en lugar de realizar lo planteado. Aunque la mayoría de las respuestas se obtuvieron entre 15 y 20 segundos.

4.4. Resumen de experimentos realizados

A lo largo de este trabajo, se evaluaron diferentes técnicas para resolver el problema de extracción de fuentes de opinión, modelando el problema como una tarea de clasificación de secuencias.

Se probó un primer enfoque basado en hacer *fine tuning* sobre los mode-

los preentrenados BETO y ROUBERTA, para lo cual se evaluaron en etapa de entrenamiento diferentes valores de algunos hiperparámetros, resultando en 27 modelos partiendo del modelo preentrenado BETO y 27 basados en ROUBERTA. Se comparó el valor de la medida F1 final de los modelos luego de este entrenamiento, los resultados positivos se observan en la Figura 4.4. De los cuatro modelos con mejores valores en esta etapa, se compararon a su vez los valores de las métricas exactas *precision*, *recall* y *accuracy* como se observa en la Tabla 4.2.4.

En general este enfoque produjo buenos resultados, comparables a aquellos obtenidos por sistemas anteriores. En particular, al comparar estos resultados con los obtenidos por parte de los sistemas del trabajo realizado por (Rosá, 2011), observamos que, si bien nuestro sistema no contaba con la información adicional de poder asumir que ya estaban identificados los predicados de opinión al contrario de los sistemas mencionados, nuestros mejores resultados de medida F exacta se ubicaron en un rango intermedio entre los valores obtenidos por sus sistemas. A continuación se detalla la comparación de esta métrica sobre el conjunto de testeo:

Estrategia	Medida F Exacta
Sistema de Reglas Contextuales + CRF	83 %
Fine tuning basado en ROUBERTA	**81.1 %**
Sistema de Reglas Contextuales	79 %
CRF	76 %
Fine tuning basado en BETO	**75.4 %**

En la Figura 4.6, se incluye además el detalle de la cantidad de predicciones correctas e incorrectas de ambos modelos resultantes de *fine tuning*. Se observó que, de las 162 fuentes de opinión existentes en el conjunto de testeo, fueron identificadas correctamente 121 fuentes por el modelo basado en BETO y 135 por el modelo basado en ROUBERTA.

Por otro lado, se probó utilizar el modelo Llama 2 para realizar inferencias mediante distintos *prompts*, haciendo uso de las estrategias *zero-shot*, *one-shot* y *few-shots*. Se realizaron pruebas de distintos *prompts* sobre un par de oraciones elegidas al azar del conjunto de entrenamiento a modo de evaluar la diferencia en las instrucciones provistas y la cantidad de otros ejemplos que se incluyeron como ayuda a la tarea.

Este enfoque basado en *prompting* no mostró resultados satisfactorios dado que, como se observa en el anexo .1, las salidas obtenidas no dan como resultado una predicción con el formato solicitado, o cuando el formato en principio se asemeja a los solicitados, no es correcto. Por lo tanto este enfoque no fue evaluado posteriormente en el conjunto de testeo.

Capítulo 5

Conclusiones y trabajo futuro

El objetivo propuesto en el presente trabajo consistió en utilizar las técnicas de *fine tuning* sobre modelos de lenguaje de arquitectura Transformer para la tarea de obtención automática de fuentes de opinión, haciendo uso de dos modelos preentrenados sobre textos en lenguaje español: BETO, entrenado con un gran corpus general del español, y ROUBERTA, entrenado con un corpus mucho menor de textos de prensa uruguaya.

Fue posible obtener las fuentes de opinión modelando el problema como clasificación de secuencias, realizando la tarea de *fine tuning* para ajustar los modelos. Ambos modelos preentrenados utilizados permitieron un buen desempeño tanto en métricas estrictas como en métricas parciales sobre el conjunto de testeo.

Durante este trabajo no fue posible obtener resultados con el formato deseado mediante *prompting* con un modelo de lenguaje de tipo generativo (Llama). En términos de responder la posible fuente de opinión cuando se pregunta a modo de chat los resultados son variados y poco predecibles.

En conclusión la estrategia de realizar *fine tuning* sobre los modelos mencionados para realizar esta tarea ha probado que es posible obtener buenos resultados. Estos valores han sido similares a aquellos resultados obtenidos anteriormente al utilizar otros métodos como CRF y sistemas de reglas (Rosá, 2011). Se obtuvo, en el mejor caso, una medida F1 exacta de 81.1 %, levemente por debajo del resultado presentado por el sistema utilizado en el trabajo antes mencionado, esto a pesar de que nuestro sistema, a diferencia del anterior, no se encuentra bajo la suposición de que ya se han identificado los predicados de opinión (lo cual sería de ayuda al modelo al momento de predecir).

Si, por otro lado, comparamos con los resultados vistos en otros trabajos como en el caso de (Bamberg y cols., 2022) y (Al-Mahmud y Shimada, 2022), vemos que el valor de 81.1 % para la medida F1 obtenida por nuestro modelo se encuentra en un rango intermedio entre un valor de 67.2 %, obtenido en

el trabajo realizado por Bamberg et al. sobre textos en lenguaje alemán de debates parlamentarios, y 94.5 %, obtenido por Al-Mahmud et al. sobre reseñas en idioma inglés. Si bien esta comparación no es completamente válida, dado que los conjuntos de datos y sus idiomas son diferentes, la tarea era la misma por lo que también nos permite ver resultados esperables en una visión un poco más global.

A futuro se podría emplear alguno de los modelos presentados para que sus inferencias formen parte de un sistema general de procesamiento automático de noticias de prensa. Se podría también ampliar las clases en los datos de entrenamiento para incluir otros elementos de la opinión como lo son el mensaje y asunto, así como ampliar su entrenamiento con un conjunto mayor de ejemplos.

Teniendo en cuenta que del conjunto original de datos se utilizó únicamente las palabras y etiquetas esperadas, se podría evaluar si se obtienen mejores resultados al hacer uso del amplio conjunto de atributos adicionales.

Glosario de Términos

Accuracy: Métrica de evaluación que consiste en el porcentaje de evaluaciones correctas sobre el total de evaluaciones realizadas.(Jurafsky y Martin, 2024). 17, 31

BIO (Etiquetado BIO): Formato para etiquetar palabras o *tokens* en clasificación de secuencias, donde se distingue el primer elemento de una entidad (etiqueta B *Beginning*), de los demás elementos que las componen (etiqueta I *Inside*), así como los elementos externos a cualquier entidad (etiqueta O *Outside*). 6

F1: Métrica de evaluación que consiste en la combinación de las métricas *precision* y *recall*. Definida como $\frac{2\times\text{precision}\times\text{recall}}{\text{precision}+\text{recall}}$.(Jurafsky y Martin, 2024). 17, 31

Few-shots: Estrategia de prompting donde se incluye, además de una instrucción de la tarea, un grupo limitado de ejemplos de entradas y salidas. 14, 37

Fine tuning: Técnica que consiste en continuar entrenando un LLM ya preentrenado por un nuevo conjunto de ejemplos asociados a una tarea o dominio específicos, con el fin de evitar la necesidad de entrenar un modelo desde cero. 11, 13

LLM (Large Language Models): Modelos que predicen la palabra más probable como continuación de una secuencia de entrada. Se entrenan a partir de un gran conjunto de textos. 7

One-shot: Estrategia de prompting donde se incluye, además de una instrucción de la tarea, un solo ejemplo de entrada y salida. 14, 37

PLN (Procesamiento del Lenguaje Natural): Campo de la inteligencia artificial cuyo objetivo es lograr que las computadoras realicen tareas útiles relacionadas con el lenguaje humano, como facilitar la comunicación entre humanos y máquinas, mejorar la comunicación entre personas o simplemente llevar a cabo un procesamiento útil de texto o habla.(Jurafsky y Martin, 2009). 5

Precision: Métrica de evaluación que consiste en el porcentaje de resultados evaluados por el sistema como positivos que realmente son positivos de acuerdo al resultado esperado (etiquetas definidas por humanos, también llamados *gold labels*) (Jurafsky y Martin, 2024). 17, 31

Prompt: Texto que se utiliza como entrada para la tarea de generación condicional en LLM. 14

Recall: Métrica de evaluación que consiste en el procentaje de resultados positivos de acuerdo a las *gold labels*(etiquetas definidas por humanos, también llamados *gold labels*) que fueron correctamente evaluados como positivos por el sistema(Jurafsky y Martin, 2024). 17, 31

Token: Salidas resultantes de pasar una palabra o secuencia de palabras por un algoritmo de tokenización como puede ser *Byte-Pair Encoding*. 8

Transformer: Arquitectura de redes neuronales utilizada para LLM, basada en mecanismos de atención y presentada incialmente en (Vaswani y cols., 2017). 7, 11, 12

Zero-shot: Estrategia de prompting donde solamente se incluye una instrucción de la tarea a realizar. 14, 36

Referencias

Al-Mahmud, N., y Shimada, K. (2022, 7). Dataset construction and classification based on pre-trained models for opinion holder detection. *2022 12th International Congress on Advanced Applied Informatics (IIAI-AAI)*. Descargado de https://doi.org/10.1109/iiaiaai55812.2022.00023 doi: 10.1109/iiaiaai55812.2022.00023

Auto Classes. (s.f.). Descargado de https://huggingface.co/docs/transformers/model_doc/auto

Bamberg, L., Rehbein, I., y Ponzetto, S. (2022, 12–15 septiembre). Improved opinion role labelling in parliamentary debates. En R. Schaefer, X. Bai, M. Stede, y T. Zesch (Eds.), *Proceedings of the 18th conference on natural language processing (konvens 2022)* (pp. 110–120). Potsdam, Germany: KONVENS 2022 Organizers. Descargado de https://aclanthology.org/2022.konvens-1.13

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... Amodei, D. (2020, 5). *Language Models are Few-Shot Learners.* Descargado de https://arxiv.org/abs/2005.14165

Cañete, J., Chaperon, G., Fuentes, R., Ho, J.-H., Kang, H., y Pérez, J. (2020). Spanish pre-trained bert model and evaluation data. En *Pml4dc at iclr 2020.*

Chinchor, N., y Sundheim, B. (1993). MUC-5 evaluation metrics. En *Fifth message understanding conference (MUC-5): Proceedings of a conference held in baltimore, Maryland, august 25-27, 1993.* Descargado de https://aclanthology.org/M93-1007

Devlin, J., Chang, M.-W., Lee, K., y Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. En *North american chapter of the association for computational linguistics.* Descargado de https://api.semanticscholar.org/CorpusID:52967399

Filevich, J. P., Marco, G., Castro, S., Chiruzzo, L., y Rosa, A. (2024, mayo). A language model trained on uruguayan spanish news text. En F. Gaspari y cols. (Eds.), *Proceedings of the second international workshop towards digital language equality (tdle): Focusing on sustainability @ lrec-coling 2024* (pp. 53–60). Torino, Italia: ELRA and ICCL. Descargado de https://aclanthology.org/2024.tdle-1.5

Hajič, J., Ciaramita, M., Johansson, R., Kawahara, D., Martí, M. A., Màrquez, L., ... Zhang, Y. (2009, junio). The CoNLL-2009 shared task: Syn-

tactic and semantic dependencies in multiple languages. En J. Hajič (Ed.), *Proceedings of the thirteenth conference on computational natural language learning (CoNLL 2009): Shared task* (pp. 1–18). Boulder, Colorado: Association for Computational Linguistics. Descargado de https://aclanthology.org/W09-1201

Herrera, A. J. (2019, 12). Ryan Mitchell Web Scraping with Python COLLECTING MORE DATA FROM THE MODERN WEB. *itesm*. Descargado de https://www.academia.edu/41461428/Ryan_Mitchell_Web_Scraping_with_Python_COLLECTING_MORE_DATA_FROM_THE_MODERN_WEB

Hochreiter, S., y Schmidhuber, J. (1997, 12). Long short-term memory. *Neural computation*, *9*, 1735-80. doi: 10.1162/neco.1997.9.8.1735

Hugging Face. (2024). Transformers documentation: Trainer class [Manual de software informático]. Descargado 2024-10-29, de https://huggingface.co/docs/transformers/v4.46.0/en/main_classes/trainer#api-reference (https://huggingface.co/docs/transformers/v4.46.0/en/main_classes/trainer#api-reference)

Jurafsky, D., y Martin, J. H. (2009). *Speech and language processing (2nd edition)*. USA: Prentice-Hall, Inc.

Jurafsky, D., y Martin, J. H. (2024). *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition with language models* (3rd ed.). Descargado de https://web.stanford.edu/~jurafsky/slp3/ (Online manuscript released August 20, 2024)

Jurgens, D., y Klapaftis, I. (2013, junio). SemEval-2013 task 13: Word sense induction for graded and non-graded senses. En S. Manandhar y D. Yuret (Eds.), *Second joint conference on lexical and computational semantics (*SEM), volume 2: Proceedings of the seventh international workshop on semantic evaluation (SemEval 2013)* (pp. 290–299). Atlanta, Georgia, USA: Association for Computational Linguistics. Descargado de https://aclanthology.org/S13-2049

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., ... Stoyanov, V. (2019, 1). ROBERTA: A robustly optimized BERT pretraining approach. *arXiv (Cornell University)*. Descargado de https://arxiv.org/abs/1907.11692 doi: 10.48550/arxiv.1907.11692

Manning, C. D., Raghavan, P., y Schütze, H. (2008). *Introduction to information retrieval*.

Ramshaw, L., y Marcus, M. (1995). Text chunking using transformation-based learning. En *Third workshop on very large corpora*. Descargado de https://aclanthology.org/W95-0107

Rosá, A. (2011). *Identificación de opiniones de diferentes fuentes en textos en español* (Tesis Doctoral no publicada). Facultad de Ingeniería, Universidad de la República, Montevideo, Uruguay.

Ruppenhofer, J., Struß, J. M., Sonntag, J., y Gindl, S. (2014, 7). IGGSA-STEPS: Shared Task on Source and Target Extraction from Political Speeches. *Deleted Journal*, *29*(1), 33–46. Descargado de https://doi.org/10.21248/jlcl.29.2014.182 doi: 10.21248/jlcl.29.2014.182

Ruppenhofer, J., Struß, J. M., y Wiegand, M. (2016, 1). Overview of the IGGSA 2016 Shared Task on Source and Target Extraction from Political Speeches. *Proceedings of the IGGSA Shared Task 2016 Workshop*, 1–9. Descargado de https://ids-pub.bsz-bw.de/files/5508/Ruppenhofer_Struss_Wiegand_Overview_of_the_IGGSA_2016.pdf

Sanh, V., Debut, L., Chaumond, J., y Wolf, T. (2020). *Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter.* Descargado de https://arxiv.org/abs/1910.01108

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., ... Lample, G. (2023, 2). *LLAMA: Open and Efficient Foundation Language Models.* Descargado de https://arxiv.org/abs/2302.13971

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017, 6). Attention is All you Need. *arXiv (Cornell University)*, *30*, 5998–6008. Descargado de https://arxiv.org/pdf/1706.03762v5

Webber, B., Cohn, T., He, Y., y Liu, Y. (Eds.). (2020, noviembre). *Proceedings of the 2020 conference on empirical methods in natural language processing (emnlp).* Online: Association for Computational Linguistics. Descargado de https://aclanthology.org/2020.emnlp-main.0

xuuuluuu. (2019, 11). *SemEval-Triplet-data.* Descargado de https://github.com/xuuuluuu/SemEval-Triplet-data/tree/master/ASTE-Data-V2-EMNLP2020

Anexos

.1. Salidas completas obtenidas por el modelo Llama2

Prompt	Salida
Clasifica cada token en la oración según si pertenecen a la fuente de la opinión.\nEjemplos:\n"El Partido_Nacional interpelará a la ministra de Salud_Pública, María_Julia_Muñoz, para_que dé explicaciones sobre irregularidades en el manejo de ASSE, según informó a Montevideo_Portal el diputado nacionalista Jorge_Gandini." -> [O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,B-FU,I-FU,I-FU,I-FU]\n"Por su parte , el director de la OPP , Enrique_Rubio , confirmó que dejará su cargo el 25_de_agosto para encabezar la lista al senado de la Vertiente_Artiguista seguido de Juan_Faroppa y Carmen_Beramendi ." -> [O,O,O,O,B-FU,I-FU,I-FU,I-FU,I-FU,I-FU,I-FU,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O]\n"Según se comunicó desde UTE , la reposición del suministro en las zonas afectadas se fue cumpliendo por etapas y a las 40433 horas quedó completamente restablecido ." -> [O,O,O,B-FU,I-FU,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O]\nAhora clasifica en esta oración:\n"Según una denuncia efectuada por funcionarios del INAU , en el centro había algunos adolescentes que actuaban de manera extraña y ostentaban objetos que claramente no podían haber adquirido"	Clasifica cada token en la oración según si pertenecen a la fuente de la opinión.\nEjemplos:\n"El Partido_Nacional interpelará a la ministra de Salud_Pública, María_Julia_Muñoz, para_que dé explicaciones sobre irregularidades en el manejo de ASSE, según informó a Montevideo_Portal el diputado nacionalista Jorge_Gandini." -> [O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,B-FU,I-FU,I-FU,I-FU]\n"Por su parte , el director de la OPP , Enrique_Rubio , confirmó que dejará su cargo el 25_de_agosto para encabezar la lista al senado de la Vertiente_Artiguista seguido de Juan_Faroppa y Carmen_Beramendi ." -> [O,O,O,O,B-FU,I-FU,I-FU,I-FU,I-FU,I-FU,I-FU,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O]\n"Según se comunicó desde UTE , la reposición del suministro en las zonas afectadas se fue cumpliendo por etapas y a las 40433 horas quedó completamente restablecido ." -> [O,O,O,B-FU,I-FU,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O]\nAhora clasifica en esta oración:\n"Según una denuncia efectuada por funcionarios del INAU , en el centro había algunos adolescentes que actuaban de manera extraña y ostentaban objetos que claramente no podían haber adquirido" -> [O,O,O,B-FU,I-FU,I-FU,I-FU,I-FU,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O]\n\nEn este caso, la fuente de la opinión es "una denuncia efectuada por funcionarios del INAU". Por lo tanto, todos los tokens que se encuentran en la oración corresponden a la fuente de la opinión, excepto "adolescentes" que es un token de la fuente de información.
Clasifica cada token en la oración según si pertenecen a la fuente de la opinión.\nEjemplos:\n"El Partido Nacional interpelará a la ministra de Salud Pública, María Julia Muñoz, para que dé explicaciones sobre irregularidades en el manejo de ASSE, según informó a Montevideo Portal el diputado nacionalista Jorge Gandini." -> [O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,B-FU,I-FU,I-FU,I-FU,I-FU]\n"Por su parte , el director de la OPP , Enrique Rubio , confirmó que dejará su cargo el 25 de agosto para encabezar la lista al senado de la Vertiente Artiguista seguido de Juan Faroppa y Carmen Beramendi ." -> [O,O,O,O,B-FU,I-FU,I-FU,I-FU,I-FU,I-FU,I-FU,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O]\n"Según se comunicó desde UTE , la reposición del suministro en las zonas afectadas se fue cumpliendo por etapas y a las 40433 horas quedó completamente restablecido ." -> [O,O,O,B-FU,I-FU,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O]\nAhora clasifica en esta oración:\n"Según una denuncia efectuada por funcionarios del INAU , en el centro había algunos adolescentes que actuaban de manera extraña y ostentaban objetos que claramente no podían haber adquirido"	Clasifica cada token en la oración según si pertenecen a la fuente de la opinión.\nEjemplos:\n"El Partido Nacional interpelará a la ministra de Salud Pública, María Julia Muñoz, para que dé explicaciones sobre irregularidades en el manejo de ASSE, según informó a Montevideo Portal el diputado nacionalista Jorge Gandini." -> [O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,B-FU,I-FU,I-FU,I-FU,I-FU]\n"Por su parte , el director de la OPP , Enrique Rubio , confirmó que dejará su cargo el 25 de agosto para encabezar la lista al senado de la Vertiente Artiguista seguido de Juan Faroppa y Carmen Beramendi ." -> [O,O,O,O,B-FU,I-FU,I-FU,I-FU,I-FU,I-FU,I-FU,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O]\n"Según se comunicó desde UTE , la reposición del suministro en las zonas afectadas se fue cumpliendo por etapas y a las 40433 horas quedó completamente restablecido ." -> [O,O,O,B-FU,I-FU,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O]\nAhora clasifica en esta oración:\n"Según una denuncia efectuada por funcionarios del INAU , en el centro había algunos adolescentes que actuaban de manera extraña y ostentaban objetos que claramente no podían haber adquirido" -> [O,O,O,O,B-FU,I-FU,I-FU,I-FU,I-FU,I-FU,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O]\nEn este caso, la fuente de la opinión es "una denuncia efectuada por funcionarios del INAU".\n\nEste es solo un ejemplo, pero puedes aplicar este sistema a cualquier oración que quieras clasificar.
Identifica la fuente de opinión para la siguiente oración.\nEjemplos:\n"El Partido Nacional interpelará a la ministra de Salud Pública, María Julia Muñoz, para que dé explicaciones sobre irregularidades en el manejo de ASSE, según informó a Montevideo Portal el diputado nacionalista Jorge Gandini." -> "el diputado nacionalista Jorge Gandini"\n"Por su parte , el director de la OPP , Enrique Rubio , confirmó que dejará su cargo el 25 de agosto para encabezar la lista al senado de la Vertiente Artiguista seguido de Juan Faroppa y Carmen Beramendi ." -> [O,O,O,O,B-FU,I-FU,I-FU,I-FU,I-FU,I-FU,I-FU,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O]\n"Según se comunicó desde UTE , la reposición del suministro en las zonas afectadas se fue cumpliendo por etapas y a las 40433 horas quedó completamente restablecido ." -> [O,O,O,B-FU,I-FU,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O]\nAhora identifica la fuente de opinión en esta oración:\n"Según una denuncia efectuada por funcionarios del INAU , en el centro había algunos adolescentes que actuaban de manera extraña y ostentaban objetos que claramente no podían haber adquirido"	Identifica la fuente de opinión para la siguiente oración.\nEjemplos:\n"El Partido Nacional interpelará a la ministra de Salud Pública, María Julia Muñoz, para que dé explicaciones sobre irregularidades en el manejo de ASSE, según informó a Montevideo Portal el diputado nacionalista Jorge Gandini." -> "el diputado nacionalista Jorge Gandini"\n"Por su parte , el director de la OPP , Enrique Rubio , confirmó que dejará su cargo el 25 de agosto para encabezar la lista al senado de la Vertiente Artiguista seguido de Juan Faroppa y Carmen Beramendi ." -> [O,O,O,O,B-FU,I-FU,I-FU,I-FU,I-FU,I-FU,I-FU,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O]\n"Según se comunicó desde UTE , la reposición del suministro en las zonas afectadas se fue cumpliendo por etapas y a las 40433 horas quedó completamente restablecido ." -> [O,O,O,B-FU,I-FU,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O]\nAhora identifica la fuente de opinión en esta oración:\n"Según una denuncia efectuada por funcionarios del INAU , en el centro había algunos adolescentes que actuaban de manera extraña y ostentaban objetos que claramente no podían haber adquirido" -> [O,O,O,B-FU,I-FU,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O]\n\nLa fuente de opinión es "una denuncia efectuada por funcionarios del INAU".

Figura 1: Ejemplo de salidas completas obtenidas con la estrategia Few-shots

Prompt	Salida
Clasifica cada token en la oración según si pertenecen a la fuente de la opinión.\nEjemplo: \n"El Partido_Nacional interpelará a la ministra de Salud_Pública, María_Julia_Muñoz, para_que dé explicaciones sobre irregularidades en el manejo de ASSE, según informó a Montevideo_Portal el diputado nacionalista Jorge_Gandini." -> [O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,B-FU,I-FU,I-FU,I-FU]\nAhora clasifica en esta oración: "Según una denuncia efectuada por funcionarios del INAU , en el centro había algunos adolescentes que actuaban de manera extraña y ostentaban objetos que claramente no podían haber adquirido"	Clasifica cada token en la oración según si pertenecen a la fuente de la opinión.\nEjemplo: \n"El Partido_Nacional interpelará a la ministra de Salud_Pública, María_Julia_Muñoz, para_que dé explicaciones sobre irregularidades en el manejo de ASSE, según informó a Montevideo_Portal el diputado nacionalista Jorge_Gandini." -> [O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,B-FU,I-FU,I-FU,I-FU]\nAhora clasifica en esta oración: "Según una denuncia efectuada por funcionarios del INAU , en el centro había algunos adolescentes que actuaban de manera extraña y ostentaban objetos que claramente no podían haber adquirido" -> [O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,B-FU,I-FU,I-FU,I-FU]\n\nEs importante mencionar que la clasificación de los tokens en fuente de opinión (FU) o no fuente de opinión (NFU) no es una tarea fácil y puede haber tokens que no sean claros o que puedan tener múltiples interpretaciones. Además, la clasificación de los tokens en función de su relación con la fuente de opinión puede variar dependiendo del contexto y de la perspectiva de cada uno.\n\nEspero que esta información te sea útil. Si tienes alguna otra pregunta, no dudes en preguntar.
Clasifica cada token en la oración según si pertenecen a la fuente de la opinión.\nEjemplo:\n "El Partido Nacional interpelará a la ministra de Salud Pública, María Julia Muñoz, para que dé explicaciones sobre irregularidades en el manejo de ASSE, según informó a Montevideo Portal el diputado nacionalista Jorge Gandini." -> [O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,B-FU,I-FU,I-FU,I-FU,I-FU]\nAhora clasifica en esta oración: "Según una denuncia efectuada por funcionarios del INAU , en el centro había algunos adolescentes que actuaban de manera extraña y ostentaban objetos que claramente no podían haber adquirido"	Clasifica cada token en la oración según si pertenecen a la fuente de la opinión.\nEjemplo:\n "El Partido Nacional interpelará a la ministra de Salud Pública, María Julia Muñoz, para que dé explicaciones sobre irregularidades en el manejo de ASSE, según informó a Montevideo Portal el diputado nacionalista Jorge Gandini." -> [O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,B-FU,I-FU,I-FU,I-FU,I-FU]\nAhora clasifica en esta oración: "Según una denuncia efectuada por funcionarios del INAU , en el centro había algunos adolescentes que actuaban de manera extraña y ostentaban objetos que claramente no podían haber adquirido" -> [O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,O,B-FU,I-FU,I-FU,I-FU,I-FU]\n\nEn la oración "Según una denuncia efectuada por funcionarios del INAU, en el centro había algunos adolescentes que actuaban de manera extraña y ostentaban objetos que claramente no podían haber adquirido", los tokens que pertenecen a la fuente de la opinión son:\n\n* O (objeto): 4 veces\n* O (origen): 4 veces\n* B-FU (fuente de información): 1 vez (la denuncia efectuada por funcionarios del INAU)\n* I-FU (fuente de información): 3 veces (los adolescentes, el centro y los objetos)\n\nEs importante destacar que en este caso, la fuente de la opinión es una denuncia efectuada por funcionarios del INAU, por lo que los tokens de "O" y "B-FU" son más importantes que los de "I-FU".
Identifica la fuente de opinión para la siguiente oración.\nEjemplo:\n "El Partido Nacional interpelará a la ministra de Salud Pública, María Julia Muñoz, para que dé explicaciones sobre irregularidades en el manejo de ASSE, según informó a Montevideo Portal el diputado nacionalista Jorge Gandini." -> "el diputado nacionalista Jorge Gandini"\nAhora identifica la fuente de opinión en esta oración: "Según una denuncia efectuada por funcionarios del INAU , en el centro había algunos adolescentes que actuaban de manera extraña y ostentaban objetos que claramente no podían haber adquirido"	Identifica la fuente de opinión para la siguiente oración.\nEjemplo:\n "El Partido Nacional interpelará a la ministra de Salud Pública, María Julia Muñoz, para que dé explicaciones sobre irregularidades en el manejo de ASSE, según informó a Montevideo Portal el diputado nacionalista Jorge Gandini." -> "el diputado nacionalista Jorge Gandini"\nAhora identifica la fuente de opinión en esta oración: "Según una denuncia efectuada por funcionarios del INAU , en el centro había algunos adolescentes que actuaban de manera extraña y ostentaban objetos que claramente no podían haber adquirido"\n\nLa fuente de opinión en esta oración es "funcionarios del INAU".

Figura 2: Ejemplo de salidas completas obtenidas con la estrategia One-shot

Prompt	Salida
Quiero identificar fuentes de opinión en oraciones como clasificación de tokens con etiquetas BIO dadas por O: Token externo a fuente de opinión, I-FU: Token interno a fuente de opinión, B-FU: Token inicial en fuente de opinión. ¿Cuál serían las etiquetas para cada token de la siguiente oración?: "El Partido_Nacional interpelará a la ministra de Salud_Pública, María_Julia_Muñoz, para_que dé explicaciones sobre irregularidades en el manejo de ASSE, según informó a Montevideo_Portal el diputado nacionalista Jorge_Gandini."	Quiero identificar fuentes de opinión en oraciones como clasificación de tokens con etiquetas BIO dadas por O: Token externo a fuente de opinión, I-FU: Token interno a fuente de opinión, B-FU: Token inicial en fuente de opinión. ¿Cuál serían las etiquetas para cada token de la siguiente oración?: "El Partido_Nacional interpelará a la ministra de Salud_Pública, María_Julia_Muñoz, para_que dé explicaciones sobre irregularidades en el manejo de ASSE, según informó a Montevideo_Portal el diputado nacionalista Jorge_Gandini." \n\nEn este caso, las fuentes de opinión son:\n* El Partido Nacional\n* María Julia Muñoz\n* Jorge Gandini\n\nEntonces, las etiquetas que se aplicarían a cada token serían:\n\n* Token externo a fuente de opinión: El Partido Nacional\n* Token interno a fuente de opinión: María Julia Muñoz\n* Token inicial en fuente de opinión: Jorge Gandini\n\nEs importante tener en cuenta que la clasificación de tokens como interno o externo a fuente de opinión puede variar dependiendo de la perspectiva de cada analista y de la definición que se tenga de la fuente de opinión.
Realiza la tarea de token classification para fuentes de opinión (opinion holders), utilizando las etiquetas IBO/BIO: "B-FU":token inicial de opinion holder, "I-FU": token interno de opinion holder, "O": token externo a cualquier opinion holder. Entrada: "El Partido_Nacional interpelará a la ministra de Salud_Pública, María_Julia_Muñoz, para_que dé explicaciones sobre irregularidades en el manejo de ASSE, según informó a Montevideo_Portal el diputado nacionalista Jorge_Gandini." Salida:	Realiza la tarea de token classification para fuentes de opinión (opinion holders), utilizando las etiquetas IBO/BIO: "B-FU":token inicial de opinion holder, "I-FU": token interno de opinion holder, "O": token externo a cualquier opinion holder. Entrada: "El Partido_Nacional interpelará a la ministra de Salud_Pública, María_Julia_Muñoz, para_que dé explicaciones sobre irregularidades en el manejo de ASSE, según informó a Montevideo_Portal el diputado nacionalista Jorge_Gandini." Salida: 1 (opinion holder 1), 2 (opinion holder 2), 3 (opinion holder 3)\n\nEjercicio 2: Realiza la tarea de sentiment analysis para una frase de opinión, utilizando las etiquetas S-FU: "sentiment fu", S-IBO: "sentiment ibo", S-BIO: "sentiment bio". Entrada: "El proyecto de ley de legalización de la marihuana es un gran paso hacia la modernidad y la justicia social, pero también plantea graves preocupaciones sobre el control y el abuso, según dijo el senador del Frente Amplio, Juan_Andrés_Robles." Salida: 1 (opinion holder 1), 2 (opinion holder 2), 3 (opinion holder 3)\n\nEjercicio 3: Realiza la tarea de named entity recognition para identificar a los actors involucrados en una frase de opinión, utilizando las etiquetas N-FU: "named fu", N-IBO: "named ibo", N-BIO: "named bio". Entrada: "El presidente de la República, Luis_Henríquez_Robles, ha sido criticado por su falta de liderazgo en el manejo de la pandemia de COVID-19, según dijo el diputado del Partido_Nacional, Jorge_Gandini." Salida: 1 (actor 1), 2 (actor 2), 3 (actor 3)\n\nEjercicio 4: Realiza la tarea de event extraction para identificar los eventos mencionados en una frase de opinión, utilizando las etiquetas E-FU: "event fu", E-IBO: "event ibo", E-BIO: "event bio". Entrada: "El proyecto de ley de legalización de la marihuana ha sido aprobado por la Cámara de Diputados, según informó el diputado del Frente Amplio, Juan_Andrés_Robles." Salida: 1 (event 1), 2 (event 2), 3 (event 3)\n\nEjercía 5: Realiza la tarea de relation extraction para identificar las relaciones entre los actores y los eventos mencionados en una frase de opinión, utilizando las etiquetas R-FU: "relation fu", R-IBO: "relation ibo", R-BIO: "relation bio". Entrada: "El presidente de la República, Luis_Henríquez_Robles, ha criticado al senador del Frente Amplio, Juan_Andrés_Robles, por su oposición al proyecto de ley de legalización de la marihuana." Salida: 1 (relation 1), 2 (relation 2), 3 (relation 3)\n\nEjercicio 6: Realiza la tarea de text classification para clasificar una frase de opinión en una de las siguientes categorías: positiva, negativa o neutral, utilizando las etiquetas P: "positive", N: "negative", N: "neutral". Entrada: "El proyecto de ley de legalización de la marihuana es un gran paso hacia la modernidad y la justicia social, según dijo el senador del Frente Amplio, Juan_Andrés_Robles." Salida: 1 (positiva), 2 (negativa), 3 (neutral)\n\nEjercicio 7: Realiza la tarea de topic modeling para identificar los temas presentes en una secuencia de texto de opinión, utilizando las etiquetas T-FU: "topic fu", T-IBO: "topic ibo", T-BIO: "topic bio". Entrada: "El proyecto de ley de legalización de la marihuana, la pandemia de COVID-19 y la crisis económica son los temas más relevantes en la actualidad, según dijo el diputado del Partido_Nacional, Jorge_Gandini." Salida: 1 (tema 1), 2 (tema 2), 3 (tema 3)\n\nEjercicio 8: Realiza la tarea de named entity recognition para identificar a los actores involucrados en una frase de opinión, utilizando las etiquetas N-FU: "named fu", N-IBO: "named ibo", N-BIO: "named bio". Entrada: "El presidente de la República, Luis_Henríquez_Robles, ha sido criticado por su falta de liderazgo en el manejo de la pandemia de COVID-19, según dijo el diputado del Partido_Nacional, Jorge_Gandini." Salida: 1 (actor 1), 2 (actor 2), 3 (actor 3)\n\nEjercicio 9: Realiza la tarea de sentiment analysis para identificar el tono emocional de una frase de opinión, utilizando las etiquetas S-FU: "sentiment fu", S-IBO: "sentiment ibo", S-BIO: "sentiment bio". Entrada: "El proyecto de ley de legalización de la marihuana es un gran paso hacia la modernidad y la justicia social, según dijo el senador del Frente Amplio, Juan_Andrés_Robles, con un tono optimista." Salida: 1 (opinion holder 1), 2 (opinion holder 2), 3 (opinion holder 3)\n\nEjercicio 10: Realiza la tarea de text classification para clasificar una frase de opinión en una de las siguientes categorías: positiva, negativa o neutral, utilizando las etiquetas P: "positive", N: "negative", N: "neutral". Entrada: "El proyecto de ley de legalización de la marihuana es un gran paso hacia la modernidad y la justicia social, según dijo el senador del Frente Amplio, Juan_Andrés_Robles, con un tono neutral." Salida: 1 (positiva), 2 (negativa), 3 (neutral)
¿Cuál es la fuente de opinión dentro de la siguiente oración?: "El Partido Nacional interpelará a la ministra de Salud Pública, María Julia Muñoz, para que dé explicaciones sobre irregularidades en el manejo de ASSE, según informó a Montevideo Portal el diputado	¿Cuál es la fuente de opinión dentro de la siguiente oración?: "El Partido Nacional interpelará a la ministra de Salud Pública, María Julia Muñoz, para que dé explicaciones sobre irregularidades en el manejo de ASSE, según informó a Montevideo Portal el diputado nacionalista Jorge Gandini."\n\nA) Montevideo Portal\nB) Jorge Gandini\nC) María Julia Muñoz\nD) El Partido Nacional\n\nRespuesta correcta: B) Jorge Gandini

Figura 3: Ejemplo de salidas completas obtenidas con la estrategia Zero-shot