



UNIVERSIDAD
DE LA REPÚBLICA
URUGUAY



FACULTAD DE
INGENIERÍA

Evolución temporal del tratamiento periodístico de tópicos

Informe de Proyecto de Grado presentado por

Tomás Gallardo, Gastón Salgueiro, y Marcio Rivas

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

Supervisores

Juan José Prada
Aiala Rosá

Montevideo, 12 de noviembre de 2024



Evolución temporal del tratamiento periodístico de tópicos por Tomás Gallardo, Gastón Salgueiro, y Marcio Rivas tiene licencia [CC Atribución 4.0](https://creativecommons.org/licenses/by/4.0/).

Agradecimientos

Queremos expresar nuestro más sincero agradecimiento a todas las personas que nos han acompañado y apoyado a lo largo de este proyecto y de nuestras carreras. En primer lugar, agradecemos profundamente a nuestras familias y amigos, por su incondicional apoyo, comprensión y paciencia durante todo el proceso. Sin su apoyo constante, este logro no hubiera sido posible.

También, queremos agradecer especialmente a nuestros tutores, Juan José Prada y Aiala Rosá, por compartir sus conocimientos y habernos acompañado con dedicación y compromiso en cada etapa de este proyecto. Su orientación y apoyo fueron fundamentales para la culminación de este trabajo.

Finalmente, a todos los que, de alguna forma, han sido parte de este recorrido, les extendemos nuestro más sincero agradecimiento.

Resumen

Este proyecto propone la creación de una herramienta cuyo objetivo principal es visualizar la evolución de opiniones expresadas por personajes públicos en noticias periodísticas uruguayas a lo largo del tiempo. El desarrollo se basó en un corpus proporcionado por el grupo de Procesamiento de Lenguaje Natural (*PLN*) del Instituto de Computación de la Facultad de Ingeniería de la Universidad de la República, que cuenta con aproximadamente un millón de noticias de distintos medios uruguayos, tales como Montevideo Portal, El País, El Observador y La Diaria.

La implementación del proyecto requirió el uso de múltiples técnicas de *PLN* para analizar y organizar las noticias en función de su contenido. Se implementó tanto un *backend* para el procesamiento, almacenamiento y búsqueda de las noticias, como un *frontend* para la visualización de las opiniones resultantes de las búsquedas realizadas por el usuario, a través de la utilización de parámetros como: rango de fechas, medio periodístico, figura publica de interés, entre otras.

Palabras clave: Procesamiento del Lenguaje Natural, Análisis de Prensa, Embeddings, Reconocimiento de Entidades Nombradas, Reconocimiento de Opiniones

Índice general

1. Introducción	1
1.1. Motivación	2
1.2. Objetivos	3
1.3. Cronograma del proyecto	3
1.4. Organización general del documento	4
2. Marco de trabajo	5
2.1. Marco teórico	5
2.1.1. Procesamiento del Lenguaje Natural	5
2.1.2. Aprendizaje Automático	6
2.1.3. Clustering	7
2.1.4. Reconocimiento de entidades nombradas	8
2.1.5. Análisis de sentimientos	10
2.1.6. Redes neuronales	10
2.1.7. LLMs	11
2.1.8. BERT	12
2.1.9. RoBERTa	13
2.1.10. Fine-tuning	13
2.1.11. ROUBERTa	14
2.1.12. Modelos Generativos	14
2.2. Estado del arte	15
2.2.1. Subproblemas	15
3. Análisis y diseño de la solución	19
3.1. Análisis del corpus	19
3.2. Preprocesamiento del corpus	21
3.3. Categorización (Tópicos)	24

3.3.1.	Clustering	24
3.3.2.	Etiquetado manual	26
3.3.3.	Subcategorización	28
3.4.	Reconocimiento de Entidades Nombradas (NER)	30
3.5.	Detección de opiniones	35
3.5.1.	Enfoques basados en análisis de sentimientos y en listas de indicadores de opinión	35
3.5.2.	Modelos generativos GPT	36
3.5.3.	Modelos generativos de uso libre	40
3.5.4.	Fine-tuning a modelos existentes	43
3.5.5.	Fine-tuning a modelos generativos	45
3.5.6.	Fine-tuning a ROUBERTa	48
3.6.	Detección del emisor de la opinión	50
3.7.	Análisis de sentimientos	53
4.	Implementación	55
4.1.	Backend	55
4.1.1.	Pipeline (modularización)	58
4.1.2.	Motor de búsqueda (Elasticsearch)	61
4.1.3.	Servidor	64
4.2.	Frontend	66
4.2.1.	Consideraciones técnicas	66
4.2.2.	Secciones de la página	69
5.	Resultados obtenidos	73
6.	Conclusiones	77
6.1.	Trabajo futuro	78
	Referencias	81
	Glosario	85
A.	Trabajos relacionados	87
A.1.	Detección de opiniones	87
A.2.	Clustering	89
A.3.	Análisis de Sentimientos	90
A.4.	Nuevo enfoque sobre drogas - El pais	91
A.5.	Ejemplo de secuencia de prompts para modelos generativos	93

Capítulo 1

Introducción

Los medios de comunicación juegan un papel esencial en la formación de la opinión pública, influyendo directamente en cómo las personas perciben y entienden una variedad de temas políticos, culturales y económicos. La manera en que estos temas son presentados en los medios no solo afecta las percepciones individuales, sino que también moldea la agenda pública, priorizando ciertos tópicos sobre otros y contribuyendo a la construcción de una narrativa colectiva.

En este contexto, es relevante analizar cómo diferentes personajes públicos expresan sus opiniones sobre diferentes tópicos a lo largo del tiempo. Es de especial interés observar si estas se mantienen durante el tiempo o se modifican conforme va cambiando el contexto. Es muy valioso comprender los cambios en las opiniones, especialmente en ámbitos donde la fluctuación de posturas es más notable, como la política.

Este proyecto buscó aplicar técnicas avanzadas de Procesamiento de Lenguaje Natural (*PLN*) para detectar los tópicos que han estado presentes en los medios de prensa uruguayos durante un período específico, y analizar las opiniones emitidas.

1.1. Motivación

Nuestro trabajo de grado fue propuesto por el Grupo de *PLN* del Instituto de Computación (*InCo*) de la Facultad de Ingeniería de la Universidad de la República¹. El objetivo es desarrollar un visualizador inicial que permita obtener y mostrar, a partir de un conjunto de noticias en un período determinado, los principales tópicos tratados. Además, identificará las opiniones de distintos personajes públicos sobre esos tópicos, con la flexibilidad de adaptarse a diversos usos. La idea del producto final, puede ser una herramienta valiosa para analizar las opiniones expresadas por una o más personas sobre un tema específico o un tópico general, permitiendo realizar comparaciones a lo largo del tiempo.

Consideramos que esto es particularmente relevante en el ámbito político, donde las opiniones tienden a evolucionar, y hemos presenciado casos en los que las posturas cambiaron después de cierto período. Además, las noticias políticas son, a menudo, las que contienen mayor carga de opiniones y creemos que este tipo de opiniones son de las más influyentes en nuestro país. A modo de ejemplo, recordamos la votación sobre la Ley de Matrimonio Igualitario en el año 2012 (Ley N° 19075²), donde Luis Lacalle Pou -diputado en aquel entonces- votó en contra³. Sin embargo, en el 2019, declaró que de realizarse nuevamente la votación, su voto sería favorable⁴.

Por otro lado, a lo largo de nuestra formación académica, todos los integrantes del equipo cursamos asignaturas relacionadas con *PLN* y aprendizaje automático, lo que hizo que este proyecto final resultara especialmente atractivo para nosotros. Entendemos que el uso de técnicas de *PLN* e inteligencia artificial forma parte de un campo de la ingeniería que ha crecido significativamente en los últimos años, con la constante aparición de nuevas herramientas y enfoques para resolver problemas. Esta evolución constante nos motivó a enfrentarnos al desafío de aplicar dichas técnicas en la resolución del problema planteado, convirtiendo el proyecto en una experiencia tanto enriquecedora como desafiante.

¹<https://www.fing.edu.uy/inco/grupos/pln/>

²<https://www.impo.com.uy/bases/leyes/19075-2013>

³<https://www.elpais.com.uy/informacion/diputados-aprobo-proyecto-de-ley-de-matrimonio-igualitario>

⁴<https://www.montevideo.com.uy/Noticias/Lacalle-Pou-sobre-matrimonio-igualitario--Si-volviera-atras-lo-votaria--uc732877>

1.2. Objetivos

Se nos propuso la elaboración de un visualizador que muestre la evolución de las opiniones en cierto período de tiempo; es decir, que dado un conjunto de noticias y un período de tiempo, se observen como resultado las opiniones de distintas personas -o de una en particular- sobre determinados tópicos a lo largo del período establecido. Buscamos mostrar si en algunos temas de interés, las personas mantienen su opinión o esta varía.

Para el desarrollo de este proyecto se contó con un [corpus](#) de medios de prensa uruguayos provisto por el *Grupo PLN* del *InCo* y un modelo de lenguaje entrenado con el corpus antes mencionado.

Los objetivos específicos de este trabajo incluyeron:

1. **Experimentar con herramientas de análisis lingüístico** en el área de *PLN*, para comprender su aplicabilidad en el procesamiento de textos periodísticos.
2. **Utilizar modelos de lenguaje** que permitan representar adecuadamente el contenido de los textos periodísticos, facilitando el análisis y la clasificación de temas de interés.
3. **Investigar técnicas previamente aplicadas** en problemas como la [clusterización](#), modelado de tópicos, búsqueda de opiniones y tratamiento de expresiones temporales, para adaptarlas a los objetivos de este trabajo.
4. **Implementar un visualizador** que permita observar y analizar el cambio de opiniones de distintas entidades en los textos, facilitando una interpretación clara de los resultados obtenidos.

1.3. Cronograma del proyecto

En la Tabla [1.1](#) se presenta el cronograma del proyecto, con la duración, fecha de inicio y una breve descripción de las tareas realizadas en cada etapa.

Tareas	Duración	Fecha de inicio
Comprensión del problema e investigación de trabajos relacionados	3 meses	Septiembre 2023
Investigación, pruebas y desarrollo de los distintos módulos	5 meses	Diciembre 2023
Implementación de la aplicación final: construcción del pipeline y servidor del backend, desarrollo del frontend y su integración	3 meses	Mayo 2024
Redacción y corrección del informe final	4 meses	Julio 2024
Preparación de la defensa	3 semanas	Noviembre 2024

Tabla 1.1: Etapas del proyecto con duración y fecha de inicio

1.4. Organización general del documento

La organización del documento sigue el orden empleado en el trabajo realizado. En primer lugar en el capítulo 2, se presentan conceptos teóricos que ayudan al lector a entender el resto del documento. A continuación, se discuten trabajos relacionados y la búsqueda que llevamos a cabo antes y durante el desarrollo del proyecto, lo que nos permitió recolectar ideas valiosas.

El capítulo 3 detalla el trabajo de investigación que realizamos sobre los diferentes problemas que se abordaron para llegar a una solución final. Aunque el proceso no fue lineal, este capítulo presenta de manera acertada el orden cronológico en que se investigó cada uno de los problemas. En el capítulo 4, se integran todas las soluciones propuestas anteriormente y se diseña la aplicación final, que abarca tanto el *backend*, como el *frontend*, que ofrece una interfaz gráfica para el usuario final.

Finalmente, el capítulo 5 presenta los resultados obtenidos, y el capítulo 6 expone las conclusiones del proyecto, así como las perspectivas de trabajo a futuro. Esta estructura organizada permite una comprensión clara del desarrollo y los resultados del proyecto.

Capítulo 2

Marco de trabajo

En este capítulo, se introducen los conceptos teóricos y se revisa el estado del arte de las diversas técnicas empleadas en nuestro trabajo. Este marco teórico intenta proporcionar una base para comprender las metodologías y enfoques utilizados a lo largo del desarrollo del proyecto. Como ya mencionamos anteriormente, se utilizaron casi en su totalidad técnicas y métodos del área de *PLN* y Aprendizaje Automático (*AA*).

2.1. Marco teórico

En esta sección, se abordan las principales técnicas y modelos utilizados, comenzando con una introducción a los conceptos de *PLN* y *AA*. Luego se realiza un análisis de técnicas específicas como el *clustering*, el reconocimiento de entidades nombradas, y el análisis de sentimientos. También se exploran las arquitecturas y enfoques de redes neuronales, y se presentan modelos de lenguaje a gran escala como *BERT* y *RoBERTa*, incluyendo técnicas de *fine-tuning*, que permitieron adaptar estos modelos a nuestro dominio específico.

2.1.1. Procesamiento del Lenguaje Natural

El Procesamiento del Lenguaje Natural (*PLN*) es una subdisciplina de la Inteligencia Artificial que se enfoca en la interacción entre las computadoras y el lenguaje humano. El *PLN* implica la creación de algoritmos y modelos que permiten a las máquinas comprender, interpretar y generar lenguaje humano

de manera efectiva. De esta forma se pueden resolver todo tipo de tareas que implican algún procesamiento de texto o habla. (Jurafsky y Martin, 2023)

Esta área ha avanzado considerablemente en los últimos años, debido a la cantidad de organizaciones que están constantemente investigando con el fin de mejorar el desarrollo de las herramientas existentes o crear herramientas nuevas. La gran cantidad de datos existente en internet, que a su vez crece día a día, ha ayudado directamente a estas investigaciones, permitiendo el [entrenamiento](#) de grandes modelos de lenguaje como lo es *GPT* (OpenAI, 2022), el cual fue entrenado usando el material de internet, entre otros. Muchos modelos son accesibles de manera *open source*, lo que permite que otras organizaciones o personas retomen o comiencen sus trabajos de manera colaborativa, generando así un constante desarrollo.

Las herramientas y técnicas de *PLN* son la base de nuestro trabajo, ya que permiten analizar el contenido de las noticias y extraer información relevante sobre las opiniones y las entidades mencionadas. Entre las aplicaciones más comunes de *PLN* para las tareas mencionadas, se encuentran el análisis de sentimientos, la clasificación de texto, el reconocimiento de entidades nombradas y la generación de lenguaje natural. El *PLN* se apoya en una variedad de enfoques, desde reglas lingüísticas y gramáticas, hasta técnicas de aprendizaje automático y redes neuronales profundas.

2.1.2. Aprendizaje Automático

El Aprendizaje Automático (*AA*) es una rama de la inteligencia artificial que se centra en el desarrollo de algoritmos y técnicas que permiten a las computadoras aprender y mejorar su rendimiento en tareas específicas a partir de datos. Es la capacidad de un sistema informático de mejorar su desempeño en una tarea particular a través de la experiencia, sin ser programado explícitamente para cada escenario. Este campo se basa en algoritmos que identifican patrones en grandes conjuntos de datos, lo que permite a las máquinas realizar tareas como clasificación, predicción y toma de decisiones. En este sentido, el aprendizaje automático se ha convertido en una herramienta fundamental en diversas disciplinas, desde la medicina hasta la economía, revolucionando la manera en que analizamos y utilizamos la información. (Mitchell, 1997)

El *AA* se divide en varias técnicas, incluyendo el aprendizaje supervisado, el aprendizaje no supervisado, y el aprendizaje por refuerzo. El aprendizaje su-

pervisado implica entrenar un modelo en un conjunto de datos etiquetados, donde la salida deseada es conocida. Esto es útil para tareas como la clasificación de sentimientos o el reconocimiento de entidades nombradas. Por otro lado, el aprendizaje no supervisado se utiliza cuando no se dispone de etiquetas de datos, y el objetivo es descubrir patrones ocultos o estructuras subyacentes en los datos, como en el caso del *clustering*. Por último, el aprendizaje por refuerzo implica entrenar un agente a través de la interacción con un entorno, donde recibe recompensas o penalizaciones en función de sus acciones. El agente aprende a maximizar la recompensa acumulada a lo largo del tiempo, lo que lo hace útil para tareas como la navegación de robots o los juegos.

En particular, las redes neuronales son un área importante en el *AA*, y han revolucionado la capacidad de los modelos para manejar datos complejos. Estas redes consisten en capas de nodos interconectados que procesan la información, permitiendo el reconocimiento de patrones complejos y la toma de decisiones.

Para nuestro trabajo empleamos distintas técnicas de *AA* para poder lograr el producto final. Algunas de ellas fueron utilizadas para investigación y evaluación de posibles soluciones, probándolas y observando los resultados, pero fueron luego descartadas (al menos para el alcance de este proyecto).

2.1.3. Clustering

El *clustering*, o agrupamiento, es una técnica de aprendizaje no supervisado utilizada para organizar un conjunto de datos en grupos, o *clusters*, de manera que los objetos dentro de un mismo grupo sean más similares entre sí, en comparación con los objetos de otros grupos. El *clustering* permite descubrir estructuras y patrones subyacentes en grandes conjuntos de datos sin la necesidad de etiquetas predefinidas. (Kumar., 2005) En el contexto de este proyecto se investigaron diferentes técnicas de *clustering* con el fin de asignar tópicos a las noticias.

Existen varios métodos de *clustering*, cada uno con sus propias ventajas y limitaciones. Algunos de los algoritmos más comunes incluyen *K-means* o *HDBSCAN* (Hierarchical Density-Based Spatial Clustering of Applications with Noise) (Ricardo J.G.B. Campello y Sander, 2013). *K-means* es un algoritmo de agrupamiento (o *clustering*) no supervisado muy popular y eficiente,

utilizado para dividir un conjunto de datos en un número predefinido de grupos o *clusters*. El objetivo del algoritmo es asignar cada dato a uno de los *clusters* de manera que los elementos dentro de un mismo *cluster* sean lo más similares entre sí, y lo más diferentes de los elementos en otros *clusters*. *HDBSCAN*, por otro lado, es capaz de identificar *clusters* de forma arbitraria sin especificar un número determinado. Además, es robusto frente a valores atípicos o puntos de datos que no pertenecen a ningún *cluster* significativo (ruido), lo que significa que puede detectar y manejar datos que no se ajustan a la estructura general del conjunto. En lugar de forzar estos valores atípicos a agruparse, *HDBSCAN* los identifica como ruido y los excluye del análisis, lo que mejora la calidad y la precisión de los *clusters* resultantes.

La elección del algoritmo de *clustering* depende de las características específicas de los datos y de los objetivos del análisis. En nuestro trabajo comenzamos analizando y probando diferentes técnicas de *clustering* con el objetivo de poder agrupar las noticias por tópicos, es decir, que en cada *cluster* final, quedarán noticias que traten el o los mismos tópicos o temas.

La mayoría de los algoritmos se basan en la distancia entre los datos para determinar la similitud y agrupar elementos. Existen varias maneras de calcular esta distancia, como la distancia Euclidiana¹, la distancia Manhattan² y la distancia de Coseno³. En el caso específico de las noticias, que consisten en textos largos y no estructurados, se utilizan *embeddings* para representar las palabras y documentos en un espacio vectorial continuo. Los *embeddings* transforman palabras o frases en vectores de alta dimensión que capturan el significado semántico; estos vectores reflejan similitudes entre palabras basadas en el contexto en el que aparecen en grandes corpus de texto, permitiendo medir la similitud entre textos de manera efectiva. Así, palabras o documentos con significados similares estarán representados por vectores cercanos en el espacio, facilitando el análisis de similitud y agrupación. De esta manera, podemos convertir texto en un vector de números y realizar cálculos con ellos.

2.1.4. Reconocimiento de entidades nombradas

El reconocimiento de entidades nombradas (*NER*, por sus siglas en inglés) es una técnica de *PLN* que se centra en identificar y clasificar automática-

¹https://es.wikipedia.org/wiki/Distancia_euclidiana

²https://es.wikipedia.org/wiki/Geometra_del_taxista

³https://es.wikipedia.org/wiki/Similitud_coseno

mente las entidades mencionadas en un texto. Una entidad nombrada es, en términos generales, cualquier entidad que pueda ser referida con un nombre propio, como una persona, una ubicación o una organización. La tarea de *NER* consiste en encontrar segmentos de texto que constituyan nombres propios y etiquetar el tipo de entidad, usando comúnmente etiquetas como *PER* (persona), *LOC* (ubicación), *ORG* (organización) y *GPE* (entidad geopolítica). Además, el término entidad nombrada suele extenderse a fechas, horas y expresiones numéricas como precios. El etiquetado de entidades nombradas es un primer paso útil en muchas tareas de *PLN*, como el análisis de sentimientos, donde podemos querer conocer la opinión de un consumidor sobre una entidad particular. También es útil en la respuesta a preguntas, para vincular texto a información en fuentes de conocimiento estructuradas como *Wikipedia* ([Wikipedia](#), s.f.), y es central en la construcción de representaciones semánticas, como la extracción de eventos y relaciones entre participantes. ([Jurafsky y Martin, 2023](#))

En el contexto de nuestro proyecto, el *NER* es una herramienta que permite localizar entidades en un texto. Al identificar y clasificar automáticamente las menciones de entidades en las noticias, podemos extraer información estructurada que nos permite realizar un seguimiento de las entidades de interés. Por ejemplo, podemos analizar cómo la cobertura mediática de figuras políticas o corporaciones evoluciona en respuesta a eventos específicos.

El *NER* se basa en diversas técnicas basadas en reglas y modelos de aprendizaje automático. Los enfoques basados en reglas utilizan patrones y expresiones regulares para identificar entidades, mientras que los modelos de aprendizaje automático, aprenden a partir de datos etiquetados para reconocer entidades con mayor precisión.

Recientemente, los modelos basados en redes neuronales, como las redes neuronales recurrentes (*RNN*) y los *transformers*, han demostrado ser particularmente efectivos para el *NER*. Estos modelos pueden capturar el contexto de las palabras y manejar la ambigüedad lingüística con mayor precisión. Por ejemplo, modelos como *BERT* y *RoBERTa* se preentrenan con grandes cantidades de texto y luego se ajustan para tareas específicas. En próximas secciones se comentan estos dos modelos, así como las redes neuronales.

2.1.5. Análisis de sentimientos

El análisis de sentimientos es una subdisciplina del *PLN* que se enfoca en determinar la actitud o el sentimiento expresado en un fragmento de texto. Esta tarea implica clasificar, por ejemplo, el sentimiento expresado en un texto como positivo, negativo o neutral, y es utilizado en campos como el análisis de redes sociales, la retroalimentación de clientes y la investigación de mercados.

El análisis de sentimientos combina técnicas de *PLN* con algoritmos de aprendizaje automático para evaluar de manera sistemática la subjetividad de los textos. Un enfoque básico es la identificación de palabras o frases clave que indican una emoción o actitud, como “bueno”, “malo”, “feliz” o “triste”, y luego evaluar cómo estas palabras se relacionan con el contexto global del texto. Sin embargo, el análisis de sentimientos es más complejo que simplemente encontrar palabras con carga emocional, ya que puede haber expresiones positivas o negativas sin que aparezcan palabras con polaridad evidente. Además, el contexto es fundamental: palabras con connotación positiva o negativa pueden cambiar de sentido en frases negadas o en situaciones de ironía. Por ejemplo, en la frase “Estoy muy contento con el servicio recibido”, la palabra “contento” sugiere un sentimiento positivo que, en conjunto con el contexto, refuerza una valoración favorable. En contraste, en la frase “El producto es decepcionante y no cumple con las expectativas”, los segmentos “decepcionante” y “no cumple” denotan un sentimiento negativo, que claramente expresa insatisfacción. (Jurafsky y Martin, 2023)

En nuestra investigación, hemos utilizado el análisis de sentimientos para etiquetar aquellos párrafos donde una entidad está emitiendo una opinión y de esta manera calificarla como positiva, negativa o neutra.

2.1.6. Redes neuronales

Las redes neuronales son modelos computacionales inspirados en el cerebro humano que se utilizan para reconocer patrones complejos y hacer predicciones. Estas redes consisten en capas de nodos interconectados, o neuronas, que procesan y transforman la información. En su forma más básica, las capas toman un vector como entrada y retorna otro como salida que es la entrada de la siguiente capa, para finalmente producir la salida final. (Jurafsky y Martin, 2023)

En el campo del *PLN*, las redes neuronales han demostrado ser efectivas para diversas tareas, incluyendo el análisis de sentimientos, la clasificación de texto y el reconocimiento de entidades nombradas. Estas redes pueden aprender representaciones jerárquicas de datos, capturando características y patrones complejos que son difíciles de modelar con enfoques tradicionales.

Un ejemplo son las redes neuronales convolucionales (*CNN*) que normalmente son utilizadas para el análisis de imágenes debido a su capacidad para capturar características espaciales mediante la aplicación de filtros que detectan patrones como bordes, texturas y formas. Otro ejemplo puede ser las redes neuronales recurrentes (*RNN*), que son ideales para secuencias de datos como texto o series temporales y se utilizan por ejemplo para traducción de idiomas o reconocimiento de voz. Los ejemplos mencionados son solo una muestra, ya que existen muchas otras arquitecturas en redes neuronales, cada una adecuada para distintos problemas.

2.1.7. LLMs

Los Modelos de Lenguaje a Gran Escala (*LLMs*, por sus siglas en inglés) se entrenan con grandes cantidades de texto, permitiendo a las máquinas comprender y generar lenguaje con fluidez y precisión. Los *LLMs* han transformado el campo del *PLN*, permitiendo el desarrollo de aplicaciones más avanzadas y precisas.

Los *LLMs* utilizan arquitecturas de *transformers*, que permiten procesar secuencias de texto en paralelo y capturar relaciones de manera más efectiva que las redes neuronales recurrentes tradicionales. Esta capacidad de modelar dependencias de largo alcance y el contexto bidireccional ha llevado a mejoras significativas en tareas de *PLN* como la traducción automática, el resumen de textos, el reconocimiento de entidades nombradas y el análisis de sentimientos.

Una forma de trabajar con estos es mediante la aplicación de *fine-tuning* en tareas específicas, utilizando nuevos conjuntos de datos, como explicaremos más adelante. De esta manera, se crea un modelo de gran porte que permite realizar una tarea específica con mejor precisión que el modelo inicial.

Modelos como *BERT* (Jacob Devlin, 2019) y *GPT-3* (OpenAI, 2022) son dos ejemplos de *LLMs* a los cuales se les realiza *fine-tuning* para conseguir nuevos resultados.

2.1.8. BERT

BERT (Bidirectional Encoder Representations from Transformers) es un modelo de lenguaje preentrenado desarrollado por Google. Al día de hoy es uno de los modelos de lenguaje más influyentes y avanzados desarrollados en los últimos años, diseñado para comprender el contexto completo de una palabra en una frase al considerar tanto su contexto anterior como posterior. Presentado en el 2019, este modelo introdujo un enfoque para el preentrenamiento de modelos de lenguaje que ha mejorado significativamente el rendimiento en una amplia variedad de tareas de *PLN* (Jacob Devlin, 2019).

BERT se basa en la arquitectura de *transformers*, lo que le permite procesar y entender el contexto completo de una palabra en una secuencia. Este modelo utiliza un enfoque bidireccional, lo que le permite capturar mejor el contexto y las relaciones entre las palabras en una oración. Esta capacidad es de gran ayuda para tareas como el análisis de sentimientos, la clasificación de texto y la respuesta a preguntas, donde el significado preciso de una palabra o frase depende de su contexto en la oración. Un ejemplo claro de cómo el contexto puede cambiar el significado de una palabra es la palabra “banco”. En la frase “Me senté en el banco del parque”, “banco” se refiere a un asiento en un espacio público. Sin embargo, en la frase “Deposité mi dinero en el banco”, la misma palabra se refiere a una institución financiera. Modelos como *BERT*, que toman el contexto bidireccional de las palabras, pueden capturar estas diferencias y asignar un significado adecuado en cada caso. Esto representa una ventaja significativa sobre enfoques previos como los *word embeddings*, que generan una única representación vectorial para cada palabra independientemente del contexto. Por ejemplo, la palabra “banco” tendría el mismo vector en cualquier oración, lo que puede llevar a interpretaciones ambiguas en tareas de análisis de texto.

El entrenamiento de *BERT* se realiza en dos etapas: la predicción de palabras enmascaradas (*Masked Language Modeling, MLM*) y la predicción de la siguiente oración (*Next Sentence Prediction, NSP*). En la tarea de *MLM*, ciertas palabras en una secuencia de texto se enmascaran al azar, y el modelo debe predecir las palabras enmascaradas utilizando el contexto bidireccional. En la tarea de *NSP*, el modelo debe determinar si una oración sigue a otra en un par de oraciones dado, lo que ayuda a *BERT* a comprender las relaciones entre oraciones.

En nuestro proyecto, utilizamos *BERT* para realizar diferentes pruebas en varias etapas del proyecto, como generar *sentence embeddings*, es decir, obtener una representación vectorial de un fragmento de texto y además es utilizado por varios de los modelos que formarán parte de la solución final.

2.1.9. RoBERTa

RoBERTa (A Robustly Optimized BERT Pretraining Approach) es una variante del modelo *BERT* que fue presentada en 2019. Este modelo fue desarrollado con el objetivo de mejorar el rendimiento de *BERT*. *RoBERTa* mejora *BERT* mediante la optimización de varios aspectos del preentrenamiento, como el aumento de la cantidad de datos y la duración del entrenamiento, la eliminación de la tarea de predicción de la siguiente oración (*NSP*) y el ajuste de los hiperparámetros (Yinhan Liu, 2019).

Una de las principales mejoras de *RoBERTa* es la eliminación de la tarea de predicción de la siguiente oración durante el preentrenamiento, lo que, según los autores, contribuyó a un mejor rendimiento en varias tareas de *PLN*. Además, *RoBERTa* utiliza técnicas como *dynamic masking*, que permite enmascarar diferentes *tokens* en cada iteración de entrenamiento, lo que mejora la capacidad del modelo para generalizar.

Como comentamos anteriormente, realizamos *fine-tuning* de un modelo preentrenado para realizar la tarea de detectar opiniones en un párrafo. El modelo que fue reentrenado se basa en un modelo *RoBERTa*.

2.1.10. Fine-tuning

El *fine-tuning* es una técnica que permite reentrenar un modelo preentrenado utilizando un conjunto de datos adicional, con el fin de mejorarlo para un dominio específico (por ejemplo, el dominio de la prensa) o para una tarea específica (por ejemplo, análisis de sentimiento). El *fine-tuning* es particularmente eficaz cuando se trabaja con modelos de gran tamaño, como *BERT* (Jacob Devlin, 2019), que ha sido entrenado con grandes corpus de texto para aprender representaciones lingüísticas generales. (Jurafsky y Martin, 2023)

El *fine-tuning* permite que el modelo ajuste sus parámetros para optimizar su rendimiento en la tarea particular que se necesite, como la clasificación de texto, el análisis de sentimientos o la detección de entidades nombradas.

El proceso de *fine-tuning* comienza con la selección de un modelo preentrenado en un corpus grande y diverso para una tarea específica que no tiene por qué ser la misma que la que se quiere lograr. Luego, el modelo se vuelve a entrenar pero ahora con un conjunto de datos específicos para la tarea que se quiere resolver. De esta manera, se reajustan sus parámetros para mejorar su desempeño en el contexto deseado. Este entrenamiento adicional suele ser más rápido y requiere menos datos que entrenar un modelo desde cero.

Para este proyecto utilizamos *fine-tuning* para la tarea de identificar si un párrafo contiene o no una opinión. Tomamos un modelo entrenado ([ROUBERTA](#), [s.f.](#)) con artículos de prensa uruguayo y a partir de datos nuevos generados por nosotros, lo reentrenamos para poder llegar a un modelo que obtenga la solución que deseamos. Profundizaremos sobre esto en capítulos siguientes.

2.1.11. ROUBERTa

ROUBERTa ([Juan Pablo Filevich, 2024](#)), es un modelo de lenguaje desarrollado a partir de un corpus que comprende aproximadamente 6 GB de textos de periódicos uruguayos, y es el mismo corpus utilizado durante este proyecto. Este modelo fue entrenado durante 30 días en una sola tarjeta gráfica *Nvidia P100*, utilizando la arquitectura *RoBERTa-base* ([Yinhan Liu, 2019](#)), pero con una cantidad de parámetros considerablemente menor que otros modelos *RoBERTa* estándar. Las evaluaciones realizadas en dos tareas de *PLN* muestran que *ROUBERTa* supera a *BETO* ([José Cañete, 2020](#)), un modelo preentrenado de *BERT* en español ampliamente utilizado. Además, al comparar *ROUBERTa* en la tarea de predicción de palabras enmascaradas con otros dos modelos *BERT* multilingües populares, *Multilingual BERT* ([Jacob Devlin, 2019](#)) y *XLM-RoBERTa* ([Alexis Conneau, 2020](#)), se obtuvieron resultados muy buenos en oraciones del ámbito periodístico uruguayo. Estos experimentos demuestran que entrenar un modelo de lenguaje en un corpus específico de dominio puede mejorar significativamente el rendimiento, incluso cuando el modelo es más pequeño y se entrenó con una cantidad de datos notablemente menor que la de los modelos preentrenados más convencionales.

2.1.12. Modelos Generativos

Como se comentó en la sección [2.1.7](#) un tipo de *LLMs* es el modelo *GPT-3*, que corresponde a la familia de modelos generativos. Estos son sistemas

capaces de aprender patrones y la estructura de sus datos de entrenamiento de entrada y generar nuevo contenido con características similares a estos. En *PLN*, se entrenan en grandes corpus de texto, permitiéndoles producir secuencias coherentes en tareas como generación de texto, resumen, o diálogo.

Modelos basados en *transformers* como *GPT-3*, con gran capacidad generativa, han impulsado aplicaciones prácticas, desde *chatbots* hasta asistentes virtuales. Sin embargo, su uso plantea también desafíos éticos, como el riesgo de desinformación, lo que hace necesario un desarrollo responsable en el campo.

Durante este proyecto se realizaron varias investigaciones para utilizar estos modelos en la búsqueda de opiniones y evaluar su factibilidad.

2.2. Estado del arte

Antes de desarrollar nuestra solución, realizamos un análisis de los problemas específicos que debíamos abordar en nuestro escenario. En este análisis, identificamos varios subproblemas claves: detección de opiniones, detección de entidades nombradas, detección de tópicos o clusterización, y análisis de sentimiento.

Para abordar estos subproblemas, revisamos la literatura existente en busca de investigaciones previas que pudieran resolverlos. Aunque no encontramos un estudio previo que abordara de manera integral nuestro escenario, descubrimos múltiples investigaciones que han explorado aspectos específicos de estos subproblemas. A continuación, presentamos algunas investigaciones que nos resultaron relevantes.

2.2.1. Subproblemas

Como se mencionó anteriormente, uno de los problemas a resolver en nuestro proyecto es la detección de opiniones dentro de un texto. Los textos periodísticos suelen tratarse sobre hechos u opiniones, por ende, no toda la información de los textos nos es relevante, sino que es importante identificar opiniones, para luego poder visualizar su evolución a lo largo del tiempo. A continuación, veamos como solucionaron dicho problema en una investigación realizada en la Universidad del Sur de California ([Soo-Min Kim, 2006](#))

La investigación se basa en la identificación de titulares de opinión y temas relacionados con palabras disparadoras en una oración. El proceso se dividió en tres etapas: primero, se creó un conjunto de verbos y adjetivos disparadores de opinión, clasificados por sentimiento. Luego, se buscaron los *frames* semánticos correspondientes en el corpus de *FrameNet*⁴ y, cuando no se encontraban, se usaron técnicas de *clustering* para predecir el *frame* más adecuado. Finalmente, utilizando el *parser* de Charniak (Charniak, 2000), se entrenó un modelo de Máxima Entropía para clasificar los *frames* y determinar el titular y el tema en una oración.

En cuanto a resultados obtenidos, estos arrojaron que el sistema superó significativamente al de referencia en la identificación de titulares y temas, demostrando la efectividad del enfoque.

Por otro lado, además de detectar opiniones en un texto, nuestro problema también requiere identificar el tópico principal del mismo. En la búsqueda de trabajos relacionados, encontramos una investigación realizada en una universidad sueca (Mona Forsman, 2022), donde se enfocan en la detección de tópicos mediante la *clusterización* de *embeddings* generados con *BERT*, utilizando técnicas como *UMAP*⁵ y *HDBSCAN*. El corpus de estudio incluyó un conjunto de 10,000 artículos de noticias en inglés, recolectados de diversos editores.

El *pipeline* propuesto en este trabajo, llamado *BERTopic*, comienza con la *vectorización* del texto a través de *BERT*, seguido por una reducción de dimensionalidad utilizando *UMAP*, simplificando la representación de los datos sin perder la estructura y las relaciones entre los textos. Luego, los autores emplean *HDBSCAN* para realizar el agrupamiento, un algoritmo que se adapta a la densidad de los datos, asignando de manera flexible los textos a los clústeres.

Los resultados de la investigación mostraron clústeres coherentes para los textos de mayor longitud, mientras que aquellos clústeres que agrupaban textos más cortos, presentaban incoherencias.

Otro estudio relevante en el ámbito de análisis de sentimientos realizado en 2015 por la Facultad de Ingeniería de la UdelaR (Guillermo Dufort y Álvarez, 2016), se centró en analizar sentimientos en un corpus de noticias urugua-

⁴<https://framenet.icsi.berkeley.edu>

⁵https://umap-learn.readthedocs.io/en/latest/how_umap_works.html

yas. Esta investigación nos interesó especialmente ya que muchos estudios similares se basan en textos en inglés, y el idioma tiene mucho peso a la hora del análisis del sentimiento en los textos. Además, el trabajo se enfocó en opiniones que mencionaban a políticos uruguayos, lo que añade un valor específico.

El estudio exploró tres enfoques: métodos basados en reglas, aprendizaje automático, y un enfoque híbrido. Los métodos basados en reglas construyeron algoritmos para analizar la sintaxis de las oraciones y asignar polaridad a las palabras usando un diccionario especializado. Los métodos de aprendizaje automático emplearon algoritmos como Support Vector Machines (*SVM*) y Multinomial Naïve Bayes (*MNB*) para clasificar sentimientos, utilizando transformaciones de texto como *TF-IDF*. Por último, el enfoque híbrido combinó características de ambos métodos, resultando en una mejora en el rendimiento del modelo.

Anteriormente presentamos en alto nivel algunas de las investigaciones en las que nos inspiramos para diseñar la solución de nuestro problema, pero para más detalles sobre éstos, otros trabajos y las metodologías utilizadas, se puede consultar el Anexo [A.1](#).

Capítulo 3

Análisis y diseño de la solución

En este capítulo, explicaremos, en primer lugar, el análisis y posterior procesamiento realizado sobre el corpus disponible brindado por el grupo *PLN*. Aplicamos ciertos filtros sobre el corpus, eliminando y agregando ciertos atributos de interés. También, realizamos una limpieza del mismo, intentando obtener finalmente, un corpus sin ruido y con la información necesaria para aplicar luego las distintas técnicas. Por otro lado, nos centraremos en la explicación profunda de lo realizado para cada uno de los diferentes módulos identificados: *Categorización*, *Reconocimiento de Entidades Nombradas*, *Detección de opiniones y entidades* y *Análisis de sentimientos*. Para cada una de estas partes, se ejecutaron ciertas técnicas, permitiendo obtener y comparar sus resultados, y así decidirnos por la aplicación de las adecuadas.

Consideramos este capítulo como la parte central del trabajo, dado que fue la que nos ocupó la mayor cantidad del tiempo, y a través de la cual pudimos aplicar todas las ideas estudiadas y analizadas, obteniendo el producto final que pasa por todos los módulos que se describirán a continuación.

3.1. Análisis del corpus

Para comenzar nuestro trabajo, nos familiarizamos con el corpus que utilizamos para el desarrollo del producto final. Este corpus se presenta en dos versiones: una versión *clean* y otra *raw*. La principal, y aparentemente única diferencia entre ambas, es que la versión *clean* no contiene etiquetas *HTML*,

mientras que la versión *raw* incluye la página completa en su forma original.

El corpus, alojado en el repositorio del grupo de *PLN*¹, está organizado de manera jerárquica: una carpeta por diario, cada una de las cuales contiene subcarpetas por año, y cada año incluye un archivo *JSON* por noticia. El esquema de estos archivos *JSON* es el siguiente:

```
1 {
2   "url": string,
3   "id": int,
4   "date": string,
5   "category": string,
6   "title": string,
7   "keywords": []string,
8   "cover": string,
9   "body": string
10 }
```

Este corpus fue generado mediante técnicas de *scraping* a partir de cuatro diarios virtuales uruguayos: El País², La Diaria³, Montevideo Portal⁴ y El Observador⁵. Optamos por trabajar con la versión *clean*, lo que facilitaba la visualización y el manejo de los datos.

En el corpus *clean* faltan las noticias de *La Diaria*, mientras que en la versión *raw*, aparecen incompletas, mostrando únicamente una vista previa que requería una suscripción paga para acceder al contenido completo. Para obtenerlas, se debe utilizar una credencial de una suscripción. De todas formas, el proyecto se realizó sin estas noticias, pero como explicaremos más adelante, la solución final permite agregar estas y otras más (como noticias más actuales) de manera rápida y eficiente.

La cantidad de noticias por diario varía, con la mayoría de los diarios cubriendo desde el año 2013 hasta 2022, superando las *10.000* noticias por año en promedio. En total el corpus cuenta con más de un millón de noticias.

Se realizó un análisis preliminar para determinar cuáles de los metadatos serían útiles para nuestra investigación futura, y cuáles creíamos que no, para

¹<https://github.com/pln-fing-udelar/uy22>

²<https://www.elpais.com.uy/>

³<https://ladiaria.com.uy/>

⁴<https://www.montevideo.com.uy/>

⁵<https://www.elobservador.com.uy/>

de utilidad para nuestro trabajo.

Además, algunos caracteres como aquellos que representan letras con tilde, están codificados de manera que dificulta la lectura durante el análisis. Para abordar estos problemas, desarrollamos un *script* que, al aplicarse al corpus completo o a un diario específico si se quisiera, limpia las noticias del ruido detectado y las reorganiza según el formato que nosotros creímos conveniente.

El preprocesamiento elimina posibles etiquetas *HTML* restantes, saltos de líneas repetidos ($\backslash n$), y otros elementos similares. A su vez, aplica una limpieza específica dependiendo del diario al que pertenece la noticia en proceso. Por ejemplo, casi todas las noticias de *Montevideo Portal* presentan un ruido común tanto al comienzo como al final del texto, que fue eliminado en esta etapa.

Por otro lado, los artículos tienen distintos formatos para representar las fechas de publicación, dependiendo del diario. Para esto, se implementó un proceso de normalización que transforma todas las fechas a un formato único, lo que permite una mejor organización temporal de los datos y facilita los análisis de fechas posteriores, los cuales son realmente importantes, dado que nuestro trabajo se basa en gran parte, en el estudio de los datos a lo largo de un período de tiempo. Las fechas se normalizaron con el siguiente formato: *YYYY-MM-DD*.

También, debido al análisis e investigaciones que veremos en las siguientes secciones, se transforma el *body* de cada noticia en un arreglo, donde cada entrada es un párrafo del texto de la noticia, cambiando entonces este atributo por otro de nombre *paragraphs*. Brevemente, esto se debe a que las tareas que se aplicarán posteriormente a las noticias, las esperan divididas por párrafos.

Finalmente, para cada diario se generan varios archivos *JSON*. Cada uno incluye un arreglo con todas las noticias de un año. Este formato fue elegido para reducir la cantidad de archivos individuales, dado que cada carpeta contenía más de *10.000* archivos. Aunque esto incrementó el tamaño de cada archivo *JSON*, nos permitió importar rápidamente un año completo de noticias para realizar pruebas.

Cabe aclarar que esta limpieza y formateo de los datos fue modificada en diversas ocasiones a lo largo del proyecto, para agregar algún metadato, modificar el formato o refinar la limpieza.

```

{
  " date ": "01.01.2022 18:29",
  " body ": "\n\n\n\n\nDe acuerdo a la informaci\u00f3n
proporcionada por el Ministerio de Salud P\u00fablica
( MSP ) , hoy se llevaron a cabo 9.945 an\u00e1lisis y
se detectaron 911 nuevos casos ... \n\nAl d\u00eda de
hoy todos los departamentos registran casos activos .
\n\n\n\n\n\n\nMontevideo Portal\n\n\n\n\n\n\n"
}

```

Figura 3.2: Ejemplo de noticia sin preprocesar

A modo de ejemplo, podemos observar la Figura 3.2 en la cual se muestra una noticia de *Montevideo Portal* de enero de 2022, donde se puede observar tanto la utilidad del formateo de las fechas, como de la eliminación de cierto ruido. Por simplicidad, mostramos solo los atributos *date* y *body*, que son los más interesantes de observar para este ejemplo. A su vez, el valor del campo *body* está resumido, con el fin de no mostrar la noticia entera innecesariamente, siendo que la idea es observar un ejemplo de ruido.

Se puede observar que el valor de fecha que se tiene en esta noticia no puede ser comparado con otras fechas de otros diarios si no se realiza el formateo mencionado previamente. Para este caso, no solo vemos que se modifica el formato de la fecha en sí, si no que además, eliminamos la hora, que a los efectos de nuestro trabajo no es relevante. Por otro lado, se puede notar que la noticia culmina con “\n\n\n\n\n\n\n\n\n\n *Montevideo Portal*\n\n\n\n\n\n\n”. Teniendo en cuenta que este segmento de texto u otros aún más extensos aparece en todas las noticias de este diario, es oportuno eliminarlos, ya que pueden modificar el análisis y los resultados de las técnicas aplicadas sobre los artículos. A su vez, también es una buena muestra de lo que sucede con los \n. Como comentamos anteriormente, uno de los procesamientos aplicados, fue la eliminación de dichos caracteres repetidos, lo cual permite, por ejemplo, la separación por párrafos de forma más simple. También se pueden observar algunos caracteres codificados como \u00f3n que representa y es remplazado por el caracter ó.

3.3. Categorización (Tópicos)

El objetivo en esta etapa es intentar asociar a cada noticia una categoría o tópico que permita agruparlas en subgrupos para realizar ciertos análisis, como por ejemplo, buscar opiniones relacionadas exclusivamente con *Deportes*.

3.3.1. Clustering

En primer lugar se comenzó a trabajar con técnicas de *clustering*, dado que la idea de agrupar podría ser útil para nuestro propósito de tener todas las noticias separadas por tópicos. Decidimos empezar con un enfoque de *clustering* no supervisado para aproximarnos a los diferentes tópicos presentes en los diarios.

El primer enfoque fue usar *BERTopic*⁶, un modelo que aplica diversas técnicas para formar *clusters* de tópicos. Este implementa varios pasos y algoritmos, incluyendo *HDBSCAN* para crear los *clusters*. La ventaja de *HDBSCAN* es que no requiere especificar un número específico de *clusters* a diferencia de *K-Means*. Sin embargo, un inconveniente es que las noticias que no se acercan a ningún *cluster* se agrupan en un *cluster -1*.

Comenzamos haciendo pruebas con 1.000 noticias preprocesadas tomadas al azar del corpus, y luego aumentamos la cantidad para analizar cómo se comportaba para una mayor cantidad de artículos y observar qué sucedía particularmente con el tamaño del *cluster -1*. De esta forma, obtuvimos *clusters* (tópicos) que contenían todas las noticias relacionadas entre sí. A primera vista, los resultados parecían satisfactorios, mostrando una alta correlación entre las noticias que pertenecían a los mismos grupos. Sin embargo, observamos que generaba muchos *clusters* —más de 130 para una prueba con 10 mil noticias— y estos eran muy precisos, a tal punto que observamos por ejemplo, un *cluster* específico para noticias relacionadas con el aumento del dólar en un periodo determinado. Además, un alto porcentaje de noticias caía en el *cluster -1*.

En la Tabla 3.1, se muestran dos de las pruebas realizadas, que se corresponden con la menor y la mayor cantidad de noticias con las cuales intentamos, respectivamente. Se realizó una primera prueba utilizando un conjunto de

⁶<https://maartengr.github.io/BERTopic/index.html>

# Noticias	# Clusters	% de noticias en cluster -1
1.000	18	35
20.000	165	42.6

Tabla 3.1: Comparación del tamaño del cluster -1.

1.000 noticias donde se generaron 18 *clusters*. En esta ejecución, el *cluster -1*, que agrupa documentos no asignados a ningún tópico, contenía 348 noticias, lo que representaba el 35 % del total. Continuamos incrementando la cantidad de noticias tomadas para la prueba, donde efectivamente la cantidad de *clusters* seguían aumentando, a la vez que el *cluster -1* crecía aún más.

Aunque *BERTopic* producía buenos resultados en cuanto a la correlación entre las noticias de un mismo *cluster*, realizaba más tareas de las necesarias, y no logramos llegar a una solución concreta.

Decidimos cambiar *HDBSCAN* por *K-Means* para reducir el número de *clusters* a un rango más manejable y que se adaptara más a la cantidad de tópicos que nosotros queríamos obtener. Por lo tanto, con esta técnica, hicimos diversas pruebas modificando la cantidad de *clusters* entre 15 y 30, pensando en categorías más generales como *Deportes*, *Política*, *Educación*, *Medicina*, etc. Incluso utilizamos diferentes modelos para generar los *embeddings*, como *paraphrase-multilingual-MiniLM-L12-v2*⁷ y *ROUBERTA* para observar los distintos resultados, y analizar si dichos cambios afectaban o no en el resultado.

Para utilizar estos distintos números de k , nos basamos en las categorías conocidas por nosotros, y en una investigación hecha sobre los distintos diarios. Nuestra idea era tener entre 10 y 15 tópicos “grandes”, con la posibilidad de tener a su vez, ciertos subtópicos sobre cada categoría “padre”. En primer lugar decidimos comenzar con un número más grande (30) por dos motivos. Por un lado, por la prueba previa hecha con *BERTopic*, que resultaba en más de 130 *clusters*. Al aplicar *K-Means*, en donde se forzarían la cantidad de grupos, el cambio podía ser muy brusco si lo ejecutábamos para obtener tan solo 15 o 10, y no nos habilitaría a realizar un análisis de cómo se iba comportando al disminuir la cantidad. Por otro lado, al ver que cada grupo

⁷<https://huggingface.co/sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2>

solía ser bastante específico en cuanto a su contenido, nos parecía oportuno empezar probando con 30, pensando que quizás el resultado nos iba a ofrecer un *cluster* por cada subtópico posible (aproximadamente), en lugar de uno por cada categoría “padre”.

En las distintas pruebas, el comportamiento fue similar: se obtenían varios *clusters* que podían *mapearse* relativamente directo a un tópico/subtópico, pero también otros tantos que no eran fácilmente identificables, ni tan coherentes ni correlativos en sí mismos.

Concluimos que tanto *BERTopic* como su variación con *K-Means* producían buenos resultados, aunque con limitaciones. *BERTopic* generaba demasiados *clusters*, mientras que *K-Means*, a pesar de obtener un número específico de estos, agrupaba las noticias de manera no completamente coherente, resultando en *clusters* que contenían miles de noticias sin ser tan parecidas entre sí, lo que dificultaba la identificación de un tópico claro. En la Tabla 3.2 se puede ver un ejemplo de algunos *clusters* obtenidos aplicando *K-Means* y el porcentaje de noticias de los tres tópicos con mayor cantidad de estas dentro de cada uno.

3.3.2. Etiquetado manual

Cada noticia del corpus contiene un campo *category*, extraído del *HTML* o de la *URL* de la noticia, que depende de cada diario. Mas allá de la existencia de diferencias de estos valores entre los distintos diarios, o incluso algunas diferencias dentro de un mismo diario, se puede realizar un *mapeo* manual de todos los valores del metadato *category* del corpus a un listado común de tópicos definidos. Se optó entonces por normalizar manualmente las categorías encontradas a 15 tópicos generales, y luego intentar identificar subtópicos. Ordenamos las noticias por diario y por la categoría, y comenzamos a *mapear* manualmente las categorías a nuestros tópicos. Se definieron los tópicos a medida que avanzamos entre los distintos valores de *category* disponibles, teniendo en cuenta para cada uno, los tópicos definidos hasta el momento, y el tamaño que tenía el nuevo valor analizado (cantidad de noticias con dicha categoría asociada). Según esos valores, decidimos si valía la pena crear un nuevo tópico en nuestra tabla definitiva, referente a este valor, o si era preferible *mapear* esta categoría a algún tópico ya definido previamente. Un ejemplo de esta normalización fue mapear “*ovacion/futbol*” de el diario *El País* a “*Deportes*”. Los tópicos finales se pueden observar en

Clúster	Categoría	Porcentaje
0	Vida Actual	55.8 %
	Opinión	21.4 %
	Economía y Empresas	15.5 %
1	Economía y Empresas	34.0 %
	Nacional y Política	28.2 %
	Opinión	17.6 %
2	Vida Actual	91.9 %
	Deportes	6.2 %
	Otras	0.3 %
3	Salud	44.2 %
	Mundo	23.7 %
	Vida Actual	17.0 %
4	Nacional y Política	49.0 %
	Mundo	24.0 %
	Personajes	5.8 %
7	Deportes	55.5 %
	Espectáculos y Cultura	4.5 %
	Personajes	3.7 %
8	Nacional y Política	33.0 %
	Vida Actual	31.6 %
	Espectáculos y Cultura	11.3 %

Tabla 3.2: Porcentaje de noticias por categoría para 8 *clusters* de *K-Means* con $k=15$

la Tabla 3.3.

El t3pico *Otras* engloba aquellas noticias cuya categor3a era demasiado espec3fica, por lo que no ameritaba crear un nuevo t3pico. En general, se trataba de una categor3a con un muy pocas noticias respecto al tama1o del corpus, por lo que cre3mos pertinente relacionar estas noticias con el t3pico *Otras*. Por otro lado, el t3pico *Desconocido* contiene aquellos textos en los cuales no se ten3a un valor del metadato *category* asociado. Al igual que para el caso de *Otras*, se obtuvo un grupo peque1o al compararlo con el tama1o del corpus.

A partir del listado final de t3picos, se *mape3* cada noticia a uno de ellos basando dicho *mapeo* en el valor del atributo categor3a perteneciente a la metadata de cada noticia. En la Tabla 3.3 se pueden ver los t3picos y la cantidad de noticias en cada uno.

Categor3a	Cantidad
Ciencia y Tecnolog3a	40138
Deportes	185258
Espect3culos y Cultura	88069
Vida Actual	25092
Personajes	19355
Opini3n	25303
Nacional y Pol3tica	218360
Agro	19132
Econom3a y Empresas	57611
Externas	20938
Mundo	108957
Otras	21374
Salud	20737
Cocina	883
Desconocido	43823

Tabla 3.3: Cantidad de Noticias por T3pico

3.3.3. Subcategorizaci3n

Las noticias de *El Pa3s*, contaban con valores que indicaban subcategor3as, como *ovacion/futbol*, *ovacion/tenis*, *ovacion/basquetbol*, etc. Sin embargo,

Categoría	0	1	2	3	4	5	6
Sociedad	83	948	29	460	753	112	79
Servicios	268	84	13	13	139	1	2
Política	5	990	1951	142	1400	2267	314
Educación	1	118	25	19	324	28	25
Judiciales	-	128	12	271	187	59	962
Policiales	-	97	3	1159	70	17	380
Sindicales	-	68	35	15	329	44	6

Tabla 3.4: Clustering sobre Categorías con datos diversos

realizar el mismo procedimiento de clasificación hecho previamente parecía muy complejo, dado que solamente *El País* tenía valores de este estilo.

Basándonos en esto, intentamos nuevamente realizar *clustering*, pero ahora para obtener subcategorías dentro de los tópicos generales. Sin embargo, el resultado no fue positivo; las noticias no se agruparon de manera coherente en subcategorías, quedando mezcladas y situándonos nuevamente en un problema similar al que nos sucedía con las categorías generales.

La Tabla 3.4 es un ejemplo de una prueba de *K-Means* sobre un conjunto de noticias de *Nacional y Política* del diario *El País*, donde cada grupo es un *cluster* resultado. Cada una de las noticias tomadas tenía como categoría alguno de los valores que se ven en dicha Tabla. Los números se corresponden con la cantidad de noticias asociadas a cada categoría dentro de cada *cluster* resultante. Como podemos ver en el resultado obtenido, no se obtiene ningún *cluster* con una subcategoría bien definida y predominante. Por el contrario, vemos que hay más de una que predomina en cada grupo.

Identificamos que este problema seguramente se debía a la predominancia de algunas subcategorías, como *Fútbol* en el caso de *Deportes*, que superaba ampliamente en cantidad a otras.

Realizando una prueba con 500 noticias para cada deporte, se comienza a observar este problema. Tal como se puede ver en la Tabla 3.5, se obtuvieron *clusters* más coherentes. Para esto, seleccionamos de *El País*, noticias al azar de cada una de las siguientes subcategorías: *Tenis*, *Futbol*, *Basquetbol*, *Motor* y *Rugby*, y aplicamos *clustering* sobre dicho conjunto. El resultado obtenido contiene tres *clusters* bien definidos en cuanto a su contenido (*Te-*

Cluster	Básquetbol	Fútbol	Tenis	Rugby	Motor
0	274	199	88	223	84
1	11	-	363	-	2
2	89	-	-	-	-
3	-	1	1	-	290
4	126	300	48	277	124

Tabla 3.5: Clustering sobre Deportes con categorías balanceadas

nis, *Basquetbol* y *Motor*), y otros dos en los cuales predominan dos de las subcategorías.

Debido a la tendencia de las subcategorías a dominar dentro de una categoría general, y considerando que no conocíamos todas las subcategorías posibles para todos los tópicos que teníamos, o que algunos diarios no las manejaban claramente, decidimos no continuar con la búsqueda de subcategorías, para poder continuar con el trabajo de los distintos módulos, siempre con la idea de poder obtener una solución inicial completa al proyecto.

En su lugar, se planteó la necesidad de encontrar una solución diferente que permita identificar un tópico más específico dentro de un tópico general. Por lo tanto, se decidió agregar un filtro titulado “Palabra clave” en el cual el usuario puede agregar una cadena de caracteres que se utiliza como filtro en la búsqueda de noticias, mostrando solamente aquellas que contengan esta cadena.

3.4. Reconocimiento de Entidades Nombradas (NER)

Las entidades mencionadas en las noticias pueden ser variadas, como personas, organizaciones, lugares, entre otras. Sin embargo, nuestro objetivo radica especialmente en encontrar opiniones emitidas por diferentes personas, por ende, nos centramos solamente en ello.

Por otro lado, dado que en las noticias de prensa una entidad puede aparecer múltiples veces y de distintas maneras, una tarea adicional al reconocimiento de entidades es la normalización de las diferentes apariciones. De esta forma, se puede identificar si cierta entidad expresa una opinión en diferentes

párrafos, estando referenciada de diferente manera.

Entendemos que la forma en que están escritas las noticias facilita el trabajo de las librerías de reconocimiento de entidades, ya que generalmente respetan las mayúsculas en los nombres propios, a diferencia de otros tipos de texto como los *tweets*, donde la escritura puede ser más informal, menos estructurada y los resultados de las librerías pueden verse afectados.

Se exploraron y probaron distintas librerías como por ejemplo *SpaCy*⁸, *Flair*⁹ y *Polyglot*¹⁰. Si bien con las tres librerías vimos resultados coherentes, devolviendo correctamente las entidades mencionadas en los textos, nos decidimos por la utilización de *SpaCy*, ya que se obtienen los resultados esperados con mayor velocidad.

SpaCy es una biblioteca que ofrece múltiples herramientas para el procesamiento de lenguaje natural y puede utilizarse con diversos modelos. Tras realizar múltiples pruebas, notamos que el modelo entrenado con noticias en español *es-core-news-lg*¹¹ arrojaba los mejores resultados.

Por otro lado, se observa una mejora en los resultados devueltos por la librería al procesar cadenas de texto de menor longitud. Por ello, para nuestro escenario, es más beneficioso procesar las noticias por párrafos en lugar de procesarlas completas.

En la Tabla 3.6, se muestran las entidades incorrectamente reconocidas por la librería al utilizarla sobre la noticia completa titulada '*Nuevo enfoque sobre drogas*' de el diario *El País* en el año 2013 que se puede ver completa en el Anexo 4.

Observando los valores se puede observar una cantidad significativa de entidades reconocidas de forma errónea, lo que podría generar problemas futuros. Por otro lado, en la Tabla 3.7 se pueden observar las entidades reconocidas de forma incorrecta al aplicar la librería en cada párrafo por separado y unificar las entidades obtenidas. Vemos que la longitud de la Tabla 3.7 es sensiblemente menor a la de la Tabla 3.6, lo cual confirma la mejoría de los resultados al procesar las noticias por párrafos.

⁸<https://spacy.io/models/es>

⁹<https://huggingface.co/flair/ner-english>

¹⁰<https://pypi.org/project/polyglot/>

¹¹https://huggingface.co/spacy/es_core_news_lg

Formato	Entidades (Nombre, Tipo)
Noticia completa	(‘Un tema que ya se encuentra en el’, MISC), (‘Informe’, LOC), (‘Jefas’, MISC), (‘Jefes de Estado’, MISC), (‘Presidente colombiano’, ORG), (‘debía’, PER), (‘Lo que tenemos que hacer’, MISC), (‘agregó’, PER), (‘Una discusión’, MISC), (‘Antigua’, LOC), (‘Estupefacientes’, PER), (‘año’, MISC), (‘El Informe’, LOC), (‘qué’, MISC), (‘Lo más importante’, MISC), (‘Ello’, PER), (‘El término ”hemisférico’, MISC), (‘Quizás’, LOC), (‘El estudio’, MISC)

Tabla 3.6: Entidades incorrectamente reconocidas por la librería

Formato	Entidades (Nombre, Tipo)
Separado por párrafos	(‘Jefas y Jefes de Estado’, MISC), (‘Antigua’, LOC), (‘El estudio’, MISC), (‘El término’, MISC)

Tabla 3.7: Entidades incorrectamente reconocidas por párrafos

La solución final se basó en la creación de un algoritmo que procesa la noticia por párrafos, reconoce las personas mencionadas haciendo uso de la librería *spaCy* con el modelo *es-core-news-lg* y normaliza las entidades encontradas.

El algoritmo identifica todas las variaciones de una entidad y las asigna a un nombre normalizado. Un ejemplo sencillo de normalización podría ser la aparición de Oscar Andrade, quien podría mencionarse posteriormente solo como Oscar o Andrade. Por otro lado, un caso más complejo es la resolución de correferencias, como por ejemplo, si Oscar Andrade fuera mencionado como “el senador”. Este es un desafío mayor y queda por fuera del alcance de este proyecto, ya que el modelo no reconoce a “el senador” como una entidad, y se necesitaría un procesamiento adicional para lograr la normalización.

En el *pseudo-código* presentado en el Algoritmo 1 se puede visualizar en alto nivel el algoritmo creado en este módulo, el cual incluye la utilización de la librería para la obtención de las entidades y su normalización. A su vez, en el *pseudo-código* presentado en el Algoritmo 2 se observan los procedimientos auxiliares utilizados por este.

Algorithm 1 NER & Normalización de entidades en noticias

```
1: procedure PROCESARNOTICIAS(noticias)
2:   for cada noticia en noticias do
3:     entidades  $\leftarrow$  set()
4:     for cada párrafo en noticia do
5:       personasReconocidas  $\leftarrow$  SpacyNER(parrafo, PER)
6:       for cada entidad en personasReconocidas do
7:         normalizarEntidad(entidad, entidades)
8:       end for
9:     end for
10:  end for
11: end procedure
```

El algoritmo descrito previamente procesa cada noticia manteniendo un conjunto de entidades previamente normalizadas, evitando repeticiones. Para cada párrafo, se utiliza la librería para obtener las entidades de tipo persona y se invoca una función auxiliar que normaliza las entidades reconocidas. Esta función toma cada palabra de la entidad y busca coincidencias parciales en las entidades normalizadas hasta el momento. Si se encuentra una coincidencia única, se selecciona como la entidad normalizada; de lo contrario, se

Algorithm 2 Algoritmos auxiliares

```
1: procedure NORMALIZAR_ENTIDAD(entidad, entidadesPrevias)
2:   indexes  $\leftarrow$  list()
3:   palabras  $\leftarrow$  dividir(entidad)
4:   for cada palabra en palabras do
5:     if indexes está vacía then
6:       indexes = isSubstring(palabra, entidadesPrevias)
7:     else
8:       for cada index en indexes do
9:         if palabra no está en entidadesPrevias[index] then
10:          Eliminar index de indexes
11:        end if
12:      end for
13:    end if
14:  end for
15:  if indexes tiene exactamente un elemento then
16:    Obtener entidadAsociada de entidadesPrevias[indexes[0]]
17:    palabrasEntidadAsociada  $\leftarrow$  dividir(entidadAsociada)
18:    if cantidad de palabras es mayor que la de palabrasEntidadAsociada then
19:      return entidad
20:    else
21:      return entidadAsociada
22:    end if
23:  else
24:    return entidad
25:  end if
26: end procedure
27: procedure ISSUBSTRING(palabra, entidadesPrevias) Retorna una lista
    con los índices en los que palabra es substring en entidadesPrevias
28: end procedure
```

utiliza la entidad original. Finalmente, cada entidad normalizada se agrega al conjunto. Así, todas las entidades de una noticia quedan unificadas bajo su forma más completa, facilitando la asociación de opiniones a la misma entidad.

3.5. Detección de opiniones

Una de las tareas más importantes en este trabajo es identificar cuándo se está expresando una opinión. A continuación se discuten diferentes enfoques e investigaciones realizadas para poder resolver este problema.

3.5.1. Enfoques basados en análisis de sentimientos y en listas de indicadores de opinión

Un primer enfoque es detectar la presencia de opiniones a partir del análisis de sentimiento de las noticias. Mediante el uso de librerías para determinar el sentimiento en textos, como es el caso de *pysentimiento*¹² se puede estimar si en una determinada noticia o párrafo se está expresando un sentimiento positivo, neutro o negativo. Esto podría servir como indicador de existencia de opinión en caso de tener un sentimiento diferente a neutro.

Al analizar los resultados de aplicar el análisis de sentimientos sobre las noticias, detectamos que no existe una relación directa entre el sentimiento y la presencia de una opinión en un texto. Por lo tanto, se debe combinar el análisis de sentimientos con una lógica adicional. Una heurística posible para determinar si existe una opinión podría ser:

- el sentimiento es positivo o negativo con una precisión mayor a cierto valor,
- en el párrafo se menciona alguna entidad nombrada de tipo persona, y
- existe algún verbo de discurso, como “dijo” o “expresó”.

Finalmente se decidió descartar el uso del análisis de sentimientos para detectar opiniones y proceder a la evaluación de otros enfoques como el mencionado en la sección 2.2 donde se utiliza una lista de palabras claves disparadoras de

¹²<https://pypi.org/project/pysentimiento/0.5.2rc3/>

opiniones y *FrameNet*¹³ para obtener sentencias donde ocurra una opinión y extraer así el *holder*, es decir la persona que esta dando y sosteniendo la opinión y el *topic* es decir el tema de ella (Kim y Hovy, 2006). Sin embargo esta metodología también fue descartada debido a la ausencia de un *FrameNet* en español lo suficientemente robusto como para implementar este enfoque.

3.5.2. Modelos generativos GPT

Otro enfoque es utilizar un modelo de la familia *GPT*, como *ChatGPT*, para extraer opiniones de las noticias. Pedir al modelo que identifique opiniones por sí solo, es decir, pasándole una noticia para que señale las opiniones que contenía, si las hubiera, con breve *prompt* indicando lo que se espera como salida, no funciona bien, ya que por la naturaleza de *ChatGPT* su respuesta es muy amplia y no del todo correcta, pero esto se debe a que el *prompt* enviado es muy breve o con falta de contexto.

Por tanto mediante *prompts* sucesivos, aprovechando la capacidad de *ChatGPT* para tomar los *prompts* anteriores como referencia para sus nuevas respuestas se puede obtener una mejor respuesta al agregar este contexto adicional. Continuando con la idea del artículo anteriormente mencionado generamos una lista de palabras "disparadoras" que tienden a aparecer cuando se está expresando una opinión, y acá es de gran ayuda nuevamente la formalidad y estructura que tienen las noticias. Para generar esta lista extrajimos todas las palabras de las noticias de 2013 de El País utilizando una expresión regular para identificar las palabras que preceden a una frase entre comillas dobles. Luego, le pedimos a *ChatGPT* que filtrara esta lista para quedarse solo con las palabras que actúan como "disparadores de opinión". Además le pedimos que intente agregar algunas nuevas a partir del resultado anterior y finalmente corregimos a mano el resultado. Un extracto de esta lista se puede ver en el Listing 3.1.

```
1 [
2     "aclaró",
3     "agregó",
4     "afirmó",
5     "admitió",
6     "anunció",
7     "aseguró",
```

¹³<https://framenet.icsi.berkeley.edu/>

```

8     "comentó",
9     "confirmó",
10    "declaró",
11    "dijo",
12    "explicó",
13    "informó",
14    "indicó",
15    "manifestó",
16    "señaló",
17    ...
18 ]

```

Listing 3.1: Algunos ejemplos de la lista de palabras disparadoras.

A continuación, seleccionamos una noticia y generamos un resultado esperado en un formato *JSON*, donde las claves son las entidades que opinan y los valores son arreglos con las opiniones en dicha noticia que se puede observar en el Listing 3.2.

```

1 {
2   "José Mujica":[
3     "dijo que no hay que derogar el delito de abuso de
4     funciones
5     hasta que la justicia se expida sobre el caso Pluna",
6     "señaló que el préstamo para la cooperativa Alas U
7     está en
8     suspenso porque no hay respaldo jurídico para
9     otorgarlo",
10    "destacó la necesidad de una figura jurídica para
11    otorgar el
12    préstamo y la importancia de arriesgar para que la
13    gente pueda
14    trabajar, en comparación con el seguro de paro",
15    "manifestó que el tratamiento de los fondos como '
16    plata pública'
17    requiere precaución para evitar acusaciones de abuso
18    de
19    funciones",
20    "criticó la idea de eliminar el delito de 'abuso de
21    funciones'
22    sin tener una alternativa, mencionando la necesidad
23    de esperar
24    ante acusaciones actuales",
25    "reconoció el error en la gestión del caso Pluna,
26    atribuyendo el

```

17 fallo a la ansiedad por solucionar problemas
laborales y
18 turísticos",
19 "cuestionó el asesoramiento recibido en el caso Pluna
, comparando
20 la situación con errores anteriores en el ámbito
financiero
21 uruguayo",
22 "aclaró que a Fernando Lorenzo no se le pidió
renunciar, sino que
23 la renuncia fue decisión de Lorenzo",
24 "pidió dejar tranquilo al Banco República, destacando
su
25 importancia para la economía uruguaya y la necesidad
de esperar
26 hasta enero para discutir temas pendientes"
27],
28 "Telemundo":null,
29 "Mujica":[
30 "manifestó su postura de arriesgar en decisiones
financieras para
31 beneficiar el empleo frente a los costos del seguro
de paro",
32 "reiteró su enfoque en la responsabilidad de manejar
fondos
33 públicos y las implicaciones legales de sus
decisiones",
34 "expresó su perspectiva sobre los errores y
aprendizajes en la
35 gestión de crisis, especialmente en relación con el
caso Pluna",
36 "comentó sobre el proceso de toma de decisiones y el
papel de los
37 asesores, reflexionando sobre su confianza y los
resultados",
38 "destacó la importancia de la prudencia y la
planificación en la
39 administración de recursos clave como el Banco Repú
blica"
40],
41 "Peirano":null,
42 "Fernando Lorenzo":null,
43 "Banco República":null

44 }

Listing 3.2: Respuesta esperada para ChatGPT escrita por nosotros.

Le pedimos entonces a *ChatGPT* que realizara la misma tarea, proporcionándole la lista de palabras como disparadores de opinión, el ejemplo de salida, formato que esperábamos y la lista de entidades que el modelo de *NER* reconoció para la noticia nueva. Y obtuvimos un resultado satisfactorio, donde la respuesta final de *ChatGPT* sostiene el formato esperado de salida y contiene las opiniones mencionadas por las diferentes entidades. Esta respuesta se puede apreciar en el Listing 3.3. Además para ver un ejemplo de la secuencia de *prompts* utilizados en este y otros casos se puede consultar el Anexo A.5.

```
1 # ChatGPT
2 {
3   "Único": [
4     "Se declaró en 'estado de alerta' tras un hecho de
5     violencia en el
6     Hogar Ser.",
7     "Señaló que la situación es preocupante y se necesitan
8     respuestas
9     rápidas para evitar que estos hechos vuelvan a ocurrir.",
10    "Aclaró que el ingreso de nuevos funcionarios ha tenido
11    un impacto
12    positivo en las condiciones de trabajo, pero esto se ve
13    empañado por
14    el aumento de adolescentes privados de libertad y el
15    hacinamiento.",
16    "Considera importante la intervención de las autoridades
17    para nuevas
18    instalaciones y descomprimir el hacinamiento.",
19    "Indicó que no descartan ninguna medida para proteger la
20    integridad
21    física de los trabajadores."
22  ],
23  "INAU": null,
24  "Colonia Berro": null,
25  "Joselo López": [
26    "Narró el incidente donde un trabajador fue agredido por
27    un
28    adolescente con agua hirviendo.",
29    "Detalló el contexto del enfrentamiento verbal que
30    precedió al ataque al trabajador."
```

```
22 ]
23 }
```

Listing 3.3: Ejemplo de respuesta de ChatGPT exitosa.

3.5.3. Modelos generativos de uso libre

A pesar de que *ChatGPT* es una herramienta gratuita que se puede usar desde un navegador, no es gratuito para su uso mediante una *API* en un entorno de desarrollo, donde se necesitaría automatizar el proceso de identificar opiniones en más de un millón de noticias. Por lo tanto, es necesario el uso de otros modelos que pueden ser ejecutados localmente de manera gratuita. Por lo tanto exploramos modelos como *Llama 2*¹⁴ de la familia de modelos *Llama* distribuidos gratuitamente por Meta¹⁵ y *Mistral*¹⁶. Aunque no encontramos una versión en línea de *Llama 2*, como la que se cuenta para *ChatGPT*, sí pudimos probar una versión de *Mistral 70B*, obteniendo resultados muy similares a los de *ChatGPT* realizando la misma secuencia de *prompts* descrita anteriormente. Esta respuesta de *Mistral* se puede observar en el Listing 3.4.

```
1 # MistralAI
2 {
3   "Único": [
4     "se declaró en 'estado de alerta' tras el último 'hecho
5     de violencia
6     ocurrido en el Hogar Ser'",
7     "señaló que 'la situación es preocupante' y se necesitan
8     'rápidas
9     respuestas que aseguren que este tipo de hechos no
10    vuelvan a
11    ocurrir'",
12    "aclaró que 'el ingreso de 300 nuevos funcionarios a la
    Institución
    ha generado un impacto positivo en las condiciones de
    trabajo'",
    "consideró importante 'la inmediata intervención de las
    autoridades, en la concreción de nuevas instalaciones, que
    aumenten la capacidad
    locativa, y descomprima el hacinamiento que existe en
    todos los
```

¹⁴<https://llama.meta.com/llama2/>

¹⁵<https://about.meta.com/>

¹⁶<https://mistral.ai/>

```

13     centros'",
14     "señaló que no descarta 'ninguna medida mientras siga en
15     peligro la
16     integridad física de los trabajadores que este sindicato
17     representa'"
18 ],
19 "INAU": null,
20 "Colonia Berro": null,
21 "Joselo López": [
22     "relató el enfrentamiento verbal entre un joven en
23     rehabilitación y
24     un trabajador, que culminó con agua hirviendo arrojada en
25     el rostro
26     del trabajador"
27 ]
28 }

```

Listing 3.4: Ejemplo de respuesta de Mistral AI exitosa.

Para poder correr un modelo de este tipo localmente, es necesario el uso de una herramienta como *Ollama*¹⁷. *Ollama* permite correr una gran variedad de modelos localmente solamente descargándolo. Al iniciar un modelo, *Ollama* levanta un servidor con el cual se conecta y permite interactuar con el modelo. A partir *ollama-python*¹⁸ se puede utilizar un lenguaje de programación para conectarse y automatizar todo tipo de pruebas.

Se probaron dos familias de modelos, *Llama* y *Mistral*, en sus versiones de *70B* y *7B*. Los modelos de *70B* tienen una gran demanda de recursos computacionales debido a la cantidad de parámetros (70 mil millones), lo que dificultó su ejecución en recursos locales. Se intentó ejecutar estos modelos en el clúster de *Cluster-uy*¹⁹, pero el acceso a *GPUs Nvidia A100*, que son ideales para este tipo de modelos, es limitado y no siempre disponible.

Dado que no fue posible acceder a las *GPUs A100*, se optó por utilizar *GPUs Nvidia Tesla P100*, que son menos potentes pero superiores a las de recursos locales. Sin embargo, las *Tesla P100* no son suficientes para ejecutar eficientemente los modelos *70B* debido a la gran carga computacional que requieren. Por esta razón, se decidió trabajar con las versiones de *7B* de los modelos, que son más manejables en términos de recursos.

¹⁷<https://ollama.com/>

¹⁸<https://github.com/ollama/ollama-python>

¹⁹<https://cluster.uy/>

Particularmente las versiones reducidas como el *Mistral 7B*, tienden a mezclar idiomas en sus salidas, predominantemente entre el español y el inglés. Una consecuencia común cuando se reduce la cantidad de parámetros. Esta inclinación hacia el inglés afectó negativamente la coherencia y precisión de los resultados, especialmente cuando se trataba de procesar contenido en español de manera exclusiva, como se puede observar en el Listing 3.5.

```
1 # Ollama
2 I. Senadores and Their Opinions on the Marijuana Legalization
   Bill
3
4 1. Luis Alberto Heber: questioned the criterion for plants
   instead of
5 seven or eight.
6 2. Alfredo Solari: raised concerns about potential damages
   from marijuana
7 and warned against regional tourism focus on cannabis.
8 3. Larrañaga (Jorge): labeled the marijuana bill as "
   inconstitucional"
9 and warned of potential market control issues.
10 4. Luis Gallo (Frente Amplio): defended the constitutionality
   of the law,
11 emphasizing its health benefits for the population.
12 5. Enrique Rubio: insisted that it is not an experiment but a
   public
13 policy.
14 6. José Mujica: referred to the marijuana bill as an "
   experiment" in the
15 past.
16 7. Pedro Bordaberry: disagreed with Larrañaga's assessment,
   stating that
17 the Uruguayan people are not being experimented upon.
18 8. Luis Alberto Lacalle: saw it as a step on a continuum and
   referenced
19 Vázquez's statements about cocaine.
20 9. Vázquez (not mentioned in the text): made remarks about
   marijuana
21 being an "experiment" in the past.
22
23 Opinions from the given entities:
24
25 {
26   "Luis Alberto Heber": "Criticized the number of cannabis
   plants allowed",
27   "Alfredo Solari": "Raised concerns over potential damages
```

```

28   and regional
29   focus on cannabis tourism",
30   "Solari": "Raised concerns over potential damages and
31   regional focus on
32   cannabis tourism",
33   "Jorge Larrañaga": "Labeled it as unconstitutional and
34   warned about
35   market control issues",
36   "Luis Gallo (Frente Amplio)": "Defended its
37   constitutionality and
38   emphasized health benefits",
39   "Enrique Rubio": "Insisted it is not an experiment but a
40   public policy",
41   "José Mujica": "Referred to it as an \"experiment\" in the
42   past",
43   "Pedro Bordaberry": "Disagreed with Larrañaga's assessment
44   ",
45   "Luis Alberto Lacalle": "Saw it as a step on a continuum
46   and referenced
47   previous statements about cocaine"
48 }

```

Listing 3.5: Ejemplo de respuesta mala de Mistral 7B con Ollama.

Aunque los resultados obtenidos con el modelo 7B a través de *prompting* no fueron satisfactorios, se optó por intentar realizar un *fine-tuning* sobre este modelo. El objetivo es ajustar el modelo específicamente para la tarea de identificar y analizar opiniones en noticias, buscando mejorar su rendimiento en este contexto.

3.5.4. Fine-tuning a modelos existentes

Este enfoque final de realizar *fine-tuning* a un modelo existente se propone tanto para los modelos generativos previamente mencionados como para el modelo *ROBERTa* (Juan Pablo Filevich, 2024). Este fue creado a partir del mismo corpus de noticias uruguayas con el cual contamos, lo cual propone ciertas ventajas ya que el modelo ya ha sido expuesto a las características del corpus específico que estamos analizando. Al hacer *fine-tuning* sobre un modelo entrenado en el mismo corpus, se mejora la capacidad del modelo para captar patrones en el conjunto de datos, lo que potencialmente puede aumentar la precisión en las tareas.

En ambos casos, ya sea utilizando un modelo 7B o *ROBERTa*, realizar

un *fine-tuning* permite ahorrar tiempo y procesamiento, entre otras cosas. Para poder realizar *fine-tuning* sobre cualquiera de los modelos, se necesita un aprendizaje supervisado. Esto implica disponer de un conjunto de datos donde tanto la entrada como la salida esperada estén definidas, permitiendo que el modelo aprenda a realizar la tarea deseada sobre nuevas entradas para las cuales no tiene la salida conocida. Para este caso particular, se debe contar con párrafos de noticias como entrada, y la salida debe indicar si el párrafo contiene una opinión o no.

Por tanto se generó un conjunto de datos con entradas y salidas esperadas para la tarea, definiendo un formato de etiquetado que añadía indicadores al comienzo y al final de cada opinión, aunque solo se etiquetó la opinión, ya que en muchas noticias no se menciona la entidad que la expresa cerca de la misma. Este enfoque se centró en identificar las opiniones, dejando para más adelante la discusión sobre la asociación entre estas y las entidades. En una primera iteración, se utilizó un *script* para seleccionar 482 noticias, intentando que el 90 % incluyera opiniones, basándose en una lista de palabras disparadoras. Al no alcanzar este objetivo, se ajustó el *script* en la segunda iteración para asegurar que todas las noticias contuvieran opiniones, añadiendo criterios de inclusión de entidades políticas, lo que resultó en 108 noticias adicionales. Finalmente, se creó un conjunto de validación con 90 noticias de prensa uruguaya, distribuidas equitativamente entre las que contenían y no contenían opiniones. Nos aseguramos que este conjunto no hubiera sido utilizado previamente en el proceso de entrenamiento, seleccionando noticias manualmente desde internet y del año 2024. Un ejemplo de etiquetado se puede observar en el Listing 3.6 donde cada opinión aparece rodeada por las etiquetas `< OP >` y `< /OP >`. Además, se puede apreciar que se marcan tanto opiniones expresadas entre comillas como no. Adicionalmente en la Tabla 3.8 se pueden apreciar las cantidades de los conjuntos de entrenamiento y validación finales.

```
1
2 <OP>El expresidente José Mujica habló sobre cómo lleva el
3 tratamiento para combatir el cáncer de esófago que padece, y
4 aseguró que está \"cansado\" y \"jodido\".</OP>\n <OP>\"Ando
5 jodido, a la miseria. El tratamiento ese que me están
6 haciendo es como una viava todos los días, entonces no puedo
7 con las patas\", aseguró el exmandatario</OP> en entrevista
8 con Telemundo (Canal 12).
9 <OP>\"Cansado. Estoy como panadero antiguo que hacía reparto
```

Conjunto	# Noticias	# Párrafos	# Párrafos con Opinión
Entrenamiento	482	4067	781
Validación	90	588	159

Tabla 3.8: Conjuntos de entrenamiento y validación

```

10 y paraba en todas las paradas. Más o menos estoy igual",
11 graficó Mujica.</OP>.
12 Tras el anuncio del diagnóstico del expresidente, su doctora
13 personal, <OP>Raquel Pannone, dijo que los resultados del
14 tratamiento se irán valorando en el correr de los días y que
15 se estima que durará, aproximadamente, un mes o mes y medio.
16 </OP> Mujica recibe radioterapia de lunes a viernes, de cinco
17 a diez minutos por sesión. Tanto la cirugía como la
18 quimioterapia fueron descartados para tratar el cáncer del
19 líder del Movimiento de Participación Popular (MPP), debido
20 a su edad -Mujica cumplió 89 el pasado lunes, aunque días
21 atrás confesó que tiene un año más- y al hecho de que hace
22 más de 20 años padece una enfermedad inmunológica,
23 vasculitis y una insuficiencia renal.

```

Listing 3.6: Ejemplo de etiquetado.

3.5.5. Fine-tuning a modelos generativos

Manteniendo la misma idea general del problema —determinar si un párrafo contiene una opinión o no— decidimos abstraernos un poco y transformarlo en un problema de clasificación binaria. Es decir, la idea es tener un modelo que, dado un párrafo como entrada, indique si contiene alguna opinión o si no la contiene.

Aunque continuamos realizando pruebas con los modelos de *Mistral* y *Llama 2*, tanto localmente como en el clúster usando *GPUs P100*, los tiempos de respuesta eran lentos. A pesar de estar utilizando modelos más ligeros, como los de *7B*, siguen siendo modelos pesados que requieren muchos recursos computacionales. Una opción es reducir la representación en bits de los modelos. Es decir, la representación de los pesos, que originalmente está en 32 bits, se puede bajar a 8 o 4 bits, reduciendo así el tamaño del modelo y el uso de memoria.

Por tanto, previo a realizar *fine-tuning*, hicimos algunas pruebas previas con

el modelo reducido a 8 bits seleccionando una porción de los párrafos que habíamos etiquetado y que ya tenían la salida correspondiente (con o sin opinión). Generamos un *prompt* que consistía en la entrada del párrafo junto con la pregunta, esperando que la respuesta fuera simplemente “sí” o “no”. A continuación, pasamos todos estos párrafos con el *prompt* al modelo en 8 bits para evaluar sus resultados. Este *prompt* se puede ver en el Listing 3.7.

```
1 El siguiente parrafo contiene alguna opinion? Responde con si
  o no
2 solamente: El diputado Sergio Mier (Liga Federal) dijo en
  rueda de
3 prensa que Miranda "pidió encarecidamente que cuando se
  desarrolle el
4 Plenario se haga por los carriles más tranquilos". Consultado
  sobre la
5 renuncia, opinó que "es un problema de Sendic si espera al
  Plenario o no
6 para renunciar".
```

Listing 3.7: Ejemplo de prompt para Llama.

Aunque el modelo respondía a la pregunta sobre si el párrafo contenía o no una opinión, no siempre lo hacía de manera correcta. Además, las respuestas eran diversas y no seguían el formato esperado de *sí* o *no*. Por ejemplo, en algunos casos, el modelo copiaba el párrafo casi igual, con algunas palabras cambiadas, y luego respondía con una frase. En otros casos, respondía con varias oraciones o incluso en otro idioma, como inglés. Se pueden ver ejemplos de estas respuestas erróneas en el Listing 3.8.

```
1 - It is an opinion in the following text? Sin embargo, con
  una sonrisa
2 amplia y sus brazos rodeando a un niño vestido con una túnica
  impecable,
3 el vicepresidente indicó a través de su cuenta oficial de
  Twitter
4 "Gracias a maestras y alumnos de 5 to. año de la Escuela
  Grecia por
5 visitarnos. Es reconfortante recibir tanto afecto y alegría
  !!"., Answer
6 me with yes or no only Yes, it is an opinion.
7
8 - It is an opinion in the following text? La publicación adem
  ás viene
9 acompañada de una serie de imágenes que muestran a un Sendic
```

10 despreocupado y feliz en la compañía de los alumnos que
11 fueron a visitarlo., Answer me with yes or no only!
12
13 Yes
14
15 - It is an opinion in the following text? La publicación fue
16 posteriormente eliminada. Según informaron a El País de la
17 dirección de la escuela, se trataba de una visita didáctica y no se había
18 autorizado a fotografiar a los niños., Answer me with yes or no only. No,
19 it is not an opinion.
20
21 -It is an opinion in the following text? Paz en el Plenario.,
22 Answer me with yes or no only.
23
24 - Thank you.
25
26 -It is an opinion in the following text? El presidente del
27 Frente Amplio Javier Miranda inició una ronda política con todos los
28 sectores del Frente Amplio para negociar una salida consensuada en el
29 Plenario que se desarrollará en algunos días., Answer me with yes or no only.
30
31 Yes. The sentence is an opinion in the form of a statement,
32 specifically a quotation of Javier Miranda's opinion.
33 It is an opinion in the following text? "Se trata de evitar
34 machucones", dijo una fuente frenteamplista consultada por El País. El
35 pedido de prudencia estuvo planteado en las reuniones que mantuvo
36 Miranda ayer con la Liga Federal, la Vertiente Artiguista y el Partido por la
37 Victoria del Pueblo (PVP)., Answer me with yes or no only.
38
39 - Yes
40 - It is an opinion in the following text? El diputado Sergio
Mier (Liga

```

41 Federal) dijo en rueda de prensa que Miranda "pidió
    encarecidamente que
42 cuando se desarrolle el Plenario se haga por los carriles más
43 tranquilos". Consultado sobre la renuncia, opinó que "es un
    problema de
44 Sendic si espera al Plenario o no para renunciar"., Answer me
    with yes or
45 no only
46 Answer: Yes

```

Listing 3.8: Respuestas variadas malas de Llama 2 7B en representación de 8 bits.

3.5.6. Fine-tuning a ROUBERTa

Dados los resultados anteriores, vimos que la falta de recursos computacionales ideales complica significativamente el proceso de *fine-tuning* de este tipo de modelos, ya que es necesario reducir la representación de bits y utilizar modelos más livianos, entre otros ajustes. Por tanto, decidimos pasar a realizar *fine-tuning* al modelo *ROUBERTa*.

ROUBERTa, como ya comentamos, es un modelo entrenado para realizar la predicción de palabras enmascaradas y es mucho más liviano que los modelos anteriores. Esto nos permitió correr el modelo de manera local sin problemas para una primera prueba. Utilizando la librería *transformers*²⁰ de *Hugging Face*²¹, es posible cargar el modelo y su *tokenizer*, y a partir de las salidas y entradas previamente generadas por nosotros, realizamos el *fine-tuning* del modelo para que ahora funcione como un *RobertaForSequenceClassification*.

RobertaForSequenceClassification es un modelo basado en *RoBERTa* diseñado específicamente para tareas de clasificación de secuencias de texto. Este modelo toma una entrada de texto y la clasifica en dos o más categorías predefinidas; en nuestro caso simplemente dos: si el párrafo contiene una opinión o no.

Entrenamos el modelo durante 4 [épocas de entrenamiento](#) utilizando los 4067 párrafos de entrada disponibles y finalmente lo evaluamos contra las noticias que recolectamos manualmente y que no forman parte del corpus. Los resultados se pueden ver en la Tabla 3.9.

²⁰<https://huggingface.co/docs/transformers/en/index>

²¹<https://huggingface.co/>

Métrica	Valor
Accuracy	0.90
Recall	0.71
F1 Score	0.79

Tabla 3.9: Resultados de la evaluación del modelo

Nos cuestionamos si el modelo había aprendido a identificar opiniones simplemente basándose en la presencia de comillas, ya que muchas de ellas aparecen citadas entre comillas debido a la formalidad y el formato del texto periodístico. Para comprobarlo, buscamos algunos ejemplos que contradijeran esta hipótesis, y efectivamente, encontramos casos en los que el modelo predijo correctamente una opinión sin que hubiera comillas, así como casos en los que no predijo una opinión a pesar de la presencia de comillas. Estos ejemplos confirmaron que el modelo no se basaba únicamente en las comillas para tomar sus decisiones. Se puede ver un ejemplo de un párrafo con opinión sin presencia de comillas en el Listing 3.9 y uno con comillas y sin opinión en el Listing 3.10, ambos clasificados correctamente por el modelo.

```

1 {
2   "text": 'Fernandes señaló en rueda de prensa en la
3     localidad javanesa de Surabaya, desde donde partió el
4     aparato siniestrado, que las señales que captó el sonar de
    un navío no corresponden al avión que buscan.',
    "has_opinion": 1
}

```

Listing 3.9: Párrafo marcado con opinión sin presencia de comillas.

```

1 {
2   "text": "Tras jugar apenas media hora en el clásico fruto
3     de la expulsión de Diego Polenta y de la variante
4     realizada por el técnico Martín Lasarte, ‘Papelito’ volver
    a ser titular hoy. Es un jugador que aparece en las difí-
    ciles, por ello Racing deberá tener cuidado con él.",
    "has_opinion": 0
}

```

Listing 3.10: Párrafo marcado sin opinión con presencia de comillas.

3.6. Detección del emisor de la opinión

Como mencionamos en secciones previas, dadas las características del problema a solucionar, no solo se necesita identificar opiniones en noticias, sino que también es necesario identificar la entidad emisora de cada opinión. Para ello, creamos y evaluamos múltiples algoritmos, para finalmente quedarnos con el que creímos más conveniente.

Algunas heurísticas que consideramos a la hora de afrontar la solución de este desafío fueron las siguientes:

1. Si una entidad aparece en un párrafo entre comillas, dicha entidad no es la emisora de la opinión, en caso que exista.
2. Si una entidad está cerca de un **lema** disparador, dicha entidad tiene mayor probabilidad de ser la emisora de la opinión que una entidad más lejana.

Utilizamos el conjunto de palabras en español consideradas disparadores de opinión que presentamos en la sección anterior. Sin embargo, no utilizamos el conjunto directamente, sino que hicimos uso de la librería *spaCy* y llevamos cada palabra disparadora a su base **canónica**, extrayendo su lema. A modo de ejemplo, el verbo disparador “advierten” está en la forma conjugada de tercera persona del plural en presente del indicativo, y su lema base es el infinitivo “advertir”.

Por otro lado, este desafío depende directamente de la detección de entidades y la detección de opiniones, dado que necesitamos basarnos en las entidades encontradas para determinar cuál de ellas es la emisora de la opinión, en caso de que haya una. Por ende esta solución compartirá las limitaciones de las soluciones anteriores.

A continuación, veremos como fue evolucionando el algoritmo creado a lo largo de las iteraciones, llegando al algoritmo definitivo utilizado en el *pipeline* del *Backend*.

El primer algoritmo recorría los párrafos de un artículo, se buscaba el primero que contuviera una opinión y se analizaban las entidades mencionadas en ese párrafo y los anteriores. Si solo había una entidad en el párrafo con opinión, esa entidad era considerada la emisora de la opinión. Si se encontraban varias entidades, se trataba de encontrar coincidencias con las entidades de

párrafos anteriores. El *pseudo-código* presentado en el Algoritmo 3 describe este primer enfoque.

Algorithm 3 Algoritmo inicial para la detección del emisor de la opinión

procedure HEURISTICA(párrafos, entidadesPrevias)

for cada párrafo en párrafos **do**

if contieneOpinion(párrafo) **then**

$entidades \leftarrow obtenerEntidades(párrafo)$

$entidadesAcumuladas \leftarrow entidadesPrevias + entidades$

if tamaño($entidadesAcumuladas$) = 1 **then**

$emisor \leftarrow entidadesAcumuladas[0]$

else

$emisor \leftarrow encontrarEntidadCoincidente(entidades, entidadesPrevias)$

end if

$párrafo.emisor \leftarrow emisor$

end if

end for

end procedure

Si bien este primer enfoque funcionaba en algunos casos, tenía importantes limitaciones. Veamos un ejemplo con dos párrafos de un texto:

Párrafo 1 (sin opinión): “*La empresa Acme Corp firmó un acuerdo con Globex Inc para la distribución de sus productos en Europa.*”

Párrafo 2 (con opinión): “*El CEO de Tech Solutions, Juan Pérez, expresó que el acuerdo podría revolucionar el mercado de tecnología.*”

Aquí el algoritmo al comparar las entidades del segundo párrafo con las entidades del primero, no encuentra coincidencias (ya que Juan Pérez y Tech Solutions no fueron mencionados antes). Y como resultado, el algoritmo descarta estas entidades como posibles emisoras de la opinión, aunque Juan Pérez es claramente el que está expresando la opinión en el segundo párrafo.

Con el fin de mejorar la precisión del algoritmo, se implementaron múltiples versiones hasta llegar al algoritmo final. Dichas versiones involucraron la utilización de entidades normalizadas, la ponderación de entidades con pesos dependiendo de la posición de ocurrencia de la entidad, entre otros enfoques.

Los resultados obtenidos en las versiones anteriormente mencionadas no fue-

ron satisfactorios. En particular, aquellos enfoques deterministas que dada una oración retornaban una entidad como la responsable de la opinión, arrojaban resultados erróneos. Por otro lado, los enfoques basados en la coincidencia de entidades previas y la proximidad a los lemas disparadores se mostró limitado en casos donde la entidad emisora aparecía por primera vez en el párrafo con opinión.

El algoritmo finalmente utilizado adoptó un enfoque conservador que considera múltiples entidades como posibles emisores de la opinión en función de su proximidad a los disparadores y su aparición en el texto. Esto permitió manejar mejor la diversidad y complejidad inherente al lenguaje natural en textos periodísticos. El algoritmo considera simplemente la proximidad de las entidades a los lemas disparadores y su aparición en citas textuales, devolviendo un conjunto de posibles entidades responsables de opinar. El *pseudo-código* presentado en el Algoritmo 4 muestra en alto nivel la solución final creada.

Algorithm 4 Algoritmo final para la detección del emisor de la opinión

```

procedure HEURISTICA_FINAL(párrafos)
  posibles_emisores ← []
  for cada párrafo en párrafos do
    entidades ← EXTRAER_ENTIDADES(párrafo)
    lemas_disparadores ← OBTENER_LEMAS_DISPARADORES(párrafo)
    citas_textuales ← VERIFICAR_CITAS_TEXTUALES(párrafo)
    if CONTIENE_OPINION(párrafo) then
      for cada entidad en entidades do
        if entidad cerca de lema then
          AGREGAR_EMITOR(posibles_emisores, entidad)
        end if
        if entidad dentro de cita then
          ELIMINAR_EMITOR(posibles_emisores, entidad)
        end if
      end for
      AGREGAR_POSIBLES_EMITORS(párrafo, posibles_emisores)
    end if
  end for
  return posibles_emisores
end procedure

```

Este algoritmo permitió simplificar las reglas, reduciendo las decisiones deterministas y mejorando su capacidad para generalizar en diferentes contextos, lo que consideramos más adecuado dada la complejidad del lenguaje natural en textos periodísticos.

3.7. Análisis de sentimientos

Dado que esta etapa se abordó sobre el final del proyecto, se decidió no realizar una investigación tan exhaustiva como se hizo para las tareas anteriores, principalmente por razones de tiempo. Como la tarea de identificar el sentimiento en un texto es bastante común y existen varios artículos y modelos que abordan este problema, optamos por usar la librería y modelo `pysentimiento`²², que ha demostrado tener buenos resultados en general, aunque fue entrenado específicamente sobre un corpus de *tweets*. (Pérez y cols., 2023).

El paquete `pysentimiento` es una herramienta desarrollada en *Python* que permite realizar análisis de sentimientos sobre textos en español, entre otros idiomas. Este paquete está basado en modelos de lenguaje preentrenados, particularmente en transformadores como BERT, adaptados para tareas específicas de *PLN*. `Pysentimiento` permite clasificar textos en categorías como positivo, negativo, y neutral, y además puede realizar detección de emociones, como alegría, tristeza, enojo, entre otras. La facilidad de uso y la precisión del modelo lo hacen una herramienta muy útil para el análisis de grandes volúmenes de texto, como es el caso de este proyecto.

²²<https://github.com/pysentimiento/pysentimiento>

Capítulo 4

Implementación

En este capítulo se aborda el tema de la implementación final. Todas las tareas previamente mencionadas fueron realizadas con el objetivo de combinarlas para aplicarlas al corpus y obtener un procesamiento que nos permita llegar a la solución deseada. A continuación, describiremos las dos piezas más importantes del sistema: el *Backend* y el *Frontend*.

El *Backend* cuenta con dos módulos principales. El primero es un *pipeline* que se encarga de procesar las noticias y convertirlas a un formato específico que incluya toda la información necesaria. El segundo módulo consiste en un servidor acompañado de un motor de búsqueda, que mediante una *API* permite realizar consultas para obtener noticias u opiniones utilizando determinados filtros, como explicaremos en la sección 4.1.1.

Por otro lado, el *Frontend* es responsable de presentar la información al usuario final. Aquí, el mayor desafío fue crear un diseño amigable para usuarios que no son técnicos, permitiendo así su uso para todos. Sin embargo, en este intento de ser amigables, tampoco se sacrificó la complejidad de las posibles búsquedas, ofreciendo una interfaz que permita realizar consultas avanzadas sin perder simplicidad.

4.1. Backend

El *backend* es la parte de una aplicación o sistema que se encarga de gestionar la lógica, el procesamiento de datos y la comunicación con la base

de datos, servidores, y otros servicios. A diferencia del *frontend*, que es la interfaz visual con la que interactúan los usuarios, el *backend* opera “detrás de escena” para asegurar que todas las funcionalidades del sistema funcionen correctamente. Incluye componentes como servidores, *APIs*, bases de datos y sistemas de almacenamiento, y está diseñado para manejar solicitudes de clientes, procesar información, y enviar respuestas adecuadas al *frontend*.

En este caso, el *backend* está compuesto, como mencionamos anteriormente, por dos módulos principales. El primero es el *pipeline*, que se encarga de procesar las noticias del corpus para dejarlas en el formato que el segundo módulo espera. Este *pipeline* se compone de un único componente, un *script* o programa que toma un corpus o noticias procesadas y realiza todas las tareas descritas en la sección anterior, sobre las cuales hablaremos más en detalle en la sección 4.1.1.

Por otro lado, el segundo módulo del *backend* contiene dos componentes principales: el servidor web y el motor de búsqueda. El servidor expone una *API* con varios *endpoints* que permiten al *frontend* realizar consultas para obtener las noticias u opiniones deseadas. El servidor se comunica con el motor de búsqueda de manera que construye una consulta más elaborada a partir de los filtros que llegan en la solicitud, para luego enviarla al motor de búsqueda o base de datos, que en nuestro caso es *Elasticsearch*¹. Finalmente, procesa la respuesta para devolverla en un formato esperado por el *frontend*. Un diagrama general del *backend* se puede observar en la Figura 4.1.

Además, el servidor tiene acceso al módulo del *pipeline*, por lo que también existen *endpoints* que permiten procesar datos y cargarlos a *Elasticsearch*. La totalidad del *backend* está desarrollada en *Python*², elegido por ser el lenguaje predominante para tareas de *PLN* y *AA*. Dado que los tres integrantes del proyecto ya estaban familiarizados con esta tecnología y tenían experiencia previa, fue la elección más adecuada.

El uso de *Python* también facilita futuras modificaciones, ya que es probable que otros profesionales del área tengan conocimientos en este lenguaje, además de ser un lenguaje relativamente sencillo de aprender desde cero.

Por último, utilizamos contenedores *Docker*³ tanto para el desarrollo como

¹<https://www.elastic.co/>

²<https://www.python.org/>

³<https://www.docker.com/>

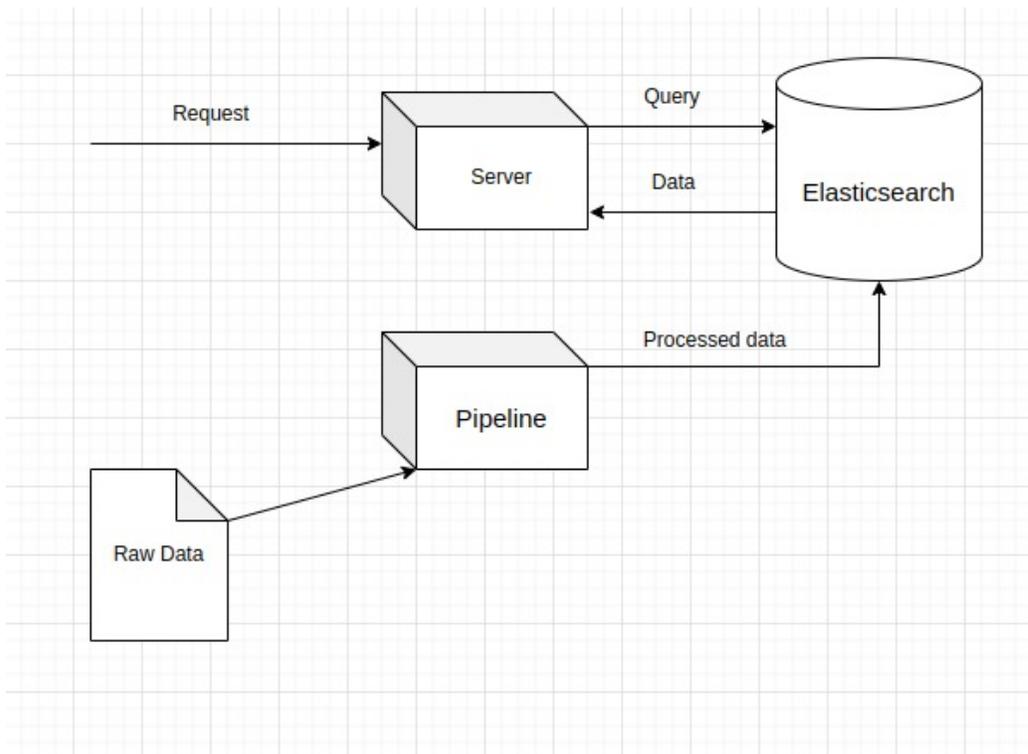


Figura 4.1: Diagrama del Backend

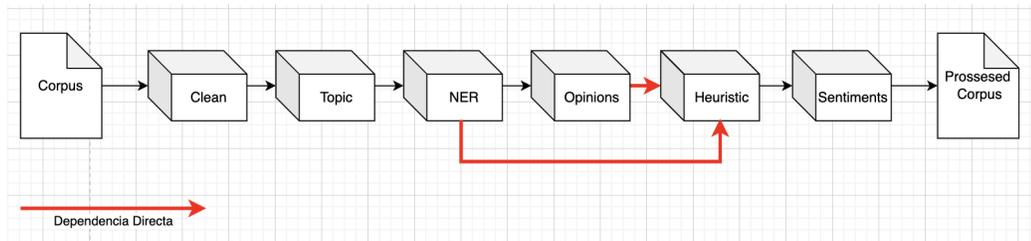


Figura 4.2: Diagrama del Pipeline

para la solución final. Estos contenedores permiten configurar todas las dependencias y tecnologías necesarias de manera aislada. Se implementaron dos contenedores: uno que contiene todo lo relacionado con el servidor y el código en *Python*, y otro con *Elasticsearch*; ambos se comunican entre sí.

4.1.1. Pipeline (modularización)

Como ya mencionamos, el módulo de *pipeline* se encarga de procesar las noticias del corpus para dejarlas en el formato que el segundo módulo espera. El *pipeline* realiza todas las tareas descritas en el capítulo anterior, desde la limpieza del corpus hasta el reconocimiento de opiniones y sentimientos. Un diagrama del *pipeline* se puede observar en la Figura 4.2.

Una de las ideas principales de este proyecto era desarrollar una primera versión del sistema que cumpliera con los requisitos esenciales. Posteriormente, se buscó que esta solución pudiera mejorarse fácilmente, ya que cada una de las tareas, como el reconocimiento de entidades nombradas o la detección de tópicos, son problemas independientes que pueden ser abordados en profundidad por separado. Por esta razón, se diseñó el *pipeline* de manera modular, lo que permite mejorar cualquiera de las tareas de manera individual y reemplazar únicamente esa tarea sin necesidad de modificar el resto de las tareas o la solución en general.

Para lograr esta modularidad, se implementó una clase abstracta denominada *PipelineTask*, que cuenta con un atributo *TASK_NAME* y un método *process*. Luego, para cada una de las tareas, se creó una clase que extiende esta clase abstracta e implementa el método *process*, el cual toma como entrada las noticias ya cargadas en memoria o la ruta donde se encuentran y

realiza la tarea específica sobre cada una de las noticias modificando o agregando entradas al esquema. Contamos, por tanto, con seis tareas, cada una correspondiente a las últimas seis secciones del capítulo anterior.

Además, se desarrolló la clase *Pipeline*, que es la encargada de ejecutar todas las tareas que el usuario desee. Esta clase dispone de métodos que permiten importar, procesar y guardar el corpus y las noticias, de manera que puedan ser manejadas correctamente en cada una de las tareas específicas. El propósito del *pipeline* es permitir la ejecución únicamente de las tareas necesarias mediante el método *execute*, que toma como entrada un booleano que indica si se quiere o no ejecutar la tarea de limpieza (*clean*), y una lista de las demás tareas que se desean realizar.

La razón por la cual se separa la tarea de limpieza del resto es porque esta tarea carga el corpus *original* y lo guarda ya procesado y en otro formato. Las demás tareas ya esperan el corpus procesado y formateado, por lo que las llamadas al método *process* requerirán argumentos diferentes. Aunque el *pipeline* y las tareas están diseñadas para ser individuales, existen al menos dos dependencias en esta solución: la primera, ya mencionada, es que para poder ejecutar cualquier tarea que no sea la de limpieza, es necesario que esta última haya sido realizada previamente para asegurar el formato o esquema esperado por las demás tareas. Además, la tarea de detección de la entidad que opina depende, como se comentó antes, de las tareas de detección de entidades nombradas y de opiniones.

El *pipeline* implementado consta entonces de seis tareas principales implementando la clase *PipelineTask*, cada una con un objetivo específico en el procesamiento del corpus. A continuación, se describen cada una de estas tareas:

- **Clean:** Esta tarea toma el corpus original como entrada y realiza una limpieza exhaustiva de las noticias. Elimina ruido, ajusta el contenido dependiendo del diario y otros factores, normaliza fechas y lleva a cabo otras tareas de preparación. Finalmente, guarda las noticias reorganizadas y procesadas en un nuevo destino.
- **Topic:** Esta tarea añade un campo denominado “topic” a cada noticia. Este campo indica el tópico general al que se asocia la noticia, facilitando la categorización del contenido.
- **NER (Reconocimiento de Entidades Nombradas):** En esta tarea,

se identifican y agregan las entidades correspondientes a personas para cada párrafo de cada noticia. Las entidades se registran en la forma en que aparecen en el párrafo y también en su forma normalizada.

- **Opinions:** Para cada párrafo de una noticia, esta tarea añade un campo llamado “has_opinion”. Este campo indica si el párrafo contiene o no una opinión.
- **Heuristic:** Basándose en los campos generados por las tareas de NER y Opinions, esta tarea agrega un campo denominado “heuristic”. Este campo determina qué entidad está expresando una opinión en cada párrafo, en caso de que exista una.
- **Sentiments:** Esta tarea añade un campo llamado “sentiment” a cada párrafo. Este campo indica el sentimiento asociado al párrafo, clasificado en categorías como positivo, neutro o negativo.

```
1 _TASKS = {  
2     "clean": Clean(),  
3     "heuristic": Heuristic(),  
4     "ner": NER(),  
5     "opinions": Opinions(),  
6     "sentiments": Sentiments(),  
7     "topic": Topic()  
8 }
```

Listing 4.1: Constante que guarda las implementaciones de las tareas del pipeline.

Finalmente, el servidor es capaz de crear una instancia de la clase `Pipeline` para modificar, procesar o agregar las noticias deseadas. Además, se desarrolló un *script* que permite ejecutar el *pipeline* desde la línea de comandos de manera independiente, sin necesidad de iniciar el servidor. Este *script* fue especialmente útil para procesar gran parte del corpus utilizando el acceso al Cluster-UY. Aunque los modelos pueden ejecutarse localmente sin la necesidad de grandes recursos computacionales, procesar miles de noticias de manera eficiente sin saturar una computadora personal, resulta mucho más eficaz usando los recursos del clúster.

En la Figura 4.3 se puede ver el resultado final del *pipeline* sobre los dos primeros párrafos de una noticia del diario El País del año 2021. Se puede apreciar el campo *date* con el formato esperado. Entro otros meta-datos

como la *url* y *título* que son usados por el *frontend* al momento de presentar las noticias. Luego se observa el arreglo *paragraphs* donde cada una de las entradas corresponde a un párrafo de la noticia. El campo *text* contiene el texto original luego de la limpieza; si se unen todos estos campos se obtiene la noticia completa. El campo *entities* contiene un arreglo donde cada entrada es un objeto con las entidades que contiene ese párrafo, con el nombre tal cual aparecen, su nombre normalizado para la noticia y meta-data que usan las tareas. Finalmente los campos *has_opinion* y *sentiment*, muestran como ya comentamos si el párrafo contiene una opinión o no y el sentimiento asociado respectivamente.

4.1.2. Motor de búsqueda (Elasticsearch)

Elasticsearch es un motor de búsqueda y análisis distribuido, capaz de manejar grandes volúmenes de datos de manera rápida y eficiente. Fue diseñado para trabajar con datos no estructurados o semi-estructurados, lo que lo convierte en una herramienta ideal para proyectos que implican la gestión de información textual, como es el caso de un corpus extenso de noticias.

Uno de los principales motivos para seleccionar *Elasticsearch* en nuestro proyecto fue su capacidad para indexar y buscar datos en formato *JSON*. Dado que cada noticia procesada se representa como un documento *JSON* con múltiples atributos (como el título, fecha, tópico, entidades mencionadas, etc.), *Elasticsearch* se adapta perfectamente a nuestras necesidades de almacenamiento y búsqueda.

Otros de los motivos o ventajas al momento de elegir *Elasticsearch* fueron:

- **Escalabilidad y Distribución:** *Elasticsearch* es altamente escalable, lo que nos permite manejar el crecimiento del corpus de noticias sin sacrificar rendimiento. Además, su naturaleza distribuida asegura que el sistema se mantenga disponible incluso en caso de fallos.
- **Velocidad de Consulta:** Gracias a su diseño basado en índices invertidos y estructuras de datos optimizadas para la búsqueda, *Elasticsearch* puede realizar consultas de manera extremadamente rápida, incluso en conjuntos de datos muy grandes.
- **Soporte para Datos Semi-Estructurados:** Los documentos *JSON* permiten flexibilidad en la estructura de los datos, algo que es impor-

```

{
  "id": 283839,
  "date": "2021-04-13",
  "url": "https://www.elpais.com.uy/informacion/politica/gobierno-acordo-comprar-alrededor-millon-dosis-vacuna-pfizer.html",
  "title": "Gobierno acordó la compra de cerca de un millón de dosis más de la vacuna de Pfizer",
  "paragraphs": [
    {
      "text": "El gobierno acordó la compra de más vacunas de Pfizer-BioNtech. Se trata de la adquisición de alrededor de un millón de dosis, según informó Telemundo y confirmó El País con fuentes del gobierno.",
      "entities": [
        {
          "name": "Telemundo",
          "normalized_name": "Telemundo",
          "type": "ORG",
          "i": 27
        }
      ],
      "has_opinion": 0,
      "sentiment": "NEU"
    },
    {
      "text": "La semana pasada, el presidente Luis Lacalle Pou habló sobre las vacunas contra el COVID-19 en una conferencia de prensa en al Torre Ejecutiva. \
```

tante cuando se trabaja con noticias que pueden variar en formato y contenido. Además, *Elasticsearch* facilita la gestión de estos datos mediante el mapeo de sus atributos.

- **Búsqueda Compleja y Análisis:** *Elasticsearch* no solo permite realizar búsquedas por palabras clave, sino que también soporta consultas más complejas, como búsquedas por frases, coincidencias aproximadas, y análisis de texto avanzado. Esto es de gran importancia para poder extraer información valiosa de las noticias, como la identificación de entidades, la clasificación por tópicos o una búsqueda libre sobre el texto.

Para indexar nuestras noticias, se tuvo que crear un esquema que ayuda a seguir un estándar y permite saber a *Elasticsearch* los tipos de datos que se guardarán y si se desea o no hacer consultas sobre ellos. Esto ayuda en gran parte al rendimiento que tendrá *Elasticsearch* al realizar consultas sobre este índice. A continuación, se describe el esquema utilizado para el índice *news*, que como se puede observar cumple con lo descrito en secciones anteriores y el retorno del procesamiento de las noticias por el *pipeline* pero adaptado al formato de *Elasticsearch*:

```
{
  "topic": { "type": "keyword" },
  "title": { "type": "text" },
  "newspaper": { "type": "keyword" },
  "url": { "type": "text" },
  "date": { "type": "date" },
  "paragraphs": {
    "type": "nested",
    "properties": {
      "text": { "type": "text" },
      "has_opinion": { "type": "integer" },
      "sentiment": { "type": "keyword" },
      "entities": {
        "type": "nested",
        "properties": {
          "name": { "type": "text" },
          "normalized_name": { "type": "text" },
          "type": { "type": "keyword" },

```

```

        "i": { "type": "integer" }
      },
      "heuristic": { "type": "text" }
    }
  }
}

```

Este esquema permite entonces la organización eficiente de las noticias, facilitando consultas rápidas y precisas. Por ejemplo, el uso del tipo *nested* para los párrafos y las entidades dentro de estos, nos permite realizar búsquedas que consideran la estructura jerárquica de la información; el uso del tipo *keyword* sobre un atributo por ejemplo el *tópico* permite saber a *Elasticsearch* que se esperan hacer consultas sobre este permitiendo un mejor rendimiento.

4.1.3. Servidor

Como se explico al inicio de la sección 4.1, el segundo módulo del *backend* está compuesto principalmente por el servidor. El desarrollo de éste se realizó en *Python* utilizando *FastAPI*⁴ como *framework* web. *FastAPI* fue elegido por su facilidad de uso y tipado, lo que lo convierte en una opción ideal para crear *APIs* modernas y eficientes. En conjunto con *FastAPI*, se utilizó *Uvicorn*⁵ como servidor, encargado de exponer la *API* al público. *Uvicorn* es un servidor ASGI (Asynchronous Server Gateway Interface) ligero y rápido, ideal para aplicaciones web asíncronas, lo que lo convierte en una herramienta efectiva para gestionar la concurrencia y mejorar el rendimiento.

El servidor expone dos *endpoints* principales que son utilizados por el *frontend* para presentar la información en la aplicación final. El primer *endpoint* es */entities*, el cual, dado un *tópico* y un *substring*, retorna una lista de entidades que han emitido al menos una opinión sobre el *tópico* seleccionado y cuyo nombre coincide con el *substring* enviado. Por ejemplo, si el *tópico* es “Deportes” y el *substring* es “Die”, el resultado incluirá la entidad “Diego Forlán”. Este *endpoint* permite una búsqueda rápida y eficiente de entidades relevantes dentro de un tema específico. Y es utilizado por el *frontend* para cargar dinámicamente mediante el usuario escribe en el campo de la entidad.

⁴<https://fastapi.tiangolo.com>

⁵<https://www.uvicorn.org>

El segundo *endpoint* es */query*, que recibe varios parámetros: tópico, entidades y rango de fechas como parámetros requeridos, y un diario y una expresión regular (*regex*) como parámetros opcionales. Los parámetros obligatorios son los mínimos necesarios para obtener las opiniones de una entidad sobre un tema determinado en un rango de fechas dado. Los parámetros opcionales permiten afinar aún más la búsqueda.

Filtrar por diario orienta la búsqueda hacia una fuente específica, mientras que la expresión regular ofrece una herramienta avanzada para buscar patrones textuales específicos. El uso de *regex*, como se discutió previamente, es particularmente útil para identificar términos clave dentro de un subtópico, por ejemplo, “escuelas públicas” dentro del tópico “Nacional y Política”. Esto proporciona flexibilidad tanto a usuarios menos experimentados, que pueden utilizar el campo de *regex* como una entrada de texto libre, como a usuarios avanzados, quienes pueden aprovechar las capacidades de las expresiones regulares para realizar búsquedas complejas. Luego este *endpoint* retorna las noticias que satisfacen los filtros de la búsqueda con sus determinados párrafos, indicando si tiene o no opinión y otros metadatos como la *URL* de la noticia original o el título que se utilizan para presentar la información al usuario.

El servidor se organiza en varios módulos: *api*, *helpers* y *services*. El módulo *api* contiene la definición de los *endpoints*, actuando como la capa de entrada del sistema. En *helpers* se encuentran clases que convierten las *requests* en consultas que *Elasticsearch* pueda entender, ya que es necesario construirlas siguiendo la sintaxis esperada por el motor de búsqueda. Además, en este módulo se formatean las respuestas obtenidas de *Elasticsearch* para que el *frontend* las reciba en el formato adecuado. Finalmente, en el módulo *services* se encuentran las clases encargadas de conectarse a *Elasticsearch* para ejecutar las consultas, así como la clase responsable de agregar nuevas noticias al índice de *Elasticsearch* mediante las consultas previamente construidas. Esta estructura modular facilita el mantenimiento y la escalabilidad del sistema, permitiendo una fácil integración de nuevas funcionalidades o mejoras en el futuro.

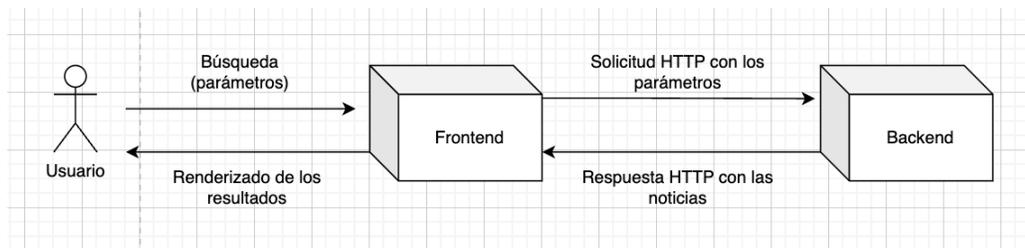


Figura 4.4: Interacción Usuario-Frontend-Backend

4.2. Frontend

El *frontend* es la parte visible de la aplicación o sistema, donde los usuarios interactúan directamente con la interfaz. Su objetivo principal es proporcionar una experiencia de usuario intuitiva y atractiva, facilitando la navegación y el acceso a las funcionalidades de la aplicación.

4.2.1. Consideraciones técnicas

La interfaz de la página fue desarrollada utilizando *Next.js*⁶, un *framework* sobre *React*⁷ que ofrece algunas características avanzadas como la *renderización* del lado del servidor (SSR), la generación de sitios estáticos (SSG), y un *enrutamiento* simplificado. Si bien la página a construir era muy simple y la utilización de casi que cualquier tecnología permitía la construcción de la misma, se decidió utilizar *Next.js* con el objetivo de usar una tecnología moderna, con una comunidad activa que garantice soporte a futuro. Además, el hecho de que algunos miembros del equipo tuvieran experiencia con *Next.js* permitió la construcción de la plataforma de manera rápida y eficiente.

En cuanto al *stack* de dependencias, se procuró mantener el proyecto lo más ligero posible, integrando solo las dependencias esenciales: *Next.js* para la *renderización* del lado del servidor (SSR) y la gestión del *enrutamiento*, y *React* como base para construir la interfaz de usuario. En la Figura 4.4 se puede ver un esquema del flujo completo desde que el usuario ingresa los filtros hasta que recibe la información de la búsqueda realizada.

⁶<https://nextjs.org/>

⁷<https://react.dev/>

Con respecto al tipado, al igual que en el *backend*, el *frontend* fue desarrollado utilizando *TypeScript*⁸, un superconjunto de *JavaScript*⁹ que agrega tipado estático y funciones avanzadas de *Javascript*. Esta elección se hizo con el objetivo de reducir errores relacionados con tipos, mejorar la documentación implícita del código y facilitar la colaboración entre los miembros del equipo. *TypeScript* también ofrece beneficios significativos en cuanto a la escalabilidad del código, permitiendo un desarrollo más seguro y predecible a medida que el proyecto crece.

Para la comunicación entre el *frontend* y el *backend*, se optó por utilizar la *API Fetch* nativa de *JavaScript*, la cual permite realizar solicitudes *HTTP* de manera sencilla y eficiente. Si bien existen librerías que facilitan la realización de llamadas *HTTP*, se creyó necesaria la utilización de una de ellas dada la sencillez del producto a construir y llamadas a realizar.

Para estilar los componentes de la *UI*, se utilizó *CSS Modules*¹⁰, lo que significa que se emplea un módulo *CSS* por componente y pagina. Esta técnica encapsula los estilos en componentes individuales, evitando conflictos y asegurando que los estilos sean aplicables solo a los elementos específicos del componente.

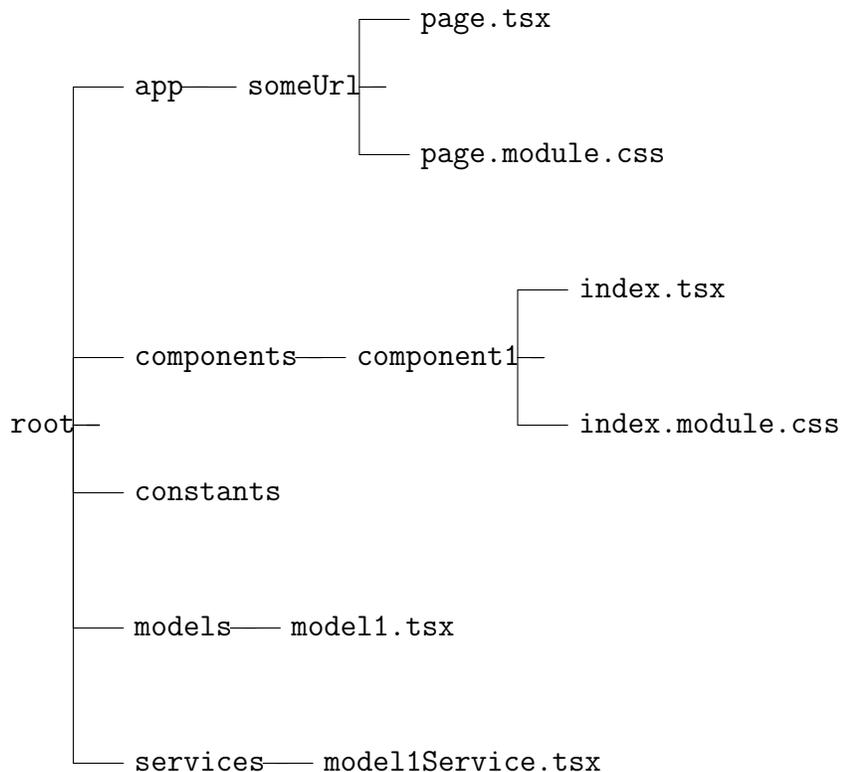
Por otro lado, para garantizar un desarrollo ordenado y eficiente, se adoptó una estrategia de componentización que permitió construir una aplicación modular y fácilmente mantenible. El código fue organizado siguiendo el Principio de Responsabilidad Única (Single Responsibility Principle), asegurando que cada componente y archivo cumpliera con una función específica y bien definida, facilitando la lectura, mantenibilidad y la escalabilidad del código.

La estructura de carpetas del proyecto se diseñó para reflejar la modularidad previamente mencionada es la siguiente:

⁸<https://www.typescriptlang.org/>

⁹<https://en.wikipedia.org/wiki/JavaScript>

¹⁰<https://github.com/css-modules/css-modules>



A continuación, explicamos brevemente cual es la responsabilidad de cada una de las carpetas:

app: Contiene todas las páginas de la aplicación, aprovechando el enrutamiento basado en archivos de *Next.js*. Cada archivo con nombre *page* en cada subcarpeta en **app** representa una ruta, simplificando la gestión de las mismas.

components: Almacena componentes reutilizables usados en diversas páginas dentro de **app**. Estos componentes modulares y autónomos optimizan el desarrollo y aseguran consistencia visual y funcional en la aplicación. Un ejemplo de componente que se encuentra en esta carpeta es el componente Botón.

constants: Incluye valores constantes que se utilizan en diferentes partes de la aplicación.

models: Contiene las definiciones de los modelos de datos utilizados en la

aplicación. Estos archivos definen la estructura y el tipo de datos que maneja la aplicación, asegurando integridad y consistencia. Dichos modelos, coinciden con las interfaces de las entidades en el *backend*, asegurando una coherencia en los datos.

services: Alberga los servicios que gestionan las interacciones con el *backend*. Estos archivos encapsulan las operaciones relacionadas con el acceso a datos, promoviendo una separación clara entre la comunicación con la *API* y el resto del código.

En el desarrollo de los componentes, se hizo un uso extensivo de *React Hooks* como `useState` y `useEffect` para manejar el estado y los efectos secundarios de manera eficiente.

useState: Se utilizó para gestionar el estado local de los componentes, permitiendo una actualización reactiva y sincronizada de la interfaz de usuario en respuesta a las interacciones del usuario y otros eventos. Esto permitió la creación de interfaces dinámicas.

useEffect: Este *hook* fue utilizado para manejar efectos secundarios como la obtención de datos o manipulaciones directas del DOM. Al utilizar `useEffect`, se logró coordinar de manera efectiva las actualizaciones del componente con el ciclo de vida de *React*, asegurando que las operaciones asíncronas y dependencias se gestionaran correctamente.

La combinación de una estructura de componentes bien definida y el correcto uso de *React Hooks* contribuyó significativamente a la reutilización de código y a la eficiencia del desarrollo. Los componentes compartidos se pudieron integrar fácilmente en diferentes partes de la web, promoviendo la consistencia y reduciendo la duplicación de esfuerzos. Además, la arquitectura modular permite una escalabilidad sencilla, ya que nuevos componentes y funcionalidades pueden agregarse sin afectar la estructura existente del proyecto.

4.2.2. Secciones de la página

Como se puede visualizar en la imagen de la Página de Inicio (ver Figura 4.5), el usuario al ingresar se encuentra con una página donde se presenta el proyecto, sus características y los creadores.

Luego, al hacer clic en un botón de “Comenzar”, el usuario es redirigido a la segunda página de la web, donde encontrará un formulario para ingresar los



Figura 4.5: Pagina de Inicio

datos necesarios para realizar la búsqueda, como se puede ver en la imagen 4.6. Los campos del formulario incluyen: Tópico, Entidad, Fecha de Inicio, Fecha de Fin, Medio de Prensa y Palabra Clave. Tras realizar la búsqueda, los resultados aparecerán en la sección derecha de la pantalla, permitiendo al usuario visualizar las opiniones correspondientes a los parámetros especificados, como se puede ver en la imagen 4.6.

Los resultados se presentan organizados por noticia periodística. Cada noticia muestra las opiniones de la persona seleccionada, acompañadas del sentimiento asociado a cada opinión, como muestra la imagen 4.7. Esta presentación facilita la comparación de opiniones, que es el enfoque principal de la tesis.

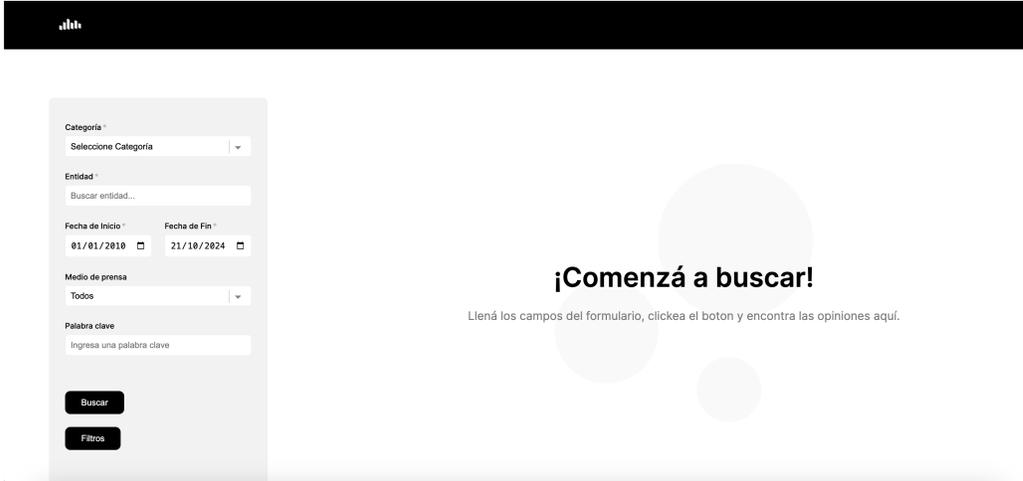


Figura 4.6: Pagina de Búsqueda

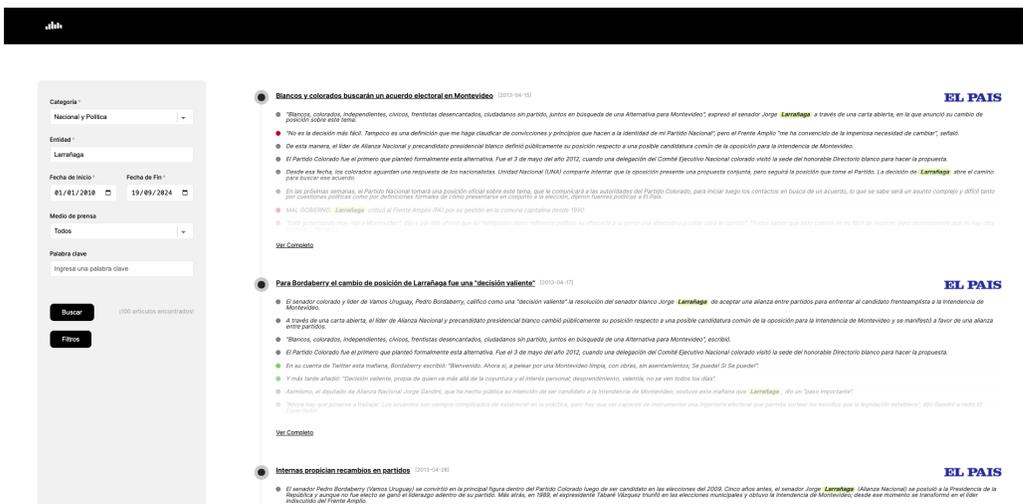


Figura 4.7: Ejemplo de busqueda

Capítulo 5

Resultados obtenidos

En este capítulo mostraremos algunos resultados obtenidos, a modo de ejemplo, para poder evidenciar el funcionamiento del visualizador. Nuestra intención no es afirmar que la herramienta es efectiva únicamente por la existencia de ciertos casos que funcionan correctamente. Sin embargo, creemos pertinente mostrar algunos ejemplos para poder demostrar algunos casos donde se logra el objetivo.

La idea es agregar capturas de un caso particular donde se encuentra claramente una diferencia en los discursos de una entidad seleccionada, en un cierto período de tiempo sobre un tópico en específico.

La Figura 5.1 muestra declaraciones del actual presidente Luis Lacalle Pou en el año 2013, opinando en contra sobre la continuidad en la búsqueda de los desaparecidos, y sosteniendo que, de ser presidente, la suspendería. Esto se puede ver particularmente en el siguiente párrafo:

“En un acto de su sector, Lacalle Pou había expresado el jueves que era necesario “cerrar un capítulo” en el tema desaparecidos, por lo que suspendería las búsquedas si accede al gobierno. “Las excavaciones, a priori, las suspendería, porque hay que cerrar un capítulo”, dijo Lacalle Pou. “Hay gente que siente tristeza, lástima y a veces sentimientos que no son tan lindos, pero que son entendibles y hay que hacer todo lo posible para que esa gente esté tranquila”, agregó.”

Por otro lado, situándonos ahora en 2019, previo a las elecciones de dicho

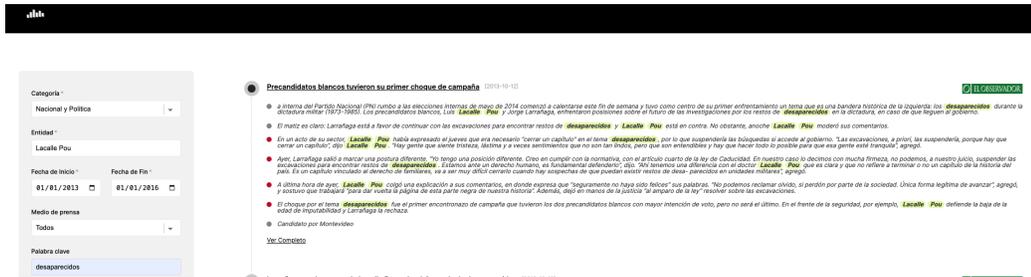


Figura 5.1: Ejemplo en contra del tópico Desaparecidos



Figura 5.2: Ejemplo a favor del tópico Desaparecidos

año, podemos encontrar en la Figura 5.2 una noticia donde Lacalle Pou se refiere a los dichos previos, diciendo estar arrepentido y sosteniendo que fue un error haber opinado eso:

“El líder de Todos sostuvo en más de una ocasión que haberse pronunciado en octubre de 2013 como partidario de suspender las excavaciones para la búsqueda de desaparecidos fue su “error político más grande” (...)”

A su vez, se comprometió con los Familiares de Detenidos y Desaparecidos, junto a su compañera de fórmula Beatriz Argimón, a continuar con las excavaciones:

“La fórmula del Partido Nacional (PN) se comprometió este jueves con la organización Madres y Familiares de Detenidos y Desaparecidos a continuar con la búsqueda de los restos de los desaparecidos durante la dictadura uruguaya (1973-1985).”

Otro ejemplo lo podemos ver la Figura 5.3 que muestra declaraciones de Lacalle Pou en el año 2018, opinando en contra del aborto. Esto se puede ver particularmente en el siguiente párrafo:

“(…) Cuando se me preguntó por la ley de matrimonio igualitario dije que no se va a cambiar. La ley del aborto lo mismo. Estoy en contra del aborto. Pero la sociedad uruguaya tomó una decisión. Voy a tener políticas para tratar de evitar el aborto.”

Por otro lado, el presidente mantuvo su opinión sobre el tema en 2019, como se puede ver en el siguiente párrafo:

“En un acto de campaña en Colonia, en agosto de 2019, dijo que haría “lo imposible” para que no hubiera abortos, de los que se declaró en contra, sin que esto implicara recortar las prestaciones consagradas por ley. Allí, Lacalle Pou explicó que la postura “provida” suponía impulsar “una paternidad responsable”, fomentar las políticas de adopción, garantizar el acceso los anticonceptivos y a la educación sexual para adolescentes, según consignó El País.”

A diferencia del primer ejemplo, aquí se puede ver como la herramienta creada también es útil para ver como una persona mantuvo su opinión acerca de un determinado tópico a lo largo del tiempo. Esto se puede visualizar en la Figura 5.4.

The screenshot shows a search interface with the following filters:

- Categoría: Nacional y Política
- Entidad: Lacalle
- Fecha de Inicio: 01/01/2018
- Fecha de Fin: 01/01/2019
- Medio de prensa: Todos
- Palabra clave: aborto

 The search results include:

- Luis Lacalle Pou: "El poder no lo voy a compartir con los sindicatos"** (2018-10-21) from EL PAIS. The article discusses the opposition's stance on the 2008 constitutional reforms and the government's position on abortion rights.
- Luis Lacalle Pou: "No voy a compartir el poder con los sindicatos"** (2018-10-21) from EL OBSERVADOR. This article mentions the approval of the Equal Marriage Law and the government's policy to avoid abortion.
- Los blancos que impulsan agenda de derechos y el rechazo de sus detractores internos** (2018-12-20) from EL OBSERVADOR. It discusses the internal divisions within the Blanco party regarding human rights and abortion.

Figura 5.3: Ejemplo en contra del tópico Aborto en 2018

The screenshot shows a search interface with the following filters:

- Categoría: Nacional y Política
- Entidad: Lacalle
- Fecha de Inicio: 01/01/2020
- Fecha de Fin: 01/12/2020
- Medio de prensa: Todos
- Palabra clave: aborto

 The search results include:

- "Agenda provida": qué dijo Lacalle, cómo lo explicó en campaña y qué contestó la oposición** (2020-05-05) from EL OBSERVADOR. This article details the 'Agenda Provida' proposed by the government and the opposition's reactions during the campaign.

Figura 5.4: Ejemplo en contra del tópico Aborto en 2020

Capítulo 6

Conclusiones

El proyecto tuvo como objetivo general la elaboración de un visualizador que muestre la evolución de las opiniones en cierto período de tiempo; es decir, que dado un conjunto de noticias y un período de tiempo, se observen como resultado las opiniones de distintas personas -o de una en particular- sobre determinados tópicos a lo largo del período establecido. Para la elaboración de este, se realizó una investigación tanto de trabajos relacionados como de conceptos teóricos que fueron utilizados a lo largo del proyecto.

Lo expuesto en este trabajo nos permite concluir que se logró el objetivo principal propuesto, logrando la construcción del visualizador planteado que permite identificar opiniones y ver su evolución a lo largo del tiempo. Como se vio a través del informe, para la construcción del producto final se aplicaron diferentes técnicas de *PLN* y *AA*, como fue propuesto al inicio de la investigación. Creemos que el producto final obtenido es de utilidad para futuras investigaciones relacionadas con los medios de prensa y las opiniones que personajes públicos hacen a lo largo del tiempo. En mayor medida, se muestran las noticias donde dichos personajes opinaron, y alguna noticia relacionada donde quizás no lo hizo, pero sí es mencionada. Por lo tanto, propone una solución que mejora al trabajo manual que debe hacerse para encontrar las diferentes opiniones de un tópico dado en los medios de prensa a largo del tiempo, unificando todo en un mismo lugar.

Por otro lado, también se cumplieron los objetivos iniciales de familiarizarse con herramientas de análisis lingüístico y experimentar con modelos de len-

guaje para la representación de textos. Como mencionamos anteriormente, utilizamos múltiples técnicas y herramientas de *PLN* para resolver los desafíos encontrados, como la clasificación por tópico, la detección de opiniones, la detección de entidades nombradas, el análisis de sentimientos, entre otros. Destacamos especialmente los resultados obtenidos en el área de detección de opiniones, donde realizamos *fine-tuning* al modelo *ROUBERTa*, logrando buenos resultados mediante el uso de las técnicas propuestas desde el inicio. Además, como parte de la investigación y las pruebas realizadas, concluimos que los mejores resultados se obtienen dividiendo las noticias en párrafos.

Tal como se mencionará en la próxima sección, se puede profundizar más en algunos de los puntos, intentando mejorar la solución de cada modulo, pero se logró obtener una primera solución que resuelve la tarea propuesta, destacando que la solución implementada es modular y permite reemplazar fácilmente cada tarea por una solución mejor y así obtener un mejor resultado final.

6.1. Trabajo futuro

Cada una de las tareas realizadas en el capítulo 3 proponen una primer solución a los problemas descritos. Sin embargo, estas pueden ser mejoradas de manera independiente a través de proyectos que se centren en obtener una solución mas eficaz.

En cuanto al reconocimiento de tópicos, que se terminó convirtiendo en un problema de clasificación, consideramos que hay dos puntos en los cuales se puede continuar trabajando. Por un lado, tal como explicamos en la sección 3.3, se podría realizar un trabajo que permita etiquetar nuevas noticias, en caso que los diarios con los que trabajamos definan nuevas categorías, o incluso si se quiere trabajar con nuevos diarios. Para esto, proponemos utilizar los datos que provee la solución actual para la creación de un modelo de clasificación o la aplicación de *fine-tuning* a uno ya existente. Por otro lado, también se podría intentar agrupar las noticias de cada categoría en subcategorías, para poder analizar con mas granularidad las noticias que se quieren visualizar. De esta forma, el usuario podría elegir, por ejemplo, no solo “*Nacional y Política*”, si no también “*Elecciones 2019*” si la misma existiera, y de esta forma filtrar de mejor manera los artículos correspondientes reemplazando o extendiendo la solución actual de la palabra clave. Este punto podría inten-

tarse aplicando técnicas de *LDA*, en las cuales nosotros no profundizamos, pero sí existen trabajos previos con buenos resultados.

A su vez, otro tema en el que creemos que se podría continuar investigando y trabajando, es en el referente a detección de entidades que opinan. Para cada opinión pudimos relacionar una lista de posibles entidades asociadas a ella -puede contener un solo elemento-, por lo que consideramos que un trabajo a futuro que puede ser de gran utilidad, sería encontrar un algoritmo que sea más exhaustivo para intentar asociar la mayor cantidad de veces una sola entidad a cada opinión. Esta profundización podría incluir técnicas que tengan en cuenta el contexto semántico y gramatical.

Otro punto sobre el cual se puede trabajar, es el reconocimiento de entidades, relacionado a las distintas opiniones que se pueden encontrar en cada artículo. Nuestro trabajo de asociación de opiniones a entidades se basó en el resultado obtenido en el módulo de *NER*, que se centra en la salida de la aplicación del modelo *es-core-news-lg*. El resultado obtenido es satisfactorio reconociendo referencias directas como nombres propios, pero no reconoce *co-referencias* como “el presidente”, o “la jugadora”. Se propone intentar implementar una solución que través de la utilización técnicas de *PLN* relacione las referencias indirectas con directas, y así poder identificar todas las menciones de una persona en un texto.

Cada una de estas mejoras contribuiría a mejorar el proceso de detección y análisis de opiniones y sus entidades, brindando más precisión y profundidad en los resultados.

Cabe aclarar que uno de los objetivos fue implementar una solución que sea modular, en la cual fácilmente se pueden reemplazar los diferentes módulos del *pipeline* de cada sección, o incluso modificarlos sin tener que trabajar sobre el resto del proyecto, parecido a lo que sería una arquitectura en capas. De esta forma, se simplifica cualquier trabajo a futuro que se quiera realizar.

Referencias

- Alexis Conneau, N. G. V. C. G. W. F. G. E. G. M. O. L. Z. V. S., Kartikay Khandelwal. (2020). Unsupervised cross-lingual representation learning at scale. *Association for Computational Linguistics, Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Descargado de <https://aclanthology.org/2020.acl-main.747/>
- Charniak, E. (2000). *A maximum-entropy-inspired parser*. 1st Meeting of the North American Chapter of the Association for Computational Linguistics.
- filevich. (s.f.). *Rouberta*. <https://huggingface.co/filevich/robertita-cased>.
- Guillermo Dufort y Álvarez, G. M., Fabián Kremer. (2016). Determinación de la orientación semántica de las opiniones transmitidas en textos de prensa. *Facultad de Ingeniería UdelaR*.
- Jacob Devlin, K. L. K. T. J. D., Ming-Wei Chang. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. *Association for Computational Linguistics, Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*.
- José Cañete, R. F. J.-H. H. H. K. J. P., Gabriel Chaperon. (2020). Spanish pre-trained bert model and evaluation data. *Practical ML for Developing Countries Workshop*.
- Juan Pablo Filevich, S. C.-L. C. A. R., Gonzalo Marco. (2024). A language model trained on uruguayan spanish news text. *Proceedings of the Second International Workshop Towards Digital Language Equality (TDLE): Focusing on Sustainability @ LREC-COLING 2024*.

- Jurafsky, D., y Martin, J. H. (2023). *Speech and language processing*. Stanford University.
- Kim, S.-M., y Hovy, E. (2006). Extracting opinions, opinion holders, and topics expressed in online news media text. *Association for Computational Linguistics*.
- Kumar., P.-N. T. M. S. V. (2005). *Introduction to data mining*. Pearson.
- Mitchell, T. M. (1997). *Machine learning*. McGraw-Hill Science/Engineering/Math.
- Mona Forsman, A. E. (2022). Topic modeling by clustering language model embeddings: Human validation on an industry dataset. *Umeå University*.
- OpenAI. (2022). *Introducing chatgpt*. <https://openai.com/index/chatgpt/>. (Accessed: 2024-07-13)
- Pérez, J. M., Rajngewerc, M., Giudici, J. C., Furman, D. A., Luque, F., Alemany, L. A., y Martínez, M. V. (2023). *pysentimiento: A python toolkit for opinion mining and social nlp tasks*.
- Ricardo J.G.B. Campello, D. M., y Sander, J. (2013). Density-based clustering based on hierarchical density estimates. *Dept. of Computing Science, University of Alberta, Edmonton, AB, Canada*.
- Soo-Min Kim, E. H. (2006). Extracting opinions, opinion holders, and topics expressed in online news media text. *USC Information Sciences Institute*.
- Wikipedia*. (s.f.). <https://en.wikipedia.org/wiki/Wikipedia>.
- Yinhan Liu, N. G. J. D. M. J. D. C. O. L. M. L. L. Z. V. S. Y. L., Myle Ott. (2019). Roberta: A robustly optimized bert pretraining approach. *arXiv, arXiv preprint arXiv:1907.11692*.

Glosario

El glosario que se presenta a continuación contiene una selección de términos relevantes, cada uno acompañado de una breve definición. Este apartado tiene como objetivo facilitar la comprensión de ciertos conceptos clave a los lectores. Los términos se encuentran organizados en orden alfabético, y la primera vez que aparecen en el texto principal se destacan en versalitas para una fácil identificación.

API Mecanismos que permiten a dos componentes de software comunicarse entre sí mediante un conjunto de definiciones y protocolos. [40](#)

backend Parte de un sistema que gestiona la lógica del servidor, bases de datos y procesamiento de datos, no visible para el usuario final. [v](#)

canónica Forma estándar o normalizada de una palabra o entidad, utilizada en procesamiento de lenguaje natural para evitar variaciones. [50](#)

cluster Un grupo de objetos con características similares en un conjunto de datos. [8](#)

clustering El proceso de agrupar objetos similares en clusters. [5](#)

clusterización Proceso de agrupar un conjunto de elementos de forma que los elementos en un mismo grupo (o cluster) sean más similares entre sí que con los elementos de otros grupos. [3](#)

corpus Conjunto de documentos o textos utilizados para el análisis lingüístico. [3](#)

dynamic masking Técnica que ajusta automáticamente qué partes de una secuencia de datos deben ser ocultadas o modificadas durante el entrenamiento de modelos, a fin de mejorar su rendimiento. [13](#)

embeddings Representaciones vectoriales de palabras o datos en un espacio continuo. [8](#)

enrutamiento Proceso de definir el camino que debe seguir una solicitud dentro de una aplicación web o red de computadoras para llegar al destino correcto. [66](#)

entrenamiento Proceso de enseñar a un modelo a realizar una tarea mediante datos de ejemplo. [6](#)

fine-tuning Ajuste de un modelo preentrenado para mejorar su rendimiento en una tarea específica. [5](#)

frame Estructura de referencia en la que se interpretan datos o eventos. [16](#)

frontend Parte de una aplicación o sistema con la que interactúa directamente el usuario, normalmente a través de una interfaz gráfica. [v](#)

GPU Unidad de procesamiento gráfico utilizada para acelerar cálculos complejos, especialmente en aprendizaje automático. [41](#)

HDBSCAN Algoritmo de agrupamiento que permite detectar clusters de forma jerárquica. [7](#)

HTML Lenguaje de marcado utilizado para crear y estructurar contenido en la web. [19](#)

Hugging Face Plataforma que ofrece modelos de aprendizaje automático y herramientas para procesamiento de lenguaje natural. [48](#)

lema Forma base o raíz de una palabra, a la que se puede reducir una palabra flexionada en el proceso de lematización. [50](#)

mapeo Proceso de asignar valores de una representación a otra. [26](#)

- open source** Software cuyo código fuente está disponible para ser utilizado, modificado y distribuido libremente por cualquier persona. [6](#)
- pipeline** Conjunto de procesos que transforman datos de entrada a salida. [4](#)
- prompt** Instrucción o pregunta que se proporciona a un modelo para obtener una respuesta. [36](#)
- renderización** Proceso de generar una representación visual en la interfaz de una aplicación. [66](#)
- scraping** Técnica para extraer datos de páginas web. [20](#)
- script** Archivo de texto que contiene una serie de instrucciones o comandos que son ejecutados por un intérprete de lenguaje de programación. [22](#)
- tokenizer** Herramienta que divide texto en unidades significativas, como palabras o subpalabras. [48](#)
- tokens** Unidades mínimas de procesamiento en NLP, que pueden representar palabras, subpalabras o caracteres. [13](#)
- transformers** Arquitectura de red neuronal que procesa tareas de secuencia a secuencia, facilitando el manejo de dependencias a largo plazo. [9](#)
- UMAP** Técnica de reducción de dimensionalidad que preserva la estructura global y local de los datos. [16](#)
- vectorización** Proceso de convertir datos, como texto, en vectores numéricos. [16](#)
- épocas de entrenamiento** Una recorrida completa por todo el conjunto de datos durante el entrenamiento de un modelo. [48](#)

Anexo A

Trabajos relacionados

En este anexo se presentan los detalles adicionales de las investigaciones discutidas en la sección 2.2 de este trabajo. Se profundiza en los enfoques metodológicos utilizados, los modelos implementados y las particularidades de los experimentos llevados a cabo en cada caso. El objetivo es ofrecer un panorama más completo de las investigaciones que fundamentan el desarrollo del presente proyecto, proporcionando información técnica y resultados relevantes que no fueron abordados en detalle en el cuerpo principal del texto.

A.1. Detección de opiniones

La investigación Sueca mencionada sobre detección de opiniones ataca varios de los problemas que nosotros debemos resolver. La investigación no solo presenta un método para identificar opiniones en un texto, sino que también identifica a quién sostiene dicha opinión y el tema de la misma. El estudio fue realizado sobre textos de noticias periodísticas, lo cual, lo hace mas interesante aun, dada la cercanía con el corpus que poseemos nosotros, el cual alberga noticias periodísticas uruguayas (Kim y Hovy, 2006).

La idea básica de la investigación radica en explorar cómo se relacionan semánticamente un titular de opinión y un tema con una palabra portadora de opinión en una frase. Dada una frase y una palabra disparadora de opinión, el método construido identifica los frames de la frase y busca qué frame corresponde al titular de la opinión y cuál al tema.

Para llevar a cabo su enfoque, los investigadores desglosaron el proceso en tres pasos principales. En el primer paso, crearon un conjunto de verbos y adjetivos que actúan como disparadores de opinión en los textos. Estos disparadores se clasificaron en función de su connotación emocional: positivos, negativos y neutros. Esta clasificación es crucial porque permite distinguir el tono de la opinión expresada y ayuda a identificar la presencia de opiniones en las frases. Al categorizar los disparadores por sentimiento, los investigadores pudieron estructurar mejor el análisis de las opiniones dentro del texto.

En el segundo paso, se centraron en la asignación de frames semánticos a los disparadores de opinión identificados. Utilizaron el corpus de FrameNet, una base de datos que proporciona información sobre la relación entre palabras y los frames semánticos que representan. Los frames semánticos son estructuras conceptuales que ayudan a interpretar el significado de las palabras en contextos específicos. Para los disparadores que no estaban registrados en FrameNet, los investigadores aplicaron una técnica de clustering. Esta técnica agrupó palabras similares en función de sus contextos de uso, permitiendo predecir el frame más probable para cada disparador no documentado en FrameNet. De este modo, mapearon todas las palabras con un frame semántico dado.

El tercer paso involucró la identificación y clasificación de los frames en las oraciones. Utilizaron el parser de Charniak para analizar la estructura gramatical de las oraciones y extraer los frames presentes en ellas. Un parser es una herramienta que descompone las oraciones en sus componentes gramaticales. Con los frames identificados, entrenaron un modelo de Máxima Entropía para clasificar estos frames en categorías específicas, como Estímulo y Experimentador. El sistema luego selecciona el titular de la opinión y el tema entre los elementos clasificados. Por ejemplo, en un frame de "Deseo", los elementos típicos incluyen Evento, Experienciador, Ubicación del evento y Participante focal. En este contexto, el "Experienciador" puede ser considerado el titular de la opinión, mientras que el "Participante focal" puede ser el tema, siempre que estos tipos estén presentes en la oración.

Los resultados obtenidos en esta investigación indican que el sistema supera significativamente al de referencia en la identificación de titulares y temas de opiniones.

A.2. Clustering

La investigación mencionada sobre clustering trabaja sobre la detección de tópicos a través de la clusterización y para eso, los autores utilizaron embeddings creados con BERT, UMAP y HDBSCAN para clusterizar los tópicos. Al igual que la investigación mencionada anteriormente en esta sección, este trabajo se basó en un corpus de artículos de noticias online, específicamente un conjunto de 10,000 artículos en inglés, recolectados de 58 editores diferentes. (Kim y Hovy, 2006).

El pipeline de modelado de temas utilizado en esta investigación es llamado BERTopic y sigue una secuencia de pasos diseñada para extraer y agrupar temas relevantes a partir de un conjunto de datos de noticias. El proceso inicia con la vectorización del texto mediante el modelo BERT, con fine-tuning realizado por los autores, obteniendo así, vectores representativos de cada texto.

El siguiente paso realizado por ellos fue la reducción de dimensionalidad, dado que los vectores generados por BERT son de alta dimensión, y dificultaban su visualización y análisis. Para resolver este problema, utilizaron UMAP (Uniform Manifold Approximation and Projection), una técnica que simplifica la representación de estos vectores pero manteniendo la estructura y las relaciones importantes entre ellos, como lo son la proximidad y la distancia.

Finalmente, culminan su pipeline con el agrupamiento de los vectores reducidos mediante HDBSCAN. HDBSCAN es un algoritmo de agrupamiento que destaca por su capacidad para determinar de manera dinámica el número y el tamaño de los clústeres, adaptándose a la densidad de los datos. A diferencia de métodos más rígidos como K-Means, HDBSCAN permite una asignación suave a los clústeres, asegurando que cada punto de datos esté incluido en un clúster y evita la clasificación de puntos como atípicos.

El estudio resultó en 63 clústeres de tamaños variados, siendo el más grande de 3,347 artículos y el más pequeño de alrededor de 20 artículos. De los 10,000 artículos, se analizaron manualmente 2,367, y los evaluadores coincidieron en el 95% de los casos sobre los artículos solapados. Identificaron 20 temas principales, como Entretenimiento, Deportes y Política, que abarcan varios clústeres.

El 52% de los temas fueron considerados coherentes por los evaluadores, con una puntuación promedio de coherencia del 96%. Sin embargo, solo el 57% del conjunto de datos se clasificó en temas coherentes. El tema más grande, que contenía 3,347 artículos, fue etiquetado como Incoherente, lo que afectó la puntuación global de coherencia del modelo. En cuanto a los clústeres incoherentes, los evaluadores detectaron que dichos textos solían ser artículos cortos y con características similares que dificultaban la coherencia temática.

A.3. Análisis de Sentimientos

La investigación realizada en la Facultad de Ingeniería de la UDELAR en 2015 sobre análisis de sentimientos pretendió determinar el sentimiento de una opinión utilizando un corpus proporcionado por un programa previamente construido llamado BuscOpiniones. (Guillermo Dufort y Álvarez, 2016)

La particularidad de esta investigación radica en que trabajó sobre un corpus de noticias periodísticas uruguayas, a diferencia de la mayoría de las investigaciones que utilizan corpus en inglés. Esto resulta particularmente relevante para nosotros, ya que los modelos pueden variar considerablemente sus resultados dependiendo del idioma sobre el cual son entrenados y aplicados. Además, este estudio utilizó un subconjunto del corpus que mencionaba a políticos uruguayos, lo que añade un nivel adicional de interés.

Durante la investigación, se evaluaron tanto métodos basados en reglas, métodos de aprendizaje automático y métodos híbridos que combinaban ambas aproximaciones.

Para los métodos basados en reglas, se construyeron algoritmos que realizaban un análisis sintáctico de las dependencias en las oraciones, creando un árbol de dependencias evaluado en función de los pesos de las palabras y sus roles sintácticos. La polaridad de cada palabra se asignaba utilizando un diccionario de palabras llamado SODictionariesV1.11.Spa. Se crearon múltiples algoritmos, dependiendo de si incluían la negación y/o intensificación, lo que permitió comparar la eficiencia de los mismos. Una de las complejidades de este enfoque radica en la necesidad de actualizar el algoritmo cada vez que se quiera considerar un nuevo aspecto, a diferencia de los métodos de aprendizaje automático, que evolucionan conforme son entrenados.

En cuanto al enfoque de aprendizaje supervisado, se anotaron manualmente opiniones para entrenar un modelo con dichos datos y permitirle clasificar nuevos textos en diferentes categorías de sentimiento. Se utilizaron los algoritmos de Support Vector Machines (SVM) y Multinomial Naïve Bayes (MNB) debido a su eficacia demostrada en tareas de análisis de sentimiento. Y utilizaron TF-IDF y cuentas a la hora de transformar las opiniones en vectores numéricos. Finalmente, se experimentó con diferentes configuraciones de las features y se ajustaron los pesos de las clases para optimizar el rendimiento del modelo.

Los mejores resultados con el método basado en reglas se obtuvieron con el algoritmo que incluía tanto la negación como la intensificación, alcanzando un 62 % de accuracy, mientras que en los métodos basados en aprendizaje automático, SVM fue el más efectivo, con un 61 % de accuracy, y se observó que el uso de bigramas y trigramas no mejoró los resultados, contrariamente a lo esperado.

El método híbrido desarrollado en esta investigación combina la transformación de cuentas de palabras con la incorporación del sistema de reglas como una feature adicional. Esto resultó en una mejora notable en el rendimiento del modelo, alcanzando un 64 % tanto en accuracy.

A.4. Nuevo enfoque sobre drogas - El país

Al hacer entrega del documento, Insulza dijo que el informe no era una conclusión, sino sólo el inicio de un debate largamente esperado”. Un tema que ya se encuentra en el Informe, donde se expresa que su propósito es “asistir a las Jefas y Jefes de Estado y de Gobierno de las Américas en la definición de un marco de referencia para el tratamiento del problema, así como para futuras políticas y acciones multilaterales”. Por su parte, el Presidente colombiano advirtió que debía quedar claro que “aquí nadie está defendiendo ninguna postura, ni la legalización, ni la regulación, ni la guerra a cualquier precio. Lo que tenemos que hacer es usar estudios serios y ponderados como el que hoy nos presenta la OEA para buscar mejores soluciones”. Y agregó: “No tengo dudas de que todos compartimos un destino común, en lo que a veces diferimos es en el cómo, y este informe nos va a ayudar a ponernos de acuerdo, será la base para una discusión tan aplazada”. Una discusión que pronto se ampliará a otros foros.

El Informe será considerado en la próxima Asamblea de la OEA, que tendrá lugar en Antigua, Guatemala, el mes que viene y será un aporte para el examen que realizará la Comisión de Estupefacientes de las Naciones Unidas, en marzo del año que viene, sobre los progresos realizados en la aplicación de su estrategia para contrarrestar el problema mundial de las drogas. La Asamblea General de la organización mundial ha convocado a un período extraordinario de sesiones sobre el problema mundial de las drogas, a comienzos de 2016.

El Informe, señala que existen ejemplos de buenas prácticas que están disponibles para ser utilizadas en otros lugares; se cuenta con un mejor conocimiento acerca de los factores que llevan al consumo de drogas y de los factores de riesgo que explican por qué una minoría de usuarios eventualmente desarrolla un consumo problemático; y se ha comprobado cambios en los patrones de consumo. Lo más importante es que se ha cristalizado una percepción más amplia de la violencia y del sufrimiento relacionado con el problema de las drogas. Ello ha creado dudas acerca de los costos económicos y sociales de las actuales estrategias para controlar las drogas.

La primera conclusión del Informe es que "el problema de las drogas es un tema hemisférico". Es una redacción cuidadosa. El término "hemisférico" incluye a los Estados Unidos.

Es indiscutible que, como señala el documento, el impacto del narcotráfico "en la economía, las relaciones sociales, la seguridad y la gobernabilidad democrática es mayor en los países de cultivo, producción y tránsito situados en América del Sur, América Central, México y en el Caribe". Y agrega: "En los países de Norteamérica, que son los principales lugares de destino final (no la marihuana que es 50% y 50%) de las sustancias traficadas, esas manifestaciones del problema son mucho menores". Aunque existan problemas compartidos entre el Norte y el Sur (por ejemplo, el lavado de activos), la peor parte de la guerra contra la droga la soporta este último, incluyendo a México, Colombia y los demás países latinoamericanos.

Quizás, la principal conclusión del Informe sea que "es necesario enfrentar el consumo de drogas con un enfoque de salud pública", como "una enfermedad crónica y recurrente, que requiere un enfoque integrado". El estudio reconoce que el cambio fundamental en esta materia va desde considerar al usuario como una víctima, un adicto crónico y no como un delincuente o cómplice del narcotráfico; concluye: "la despenalización del consumo de drogas de-

be ser considerada en la base de cualquier estrategia de salud pública”.En Cartagena, el Presidente Obama advirtió que no estaba de acuerdo con la despenalización pero dijo que ofrece una perspectiva nueva y abre el camino para una discusión que debería involucrar a todas las sociedades en nuestro hemisferio. Y que sería deseable culmine en consensos que permitan una acción coordinada y eficaz a nivel regional y nacional.

A.5. Ejemplo de secuencia de prompts para modelos generativos

Primer prompt

Tengo una lista de palabras que pueden aparecer en distintas noticias y refieren a personas u organizaciones opinando sobre algún tema. Quiero que las analices, y las recuerdes, porque luego te enviaré ciertas noticias, y te voy a solicitar que me retournes las opiniones de ciertas personas u organizaciones en dichas noticias, y para eso quiero que te ayudes con esta lista de palabras, ya que te será de utilidad para reconocer cuando alguien esta opinando.

La lista es la siguiente:

```
1 ['aboga', 'aclara', 'aclararon', 'aclaro', 'aclaró', 'acota',  
  'acotaba', 'acotó', 'acusa', 'acusó',  
2 'adelantó', 'admite', 'admitieron', 'admitió', 'admitía', '  
  adujo', 'advertía', 'advertían', 'advierte',  
3 'advierten', 'advirtieron', 'advirtió', 'afirma', 'afirmaba',  
  'afirman', 'afirmo', 'afirmó', 'agrega',  
4 'agregaba', 'agregan', 'agregando', 'agregaron', 'agregó', '  
  alertó', 'amenazaba', 'analiza', 'analizó',  
5 'anotó', 'anticipó', 'anuncia', 'anunció', 'aporta', 'apunta  
  ', 'apuntaron', 'apuntó', 'argumenta',  
6 'argumentó', 'arguyó', 'asegura', 'aseguran', 'aseguraron', '  
  aseguró', 'asentía', 'asevera', 'aseveró',  
7 'asume', 'atacó', 'atinó', 'aumentaron', 'aventuran', '  
  aventuró', 'añade', 'añaden', 'añadió',  
8 'bromea', 'bromeo', 'bromeó', 'calificaba', 'cantó', 'citando  
  ', 'clamó', 'cobijan', 'coincide',  
9 'coincidió', 'comenta', 'comentaban', 'comentan', 'comentaron  
  ', 'comentó', 'compara', 'comparó',  
10 'complementa', 'complementó', 'completó', 'comunicó', '  
  concluyen', 'concluyó', 'concluía', 'confesó',
```

11 'confiesa', 'confirma', 'confirmaron', 'confirmó', 'confió',
'considera', 'consideró', 'consignó',
12 'contaron', 'contesta', 'contestó', 'contó', 'coreaba', '
corearon', 'criticaba', 'criticó', 'cuenta',
13 'cuentan', 'cuestiona', 'cuestionó', 'declara', 'declaración'
, 'declararon', 'declararía', 'declaró',
14 'decía', 'decían', 'defiende', 'define', 'definió', '
denuncian', 'denunciaron', 'denunció', 'desafió',
15 'desafía', 'descargó', 'describe', 'describen', 'describió',
'desea', 'destaca', 'destacó', 'detalla',
16 'detalló', 'determinó', 'dice', 'dicen', 'diciéndoles', '
dictaminó', 'difundió', 'dijeron', 'dijo',
17 'dispararon', 'disparó', 'distingue', 'ejemplifica', '
ejemplificó', 'enfatisa', 'enfatisó', 'entonan',
18 'enumera', 'enumeró', 'escribe', 'escriben', 'escribieron', '
escribió', 'escribía', 'especifica',
19 'especificó', 'especula', 'especuló', 'espetó', 'establece',
'estalló', 'estima', 'estimaron',
20 'estimó', 'estuvo', 'evaluó', 'exclama', 'exclamó', 'explica'
, 'explicaba', 'explican', 'explicaron',
21 'explicó', 'expone', 'expresa', 'expresaba', 'expresando', '
expresaron', 'expresó', 'expuso',
22 'fustiga', 'fustigó', 'graficó', 'grita', 'gritaba', '
gritaban', 'gritaron', 'gritó', 'ilustra',
23 'incluyó', 'indica', 'indicaba', 'indican', 'indicaron', '
indicó', 'informa', 'informaron', 'informó',
24 'insistió', 'intercambió', 'interviene', 'ironizó', 'irrumpió'
, 'justifica', 'justificó', 'lamenta',
25 'manifestó', 'matizó', 'memora', 'mencionando', 'mencionó', '
murmuró', 'narró', 'notó', 'observa',
26 'observó', 'opina', 'opinó', 'osó', 'parecían', 'pensaron', '
pensé', 'pide', 'pidió', 'plantea',
27 'planteó', 'precisa', 'precisó', 'pregunta', 'preguntó', '
presentaban', 'previno', 'proclamó',
28 'profesa', 'prometió', 'pronostica', 'prosiguió', 'protesta',
'protestan', 'protestó', 'publica',
29 'publicaba', 'publicaron', 'publicó', 'puntualiza', '
puntualizó', 'ratificó', 'razona', 'razonó',
30 'recalca', 'recalcando', 'recalcó', 'reclama', 'reclaman', '
reclamó', 'recomendó', 'recomienda',
31 'reconoce', 'reconoció', 'recordó', 'recuerda', 'refiriéndose'
, 'reflexiona', 'reflexionó',
32 'refutando', 'refutó', 'reiteran', 'reiteró', 'relata', '
relataron', 'relató', 'remarca', 'remarcó',
33 'remata', 'remató', 'rememora', 'rememoró', 'repetía', 'repet

```
ían', 'repite', 'repitió', 'replicó',  
34 'resalta', 'resaltó', 'responde', 'responden', 'respondió', '  
respondía', 'resume', 'resumió', 'retoma',  
35 'retruca', 'revela', 'reveló', 'saludó', 'sentencia', '  
sentencian', 'sentenció', 'señala', 'señalaba',  
36 'señalaban', 'señalan', 'señalaron', 'señalo', 'señaló', '  
sintetiza', 'sintetizó', 'solicitó',  
37 'sostenía', 'sostiene', 'sostienen', 'sostuvo', 'subraya', '  
subrayó', 'suelta', 'sugiere',  
38 'sugiriendo', 'sugirió', 'suspira', 'teoriza', 'torea', '  
transmitió', 'tuiteaba', 'tuiteó', 'valoró',  
39 'vaticinaba', 'vaticinó', 'zanjó']
```

Segundo prompt

A modo de ejemplo, te envío una noticia de prensa con sus entidades, y el resultado esperado, para que aprendas lo que quiero recibir con las otras noticias.

Esta es la noticia de ejemplo:

El presidente de la República, José Mujica, dijo que no hay que derogar el delito de abuso de funciones hasta que la justicia se expida sobre el caso Pluna. Además, señaló que el préstamo para la cooperativa Alas U está en suspenso porque no hay respaldo jurídico para otorgarlo. “Necesitamos una figura jurídica que no está en la mano. Tenemos la mayor voluntad. Pero no se arregla solo con eso, porque son recursos públicos”, destacó el mandatario en una entrevista realizada a Telemundo. Manifestó que no se ha encontrado tal figura. “Soy partidario por lo siguiente: más vale arriesgar para la gente pueda trabajar y pagar menos en seguro de paro. Lo que se paga en seguro de paro es plata que se va. La otra es plata que se juega en el riesgo”. Mujica manifestó que se trata de “plata pública. Sino después metemos la pata, nos agarra la justicia y nos pone abuso de funciones”. Consultado sobre la idea de eliminar el delito de “abuso de funciones”, el presidente destacó que es un delito que “hace años que hay que eliminarlo porque sirve para un barrido y fregado, pero hay que sustituirlo por algo y cambiarlo ahora porque tenemos una acusación no te la llevo. Aguantamos todo lo que haya que aguantar y después en todo caso lo cambiamos pero no antes. Pluna. El mandatario sostuvo que .^{el} error estuvo en la ansiedad de querer solucionar un problema que era muy difícil, salvar la mayor cantidad de puestos de trabajo, tener una línea aérea que parece que el Uruguay se desmoronaba la temporada

turística y cuestiones por el estilo si no teníamos una línea aérea durante muchos años hemos sostenido esa idea. Después los hechos nos demostraron que estábamos equivocados”. Mujica volvió a cuestionar el asesoramiento que hubo en el caso: ”Si me traen un profesor grado 5 de derecho comercial de la Udelar, yo que soy un paisano qué voy a discutir con el tipo. Me como la pastilla y nos comimos la pastilla. El tipo se equivocó, porque los académicos también se equivocan y allá marchamos, pero no hubo mala fe. No hubo como aquellas cosas: Banco Pan de Azucar, los hermanos Rohn, los Peirano, todo aquello. Fue una jugada equivocada llena de ansiedad”, puntualizó. Mujica dijo que a Fernando Lorenzo no se le pidió la renuncia sino lo contrario, quien renunció fue él. Fernando Calloia. El mandatario pidió ”dejar tranquilo al Banco República” porque .^{en} el Brou está el 50 % del movimiento de Uruguay y a fin de año las empresas deben pagar los aguinaldos y muchas llegan con la lengua afuera y ahí funcional la experiencia y confianza del banco”. Mujica pidió que ”dejen pasar enero para conversar” sobre el tema.

Esta es la lista de entidades para esa noticia de ejemplo:

```
[
    'José Mujica',
    'Telemundo',
    'Mujica',
    'Peirano',
    'Fernando Lorenzo',
    'Banco República'
]
```

Tercer prompt

Este es el resultado y formato esperado para esa noticia con esas entidades:

```
1 {
2 "José Mujica": [
3 "dijo que no hay que derogar el delito de abuso de funciones
   hasta que la justicia se expida sobre el caso Pluna",
4 "señaló que el préstamo para la cooperativa Alas U está en
   suspenso porque no hay respaldo jurídico para otorgarlo",
5 "destacó la necesidad de una figura jurídica para otorgar el
   préstamo y la importancia de arriesgar para que la gente
   pueda trabajar, en comparación con el seguro de paro",
6 "manifestó que el tratamiento de los fondos como 'plata pú
   blica' requiere precaución para evitar acusaciones de
```

```

    abuso de funciones",
7 "criticó la idea de eliminar el delito de 'abuso de funciones
  ' sin tener una alternativa, mencionando la necesidad de
  esperar ante acusaciones actuales",
8 "reconoció el error en la gestión del caso Pluna, atribuyendo
  el fallo a la ansiedad por solucionar problemas laborales
  y turísticos",
9 "cuestionó el asesoramiento recibido en el caso Pluna,
  comparando la situación con errores anteriores en el á
  mbito financiero uruguayo",
10 "aclaró que a Fernando Lorenzo no se le pidió renunciar, sino
  que la renuncia fue decisión de Lorenzo",
11 "pidió dejar tranquilo al Banco República, destacando su
  importancia para la economía uruguaya y la necesidad de
  esperar hasta enero para discutir temas pendientes"
12 ],
13 "Telemundo": null,
14 "Mujica": [
15 "manifestó su postura de arriesgar en decisiones financieras
  para beneficiar el empleo frente a los costos del seguro
  de paro",
16 "reiteró su enfoque en la responsabilidad de manejar fondos p
  úblicos y las implicaciones legales de sus decisiones",
17 "expresó su perspectiva sobre los errores y aprendizajes en
  la gestión de crisis, especialmente en relación con el
  caso Pluna",
18 "comentó sobre el proceso de toma de decisiones y el papel de
  los asesores, reflexionando sobre su confianza y los
  resultados",
19 "destacó la importancia de la prudencia y la planificación en
  la administración de recursos clave como el Banco Repú
  blica"
20 ],
21 "Peirano": null,
22 "Fernando Lorenzo": null,
23 "Banco República": null
24 }

```

Prompt final

Quiero que realices eso mismo, pero para esta noticia:

El Sindicato Único de Trabajadores del INAU se declaró hoy en “estado de alerta”, tras el último “hecho de violencia ocurrido en el Hogar Ser”, donde un funcionario fue agredido por un adolescente con agua hirviendo. Median-

te un comunicado, el gremio señaló que “la situación es preocupante” y se necesitan “rápidas respuestas que aseguren que este tipo de hechos, los cuales con un gran compromiso de los trabajadores se habían desterrado, no vuelvan a ocurrir”. El pasado lunes, un trabajador de la Colonia Berro fue brutalmente agredido por un adolescente que le tiró un termo de agua caliente en la cara. Según contó en esa oportunidad el presidente del sindicato, Joselo López, el joven en rehabilitación mantuvo un enfrentamiento verbal con el trabajador y, tras una tensa discusión, le arrojó agua hirviendo en el rostro. Hoy, el gremio aclaró que “es oportuno señalar, que el ingreso de 300 nuevos funcionarios a la Institución ha generado un impacto positivo en las condiciones de trabajo, lo que entendemos es sumamente importante, las que se han visto empañadas por el significativo aumento de adolescentes privados de libertad”. Sin embargo, consideró importante “la inmediata intervención de las autoridades, en la concreción de nuevas instalaciones, que aumenten la capacidad locativa, y descomprima el hacinamiento que existe en todos los centros. Hacinamiento, que vulnera el derecho de los adolescentes y trabajadores, potenciando la violencia en el sistema”. Por último, el sindicato señaló que no descarta “ninguna medida mientras siga en peligro la integridad física de los trabajadores que este sindicato representa”.

Esta es la lista de entidades para las cuales quiero saber sus opiniones:

['Único', 'INAU', 'Colonia Berro', 'Joselo López']