Hacia el *fact-checking* automático: Un estudio exploratorio sobre la identificación de frases críticas para verificación

Galo Emanuel Pianciola Bartol
Facultad de Ciencias Exactas, UNICEN
Buenos Aires, Argentina
gpianciolabartol@alumnos.exa.unicen.edu.ar

Antonela Tommasel
ISISTAN, CONICET & UNICEN
Buenos Aires, Argentina
antonela.tommasel@isistan.unicen.edu.ar

***Abstract*—En un contexto saturado de información en los medios sociales, es esencial priorizar la verificación de afirmaciones que tienen el potencial de ser falsas o engañosas. Esto ha impulsado el *fact-checking*, un proceso dedicado a analizar afirmaciones con el fin de verificar su veracidad. Dada la limitación de recursos humanos para examinar todas las afirmaciones en Internet, resulta vital identificar aquellas que resulta crítico verificar. Por consiguiente, se requiere un sistema (semi-)automatizado con la capacidad de detectar las afirmaciones más urgentes y relevantes para su verificación. Para abordar este desafío, se propone un enfoque basado en técnicas de Procesamiento de Lenguaje Natural y *Machine Learning*. Utilizando colecciones de datos incluyendo tanto *tweets* como frases chequeables de discursos políticos, se clasificarán las frases según su relevancia para ser verificadas (es decir, su *check-worthiness*). Se explorarán tanto características léxicas como modelos de *embeddings*, LLMs y técnicas tradicionales de clasificación para desarrollar un sistema automatizado de evaluación del *check-worthiness* de frases. Las técnicas basadas en embeddings y LLMs mostraron un gran potencial para mejorar la eficiencia de los procesos de verificación de frases, al lograr priorizar efectivamente aquellas más críticas y relevantes para su revisión.**

***Index Terms*—*check-worthiness*, *fact-checking*, medios sociales, análisis de texto**

I. Introducción

Actualmente, los políticos y otros actores públicos se comunican directamente con sus audiencias a través de los medios sociales, eludiendo los filtros de los canales tradicionales de comunicación. Si bien esta libre expresión y comunicación directa tiene sus ventajas, también provoca la difusión de afirmaciones inexactas o engañosas, comúnmente conocidas como desinformación o *fake news* [1, 2]. Aunque la propagación de desinformación no es algo nuevo, su alcance, viralización y efectividad han aumentado significativamente en la actualidad gracias a Internet y los medios sociales [3]. Esto facilita la tergiversación de datos cruciales, tanto de manera intencionada como involuntaria, lo que resulta en la difusión de desinformación con mínimas o nulas consecuencias [2]; pudiendo llegar a millones de usuarios y dirigirse fácilmente a diferentes grupos según su perfil geográfico, demográfico, psicológico y/o político.

Asimismo, dicha propagación suscita inquietudes acerca de cómo se manipulan las opiniones públicas [4, 5]. Cuando son las propias agencias gubernamentales y los políticos quienes difunden información incorrecta o sesgada, también surge preocupación debido al creciente escepticismo hacia ellos y las instituciones. A largo plazo, esta desconfianza puede conducir a la apatía política y una menor participación en los procesos democráticos, lo cual puede afectar directamente a las democracias [6, 7]. En este contexto, resulta crucial verificar la veracidad y autenticidad de la información compartida.

El *fact-checking* puede definirse como el proceso de analizar afirmaciones a fines de verificar los hechos enunciados [2]. En los últimos tiempos, han surgido numerosas organizaciones dedicadas a esta tarea, con más de 50 iniciativas en todo el mundo. Se trata de organizaciones periodísticas que tienen como objetivo verificar la información que circula públicamente, siendo algunas de las más relevantes *FactCheck.org* y *Politifact* de Estados Unidos, y *Full Fact* en Inglaterra. En Latinoamérica, *Chequeado*[1] ha sido pionera desde 2010, en conjunto con otros proyectos en Brasil, Honduras, Uruguay, Colombia y entre otros países [3, 8].

Considerando el volumen de (des)información que circula diariamente en los diversos canales de comunicación a gran escala, y el número limitado de recursos disponibles para el *fact-checking*, los periodistas y verificadores (o *fact-checkers*) se enfrentan a desafíos para mantener un flujo de trabajo constante, inmediato y ágil. Esto puede dar lugar a un desfase temporal poco deseable desde el punto de vista periodístico, entre la viralización de una afirmación (por ejemplo, por parte de un político) y su verificación [9].

El *fact-checking* ha demostrado ser una tarea manual altamente exigente, los verificadores dedican mucho tiempo a buscar afirmaciones para verificar, analizando transcripciones de discursos, debates y entrevistas [1]. Esto limita la cantidad de frases que pueden ser verificadas, especialmente frente al enorme volumen de información compartida diariamente en noticias, medios sociales y organismos gubernamentales. Por lo tanto, resulta esencial automatizar este proceso [10].

[1] https://chequeado.com/

El proceso de *fact-checking* incluye diversas tareas como, i) la detección de frases verificables que incluyen afirmaciones fácticas de hechos; ii) la identificación de frases verificables cuya verificación sea crítica en términos de su probable falsedad en la información, su potencial impacto negativo en la sociedad, individuos, compañías o productos, su relevancia general y la necesidad de la intervención de un profesional que se ocupe de chequearla [11]; y iii) la contrastación de las frases con la evidencia disponible [2, 12].

Además de la verificación de afirmaciones y de la identificación de frases que pueden ser verificadas, una tarea fundamental del proceso es determinar qué frases son críticas para verificar, es decir, determinar el *check-worthiness* de las mismas. Esta tarea busca emular la estrategia de selección de frases en una organización de *fact-checking*, tratando de automatizar el criterio de selección de las frases, considerando preguntas como: *¿hasta qué punto aparenta contener información falsa?, ¿resulta dañina para la sociedad, una persona, una compañía o producto?, ¿es de interés general? e incluso ¿debería un profesional ocuparse de chequearla?* [11].

Este trabajo tiene como objetivo *determinar si una frase es crítica o no para ser verificada*. Para ello, partiendo de un conjunto de frases verificables, es decir, aquellas que pueden ser confrontadas con hechos y datos contrastables que están registrados en algún lado al que se puede acceder de manera pública[2] [13], se busca determinar su *check-worthiness* o si son críticas de ser verificadas. Para lograr dicho objetivo, se compararon diversas técnicas de Procesamiento de Lenguaje Natural (NLP) y *Machine Learning*, incluyendo enfoques basados en características léxicas, y modelos de *embeddings*, así como también diferentes modelos de clasificación, tanto tradicionales como basados en LLMs. Los resultados indican un potencial prometedor de las técnicas basadas en *embeddings* y de los LLMs para la clasificación con pocos datos disponibles. Ambas técnicas poseen un gran potencial para mejorar la eficiencia de los procesos de verificación de hechos, al lograr priorizar efectivamente las afirmaciones más críticas y relevantes para su revisión. Automatizar esta tarea contribuiría a mitigar el sesgo humano en la selección de afirmaciones a ser verificadas, pudiendo potencialmente reducir el tiempo requerido para la tarea, permitiendo así a las organizaciones asignar recursos limitados a las tareas más relevantes, aumentando la cobertura de la verificación y, posiblemente, mejorando su efectividad.

El resto de este trabajo se organiza de la siguiente manera. La Sección II presenta trabajos relacionados. La Sección III describe la metodología del estudio, incluyendo los datos, las representaciones y los modelos de clasificación utilizados. La Sección IV presenta los resultados obtenidos. Finalmente, la Sección V presenta las conclusiones del estudio y las perspectivas futuras sobre la tarea.

II. Trabajos Relacionados

Detectar frases críticas para verificación puede ser un proceso que requiere una gran inversión de tiempo y recursos, los cuales suelen ser escasos en las organizaciones de *fact-checking*. Los verificadores eligen qué frases verificar de acuerdo con el esfuerzo requerido y las posibles consecuencias de no hacerlo, incluyendo el daño que podría causar en términos de riesgos para la salud, empeoramiento de emergencias (por ejemplo, respuestas a desastres naturales) o incluso el impacto en procesos democráticos. Además, aunque una afirmación pueda ser verificada, la información necesaria para hacerlo podría no estar directamente disponible, y buscarla en fuentes alternativas o consultar a expertos puede llevar días y no siempre arrojar resultados útiles [14]. En la literatura es posible encontrar diversas técnicas, en su mayoría evaluadas en idioma inglés, enfocadas en el uso de representaciones léxicas, semánticas, y muy recientemente, en la exploración de LLMs (*Large Language Models*).

A. Técnicas basadas en representaciones léxicas

Entre los primeros trabajos de la literatura basados en representaciones léxicas de las frases a analizar se encuentra *ClaimBuster* [15], una herramienta diseñada para determinar el *check-worhthiness* de frases extraídas de debates políticos en inglés. Las representaciones de las frases se construyeron utilizando TF-IDF (*Term Frequency – Inverse Document Frequency*), etiquetado de partes del discurso (POS, *Part-of-the-Speech*), detección de entidades, la longitud de las frases y análisis de sentimiento. Luego, la clasificación se realizó utilizando modelos tradicionales como *Naïve Bayes multinomial*, *SVM* y *Random Forest*.

Debido a la creciente importancia del *fact-checking*, desde hace varios años se lleva a cabo la competencia *CheckThat!*[3], la cual aborda diversas tareas relacionadas, además de disponibilizar múltiples colecciones de datos en diversos idiomas, entre ellos, el español. Estas colecciones no solo cubren cuestiones políticas, sino también la difusión de información sobre la pandemia de COVID-19 en medios sociales (en particular, *Twitter/X*). Entre los equipos participantes, se destacan McDonald et al. [16] y Tarannum et al. [17], quienes también abordaron el problema utilizando representaciones TF-IDF en la forma de unigramas, bigramas y trigramas con el fin de incorporar información contextual y secuencial en el análisis. En ambos casos preprocesaron el texto proveniente de medios sociales eliminando URLs, emojis, *stopwords*, símbolos de hashtag (#) y menciones (@); y en el caso de McDonald et al. [16] aplicando lematización. En ambos casos, la clasificación se basó en técnicas tradicionales como *Naïve Bayes*, *Random Forest*, *SVM*, *Gradient Boosting* y *AdaBoost*. De acuerdo con McDonald et al. [16], los mejores resultados se obtuvieron con *Random Forest*, el cual superó a otras alternativas más complejas basadas en representaciones semánticas y redes neuronales. Estos resultados sugieren la existencia de una dependencia de la tarea en el vocabulario utilizado en las frases, lo que podría deberse a la existencia de un único tópico común en los datos, afectando la capacidad de generalización de los modelos. Por su parte, Tarannum et al. [17] aplicaron técnicas de equilibrado de clases, lo que les

[2] https://chequeado.com/que-es-chequeable/

[3] https://checkthat.gitlab.io/

permitió mejorar sus resultados. Mientras que para inglés, los mejores resultados fueron obtenidos con modelos basados en redes neuronales, para español, los mejores resultados fueron obtenidos utilizando *Random Forest*, seguido por *SVM*.

B. Técnicas basadas en representaciones semánticas

El aprendizaje profundo ha transformado el campo del NLP al ofrecer métodos más efectivos para representar la complejidad del lenguaje, logrando capturar su semántica de manera más precisa. Varios estudios en la literatura, como los de Nikolov et al. [18], Tarannum et al. [17] y Zhou et al. [19], han abordado diversos aspectos del *fact-checking* utilizando este tipo de técnicas.

Nikolov et al. [18] emplearon representaciones basadas en *embeddings* (como *GloVe*, *FastText* y *Sent2vec*) y modelos prentrenados de *transformers*, (BERT base, RoBERTa base, DistilBERT y AlBERT base). Es importante destacar que no realizaron un *fine-tuning* de los modelos, sino que todos fueron utilizados para la generación de representaciones que luego se clasificaron utilizando técnicas tradicionales. Considerando la naturaleza preentrenada de los modelos y para reducir la existencia de términos desconocidos como COVID-19 y sus variantes, los autores aplicaron un preprocesamiento que reemplazaba estas expresiones por "ébola", termino conocido por los modelos (debido a la epidemia en el año 2014) y con una naturaleza similar a la de COVID. De forma complementaria, aplicaron preprocesamientos similares a los de Tarannum et al. [17]. Los mejores resultados fueron obtenidos combinando la representación generada con BERT y *SVM* y *Logistic Regression*. Tarannum et al. [17] también utilizaron representaciones basadas en BERT y XML-RoBERTa, las cuales para español, como ya se mencionó previamente, obtuvieron resultados inferiores a los de las representaciones tradicionales. Finalmente, Zhou et al. [19] se enfocaron en frases en inglés, utilizando representaciones basadas en Word2vec y BERT, combinadas con características derivadas de la cantidad de palabras, *hashtags*, URLs y signos de puntuación en las frases, entre otros. En este caso, los autores aplicaron un preprocesamiento más agresivo que incluía la normalización de puntuaciones, la eliminación de referencias a entidades (por ejemplo, personas, lugares y fechas), URLs, caracteres no representables en ASCII (con la excepción de los emojis), expansión de abreviaturas y *hashtags*, y unificación de términos relacionados al COVID-19. Con el objetivo de expandir el conjunto de datos, utilizaron la técnica de *Back Translation*, que consiste en traducir una frase temporalmente a otro idioma, y luego traducirla nuevamente al idioma original con el fin de producir nuevas frases con diferentes expresiones, manteniendo la semántica. Finalmente, realizaron *fine-tuning* de BERT, RoBERTa, BERTWWM, BERTweet y distilRoBERTa, las cuales obtuvieron los mejores resultados, superando ligeramente los resultados obtenidos con las representaciones de *embeddings* y clasificadores tradicionales. Asimismo, los resultados mostraron que la inclusión de características no relacionadas directamente con el contenido del texto (como por ejemplo, las referidas al uso de signos de puntuación) no permitieron mejorar significativamente los resultados.

Competencia	Partición	Cantidad de instancias		
		Total	Clase positiva	Clase negativa
CLEF-2021 *CheckThat!* [11]	Entrenamiento	952	200	752
	Desarrollo	475	109	366
	Prueba	476	120	356
CLEF-2022 *CheckThat!* [23]	Entrenamiento	4.990	1.903	3.087
	Desarrollo	2.500	305	2.195
	Prueba	2.500	305	2.195
CLEF-2023 *CheckThat!* [24]	Entrenamiento	7.488	2.208	5.280
	Desarrollo	2.460	299	2.161
	Prueba	5000	704	4.296

TABLE I: Descripción de los conjuntos de datos utilizados.

C. Técnicas basadas en LLMs

En los últimos años, los LLMs (*Large Language Models*) [20] han transformado el estado del arte en diversas tareas de NLP al ser preentrenados en grandes conjuntos de datos y luego realizar un *fine-tuning* de acuerdo a instrucciones humanas. En este contexto, una alternativa para la clasificación de texto es utilizar LLMs. Los LLMs constituyen un cambio de paradigma en NLP ya que facilitan el aprendizaje en contexto simplemente definiendo instrucciones en lenguaje natural [20, 21], lo que los hace aplicables en diversos dominios.

Agresti et al. [22] fueron pioneros en utilizar estos modelos para tareas relacionadas al *fact-checking*. En particular, utilizaron GPT-3 (OpenAI) para la detección de frases verificables, tanto para idioma inglés y español. Sin aplicar ningún preprocesamiento ni modificación de los datos, lograron destacarse en la competencia *CheckThat! 2022*, superando a modelos basados en representaciones y modelos de redes neuronales más complejos. Estos resultados mostraron que los LLMs pueden alcanzar resultados competitivos en comparación con modelos de *embeddings* como BERT, resaltando su utilidad para diversas tareas.

III. Descripción del estudio

El objetivo de este trabajo de determinar qué frases verificables son críticas de verificar. Para esto, se definió una tarea de clasificación binaria, donde cada frase se clasifica como crítica (*check-worthy*) o no (*no-check-worthy*) para su verificación. La Figura 1 presenta una representación esquemática del *pipeline* de procesamiento definido[4]. En primer lugar, se obtuvieron y preprocesaron los datos; luego se definieron diferentes tipos de representaciones. Tomando como base las representaciones construidas, se entrenaron diversos modelos de clasificación. Finalmente, se evaluó la performance de dichos modelos.

A. Colección de datos

Los datos utilizados provienen de varias ediciones de la competencia *CheckThat!*[5][6][7] en idioma español. La Tabla I presenta la distribución de los datos para cada colección y particiones. Para aumentar el conjunto de datos disponible,

[4]La implementación se encuentra disponible en https://github.com/galopianciola/check-worthiness-project

[5]https://gitlab.com/checkthat_lab/clef2021-checkthat-lab

[6]https://gitlab.com/checkthat_lab/clef2022-checkthat-lab/clef2022-checkthat-lab

[7]https://gitlab.com/checkthat_lab/clef2023-checkthat-lab

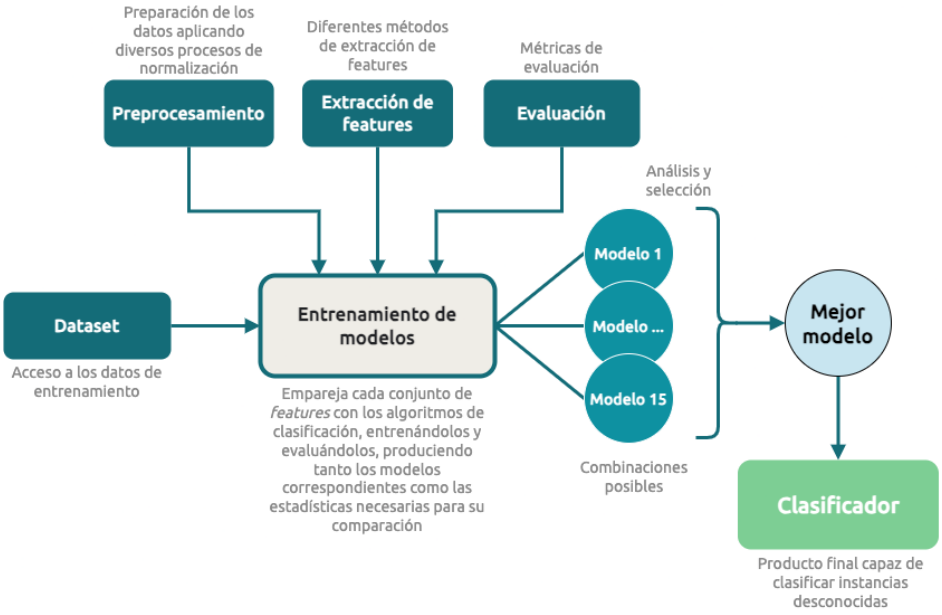

Fig. 1: *Representación esquemática de la metodología empleada*

se combinaron los datos de las diferentes competencias para permitir trabajarlos de forma conjunta. Asimismo, esta combinación también permite ampliar la variedad de temas y expresiones utilizadas en la colección, lo que puede ayudar a limitar la dependencia de los modelos entrenados en frases y palabras específicas, como fuera analizado por McDonald et al. [16].

B. Representación de frases

Una vez que se obtuvieron los datos, se procedió a definir las características que se utilizarían para describir las frases y entrenar los modelos de clasificación correspondientes. En primer lugar, se aplicaron diversas técnicas de preprocesamiento para normalizar las frases. Considerando la naturaleza informal de las frases extraídas de medios sociales, además de técnicas tradicionales como la tokenización, lematización, conversión a minúsculas, eliminación de *stopwords* y caracteres no alfabéticos, se aplicaron técnicas específicas para este tipo de datos:

• *Separación de hashtags en palabras según su capitalización.* Los *hashtags* a menudo siguen la convención *UpperCamelCase*, donde múltiples palabras se unen como una sola, con la primera letra de cada palabra en mayúscula. Por ejemplo «#HoyVotaArgentina» se compone de tres palabras: "hoy", "vota" y "argentina". Esta estrategia permite reducir la cantidad de palabras únicas o poco frecuentes, limitando la importancia del conjunto (lo que podría generar dependencias respecto a su aparición o no) y extrayendo así posiblemente información textual más relevante y facilitando la comprensión de las frases.

• *Reemplazo de URLs con el token especial* `URL`. Si bien la presencia de una URL puede tener un gran impacto en la decisión final del modelo sobre si una frase es crítica para ser verificada, el destino específico de dicha URL podría no ser relevante. El modelo entrenado podría tener dificultades en diferenciar la importancia de los distintos enlaces, pudiendo generar dependencias respecto a la aparición de determinadas fuentes (por ejemplo, de medios periodísticos).

• *Reemplazo de* `@` *con el token especial* `USER`. De forma similar a las URLs, si bien la aparición de menciones en las frases puede ser relevante porque pueden aumentar el impacto social de la afirmación, la identidad específica de los usuarios mencionados puede no serlo.

Trabajos previos [16, 17] han proporcionado guías sobre qué características podrían ser útiles para la tarea. Sin embargo, dichas características fueron mayormente propuestas para clasificar textos en inglés. Por lo tanto, es crucial continuar evaluando su relevancia para el idioma español. Para la representación de las frases se adoptaron diferentes representaciones basadas en *embeddings*, que permiten capturar la semántica de los textos de una manera más efectiva que representaciones solo basadas en aspectos léxicos.

• *Word2vec.* Este modelo define representaciones para cada palabra en la frase, las cuales luego se promedian para obtener la representación completa de la frase. Se utilizó un modelo entrenado con el texto *tokenizado* y preprocesado de cada uno de los casos de entrenamiento.

• *FastText.* Este modelo también genera una representación para cada palabra, las cuales se promedian para obtener la representación final de la frase. A diferencia del modelo anterior, *FastText* permite representar palabras fuera del vocabulario (es decir, palabras que no se encontraban en el texto utilizado para entrenar el modelo), lo que le otorga más robustez para la representación de textos que pueden diferir en contenido o tópico de los textos de entrenamiento. Se utilizó un modelo entrenado con el texto *tokenizado* y preprocesado de cada uno de los casos de entrenamiento.

• *InferSent.* Permite obtener una representación única para

cada frase. Fue entrenado con las representaciones de cada palabra obtenidas de *FastText*.

• *BETO*. Modelo preentrenado para español creado por la Universidad de Chile[8]. A diferencia de los modelos anteriores, genera representaciones a nivel de palabra que son contextuales, lo que significa que una misma palabra puede tener diferentes representaciones según el contexto en el que es utilizada. Para obtener la representación de la frase completa se utiliza la representación del token especial final que captura el significado general y el contexto de toda la frase.

C. Clasificación tradicional de frases

Como se mencionó previamente, la detección de frases críticas de ser verificadas se abordó como una tarea de clasificación binaria donde las frases se clasifican como "*check-worthy*" o "*no-check-worthy*". Para la evaluación, en primer lugar, se eligieron técnicas de clasificación comúnmente utilizadas en la literatura *SVM*, *Naïve Bayes*, *Random Forest* y *Logistic Regression*; las cuales ya fueron empleadas para esta tarea, por ejemplo, por McDonald et al. [16] y Tarannum et al. [17].

En ámbito de *machine learning*, el ensamble de modelos se ha establecido como una técnica efectiva para mejorar la *precision* y la performance de las tareas de clasificación [25]. Esta metodología consiste en combinar las predicciones de varios modelos para producir una única salida. En general, esta salida combinada obtiene una mejor performance que los modelos individuales que la componen. Esta técnica no solo aprovecha la diversidad entre los modelos, sino que también reduce el riesgo de *overfitting*, contribuyendo a una mayor generalización y robustez en la solución de problemas complejos. En este contexto, los clasificadores tradicionales previamente mencionados fueron combinados con cuatro modelos de ensamble: *Voting* (para cada instancia, se elige la clase más repetida entre los modelos base), *Bagging* (combina múltiples modelos entrenados en conjuntos de datos de entrenamiento generados mediante muestreo con reemplazo), *Stacking* (combina los modelos base utilizando un meta-modelo para producir una predicción final), y *Boosting* (combina los modelos de manera secuencial, donde cada modelo se enfoca en corregir los errores del anterior, se seleccionó *AdaBoost*).

Finalmente, el *fine-tuning* es una técnica de aprendizaje profundo que se aplica a modelos preentrenados. Consiste en tomar un modelo que ya ha sido entrenado en una tarea general (por ejemplo, BERT) y ajustarlo para que funcione en una tarea específica, como la clasificación de texto en este caso. Esta técnica tiene la ventaja de necesitar menos datos específicos de la tarea para lograr un alto rendimiento, ya que el modelo ya ha aprendido representaciones útiles del lenguaje durante su entrenamiento previo. En este contexto, se tomó el modelo BETO preentrenado y se le realizó *fine-tuning* para la detección de frases *check-worthy*.

D. Clasificación basada en LLMs

La entrada proporcionada al LLM para resolver una tarea es de vital importancia. Esta entrada, a menudo de carácter instruccional, se conoce como *prompt*. El *prompting* es lo que le permite a cualquier usuario (experto o no), interactuar con un LLM utilizando lenguaje natural de manera controlada, aprovechando su capacidad para comprender texto y generar una respuesta [26].

Al diseñar un *prompt*, es esencial definir el rol del LLM, es decir, qué función desempeña el LLM en la tarea o a qué perfil está representando. Esta definición tiene un impacto significativo en las respuestas, afectando no sólo en el contenido generado, sino también en el nivel de experiencia reflejado en la respuesta y consideraciones éticas [27]. En el contexto de este problema, se le asignó al LLM el rol de *periodista* de la siguiente manera:

> Sos un periodista experto en *fact-checking* y tu tarea es discernir de la forma más certera posible entre afirmaciones que merecen ser verificadas por un experto (check-worthy) y aquellas que no. Utiliza tu experiencia y criterio para evaluar cada afirmación.

Una vez definida la tarea de clasificación, se espera que el LLM generalice utilizando su comprensión de las instrucciones proporcionadas y pueda resolver una variedad de problemas, adaptándose a diferentes situaciones. En este contexto, se han establecido el contexto de la tarea y las siguientes instrucciones e indicaciones sobre la salida esperada del LLM:

> *[DESCRIPCIÓN TAREA]* Una frase que es crítica para ser verificada es aquella que amerita ser verificada en términos de su probable falsedad en la información, su potencial impacto negativo en la sociedad, individuos, compañías o productos, su relevancia general y la necesidad de la intervención de un profesional que se ocupe de chequearla.
>
> Realizá las siguientes tareas:
>
> 1) Identificá a qué categoría pertenece el texto dado con la mayor probabilidad.
> 2) Asigná el texto a dicha categoría.
> 3) Retorná la respuesta en formato JSON conteniendo solo la clave '*label*' y el valor correspondiente a la categoría asignada.

Los LLMs han demostrado habilidades destacadas de generalización en situaciones de aprendizaje *zero-shot* y *few-shot*. En contraste con las tareas tradicionales de clasificación de texto, donde se entrena un modelo para clasificar un conjunto predefinido de clases o etiquetas, en estos escenarios el modelo no se entrena con los datos disponibles (si los hay), sino que recibe un *prompt* que proporciona información sobre los datos (limitados) disponibles para llevar a cabo la tarea.

La estrategia *zero-shot* refiere a la capacidad de realizar una tarea sin haber visto ningún ejemplo de entrenamiento relacionado, aprovechando solo los grandes volúmenes de datos utilizados para el entrenamiento del modelo. De esta forma, el *prompt* definido incluye únicamente el texto a clasificar:

[8]https://github.com/dccuchile/beto

Se te dará la siguiente información:

1) Un texto delimitado por comillas triples.
2) Una lista de categorías a las cuales puede asignarse el texto. La lista está delimitada por corchetes. Las categorías en la lista se encuentran entre comillas simples y separadas por comas.

[DESCRIPCIÓN TAREA]

Lista de categorías: ["check-worthy", "non-check-worthy"]

Texto: "'{text}"'

Tu respuesta en formato JSON:

A pesar de la notable capacidad de estos modelos para generar respuestas en el contexto de aprendizaje *zero-shot*, la performance puede variar considerablemente dependiendo de la naturaleza y la complejidad de la tarea. Es aquí donde entra en juego el aprendizaje *few-shot*, donde se proporciona un número limitado de ejemplos relacionados en el *prompt*, los cuales sirven como marco de referencia para orientar al modelo en cómo responder a casos posteriores. Estos ejemplos pueden considerarse como una forma de "entrenamiento", aunque siguen siendo un número acotado de ejemplos en comparación a los necesarios para realizar un entrenamiento completo del modelo. En este enfoque, el *prompt* se modifica para incorporar los ejemplos a utilizar como entrenamiento. Los ejemplos utilizados fueron elegidos individualmente en base a las etiquetas de clase predichas con *zero-shot*. Los casos incorrectamente etiquetados como negativos se utilizan como ejemplos positivos, y viceversa para los ejemplos negativos[9]. Así, este método permite al modelo adaptarse mejor a la tarea específica y mejorar su *precision* en la clasificación, corrigiendo y aprendiendo de los errores cometidos en el aprendizaje *zero-shot*.

Para el propósito de este trabajo, se utilizaron tres modelos LLMs disponibles públicamente, dos *open source* (`Gemma:2b`, `Llama2:7b`) y uno comercial (`GPT-3.5 Turbo`).

E. Detalles de implementación

La implementación fue realizada en Python. El preprocesamiento de las frases fue realizado utilizando mayoritariamente *Spacy*. *Spacy* provee modelos preentrenados para tokenización, etiquetado POS y lematización. Los clasificadores fueron implementados utilizando *scikit-learn*. Para la configuración de las diferentes combinaciones de las representaciones y los clasificadores, se hizo uso de los *Transformers* de *scikit-learn*. No se requiere de hardware especial o de alta performance para la replicación del estudio.

Se accedió a los LLMs seleccionados a través del framework *LangChain*[10], configurando su temperatura en 0 para minimizar la aleatoriedad en las respuestas[11]. Cada texto se procesó en una interacción individual con *LangChain* y no se empleó memoria conversacional.

Los datos recolectados se dividieron en conjuntos de entrenamiento y evaluación siguiendo un modelo de validación cruzada estratificada k-fold, estableciendo k en 10. Se mantuvieron las mismas particiones de datos para todas las evaluaciones para garantizar la comparabilidad de los resultados. Para evitar la filtración de datos (*data leakage*), en el caso de las representaciones entrenadas desde cero (como *Word2vec* o *FastText*), el entrenamiento de los modelos se realizó solo considerando las características presentes en la partición de entrenamiento de cada *fold*. Esta estrategia de particionamiento se eligió debido a que no se requería considerar una relación temporal en los datos, evitando así la necesidad de una división temporal donde los datos de evaluación sean posteriores a los de entrenamiento.

La evaluación de los modelos se basó en *precision*, *recall* y *f-measure* para la clase positiva (*check-worthy*). Dado el desbalance significativo de clases en la colección de datos utilizada, se consideraron también métricas ponderadas que calculan las métricas para cada clase individualmente y luego promedian estas considerando el número de instancias en cada clase[12]. Dada la naturaleza de la tarea, se considera que es preferible contar con modelos con un mayor *recall* (es decir, que identifiquen a la mayoría de las frases *check-worthy*), aunque esto pueda resultar en una menor *precision*. Esto se debe a que minimizar el *recall* aumenta el riesgo de falsos negativos, es decir, omitir frases que podrían tener consecuencias significativas para la sociedad si no se verifican. Asimismo, también se incluyó el Coeficiente de Matthews [28], el cual suele ser más confiable que AUC-ROC para conjuntos de datos desbalanceados.

Finalmente, de forma complementaria, se realizó un análisis estadístico por pares de los resultados obtenidos para cada *fold* y cada combinación de representación y modelos de clasificación (definiendo un $\alpha = 0.01$). La hipótesis nula establecía que no había diferencia entre los resultados de las diferentes combinaciones (es decir, que tenían una performance similar), mientras que la hipótesis alternativa establecía que las diferencias observadas eran significativas y no accidentales.

IV. Evaluación Experimental

Es importante establecer un marco de referencia (es decir, los *baselines*) para la tarea a desarrollar definiendo estrategias triviales como la selección aleatoria de clase para cada instancia de evaluación (*Random*) y la selección de la clase mayoritaria (*Majority*), que muestren los valores los valores mínimos alcanzables. En términos generales, estas estrategias fueron superadas por las combinaciones evaluadas.

Asimismo, también a modo de comparación, se incluye una representación tradicional basada en TF-IDF, como fuera utilizada en trabajos anteriores [15, 16, 17]. Es importante destacar que, aunque la evaluación experimental se realizó con un conjunto predeterminado de frases, el análisis de frases

[9]El detalle de los ejemplos utilizados se encuentra en el repositorio que acompaña este trabajo.

[10]https://python.langchain.com

[11]https://platform.openai.com/docs/api-reference/chat

[12]Todas las métricas fueron implementadas utilizando scikit-learn: https://scikit-learn.org/stable/modules/model_evaluation.html

Modelo	*Precision*	*Weighted Precision*	*Recall*	*F-measure*	*Weighted F-measure*	*Accuracy*	*Balanced Accuracy*	*MCC*
Random	0,14	0,75	0,49	0,21	0,57	0,50	0,50	-0.0007
Majority	0,00	0,73	0,00	0,00	0,79	0,85	0,50	0,00

TABLE II: Resultados de evaluación con estrategias de referencia (*baselines*).

críticas para verificar puede ser utilizado en un entorno cuasi de tiempo real. En este escenario, técnicas como TF-IDF pueden no ser adecuadas, ya que, por un lado, si las frases llegan constantemente, no existiría un conjunto de frases fijo sobre el cual calcular el IDF, y por otro, al agregar nuevas frases a la colección, los puntajes de las características deberían ser actualizados periódicamente, lo que podría resultar ineficiente.

A. Representación tradicional

La Tabla III presenta los resultados obtenidos para la representación TF-IDF combinada con los diferentes modelos de clasificación seleccionados. Como puede observarse, para la mayoría de las métricas, *SVM* obtuvo los mejores resultados. Es importante reconocer el *trade-off* entre *precision* y *recall*, siendo que el incremento de uno, generalmente se ve acompañado por la disminución del otro. En este sentido, los resultados sugieren que aunque *SVM* es capaz de detectar una gran proporción de las frases críticas, también clasifica incorrectamente un número considerable de frases no críticas como *check-worthy*. En la práctica, esto se traduce en un aumento de los falsos positivos, lo que podría significar una carga adicional de trabajo para los profesionales, quienes tendrán que discernir entre las frases genuinamente *check-worthy* y aquellas clasificadas incorrectamente.

Por otra parte, los resultados de *Naïve Bayes* mostraron una *precision* considerablemente alta, pero a expensas de un *recall* bajo, lo que va en detrimento del objetivo principal del trabajo. Por último, *Logistic Regression* mostró un buen balance con un *recall* de 0,24 sólo por detrás de *SVM*, y una *precision* de 0,78 superando a este último, además de buenos resultados en las demás métricas. Su performance general lo convierte en un candidato competitivo, especialmente en contextos donde la carga de trabajo de los verificadores es una consideración importante.

En este contexto, el análisis estadístico permite determinar si las diferencias observadas al comparar los resultados obtenidos son significativas. El test realizado confirmó que las diferencias observadas para la precision y recall de los modelos *SVM* y *Logistic Regression* son estadísticamente significativas.

> *SVM* destacó debido a su *recall* superior, alineado con el objetivo prioritario de identificar el máximo número de frases *check-worthy* posibles. No obstante, la elección de dicho modelo implica aceptar la necesidad de implementar estrategias adicionales para manejar eficientemente los falsos positivos debido a su bajo nivel de *precision*.

B. Representaciones basadas en embeddings

La Tabla IV presenta los resultados obtenidos para las representaciones basadas *embeddings*. En este caso, la combinación de BETO + *Naïve Bayes* alcanzó el mejor *recall* (0,79), acompañado por una *precision* ligeramente menor. Esta combinación mostró una buena performance en todas las métricas, en comparación al resto de los modelos. Sin embargo, es preciso destacar su variabilidad en la performance, evidenciada por los desvíos estándar observados, lo cual refleja la necesidad de no solo considerar la eficacia del modelo, sino también su robustez. En segundo lugar, se ubicó el modelo combinando *Word2vec + Naïve Bayes*, alcanzando un *recall* de 0,67 y una *precision* de 0,39, ofreciendo un balance adecuado de *precision* y *recall*.

El enfoque de *fine-tuning* de BETO destacó al lograr resultados superiores a utilizar la representación preentrenada de BETO con un clasificador tradicional, mostrando una mejor adaptación a la tarea específica de clasificación. Los resultados mostraron un equilibrio favorable entre *precision* (0,64) y *recall* (0,61), lo que indica una adecuada identificación de frases críticas y una cantidad moderada de clasificaciones erróneas. Asimismo, también obtuvo una buena performance en relación con las otras métricas, obteniendo los mejores *F-measure* y *MCC*.

> Aunque el modelo de BETO + *Naïve Bayes* sobresalió al obtener el *recall* más alto, si se tiene en cuenta el equilibrio entre todas las métricas de evaluación y la estabilidad de su rendimiento, el *fine-tuning* de BETO emerge como el mejor modelo. Este modelo ofrece una performance alta no solo en relación con las métricas de la clase positiva, sino también en las métricas ponderadas.

C. Ensambles de modelos

La Tabla V presenta los resultados obtenidos para los ensambles de los diferentes clasificadores tradicionales utilizando la representación de BETO en todas las variantes. Se eligió BETO como representación ya que fue la que obtuvo los mejores resultados con los clasificadores tradicionales. Dada la relevancia crítica de minimizar los falsos negativos, *AdaBoost* se destacó por su *recall* de 0,79, posicionándose como el más prometedor para la tarea definida, incluso a costa de incrementar la carga de trabajo con algunos falsos positivos por su menor *precision*. Su *Weighted precision* alcanza el valor más alto (0,88), resaltando su eficacia para la clasificación de instancias de ambas clases. Esto también se ve reflejado por su valor de *MCC*. Al analizar las otras técnicas de ensamble, *Voting*, *Bagging*, *Stacking* y *Gradient Boosting*, se observa una diversidad en el rendimiento que resalta la complejidad de la tarea llevada a cabo. Cada uno de estos modelos exhibe un perfil único de *trade-off* entre *precision* y *recall*, reflejando diferencias en cómo equilibran la identificación de las instancias de ambas clases. De los resultados surge que

Modelo	*Precision*	*Weighted Precision*	*Recall*	*F-measure*	*Weighted F-measure*	*Accuracy*	*Balanced Accuracy*	*MCC*
SVM	0,73±0,16	0,85±0,02	0,33±0,09	0,42±0,03	0,83±0,03	0,85±0,05	0,64±0,01	0,41±0,06
Random Forest	0,76±0,19	0,83±0,03	0,13±0,07	0,20±0,06	0,79±0,01	0,84±0,02	0,55±0,01	0,25±0,06
Naïve Bayes	0,83±0,17	0,85±0,02	0,14±0,05	0,23±0,03	0,80±0,01	0,84±0,02	0,56±0,01	0,29±0,04
Logistic Regression	0,78±0,17	0,85±0,02	0,24±0,07	0,35±0,03	0,82±0,02	0,85±0,03	0,60±0,01	0,36±0,06

TABLE III: Resultados de evaluación con algoritmo de validación cruzada K-Fold (k=10) para el conjunto de *features* tradicionales (*TF-IDF*).

Features + Modelo	*Precision*	*Weighted Precision*	*Recall*	*F-measure*	*Weighted F-measure*	*Accuracy*	*Balanced Accuracy*	*MCC*
Word2vec + *SVM*	0,72±0,16	0,83±0,02	0,15±0,05	0,24±0,02	0,80±0,01	0,84±0,02	0,56±0,00	0,27±0,03
FastText + *SVM*	0,76±0,15	0,83±0,02	0,13±0,09	0,20±0,07	0,79±0,01	0,84±0,02	0,55±0,02	0,25±0,04
InferSent + *SVM*	0,75±0,16	0,83±0,02	0,14±0,09	0,22±0,07	0,79±0,01	0,84±0,02	0,56±0,02	0,26±0,05
BETO + *SVM*	0,70±0,15	0,86±0,01	0,45±0,09	0,53±0,04	0,85±0,05	0,85±0,06	0,69±0,01	0,48±0,07
Word2vec + *Random Forest*	0,72±0,15	0,83±0,02	0,17±0,03	0,26±0,03	0,80±0,01	0,84±0,01	0,57±0,01	0,29±0,05
FastText + *Random Forest*	0,73±0,16	0,82±0,02	0,05±0,02	0,09±0,04	0,77±0,00	0,84±0,00	0,52±0,01	0,15±0,05
InferSent + *Random Forest*	0,74±0,03	0,05±0,01	0,10±0,03	0,10±0,03	0,77±0,00	0,84±0,00	0,52±0,00	0,16±0,05
BETO + *Random Forest*	0,82±0,16	0,85±0,02	0,19±0,05	0,30±0,04	0,81±0,01	0,85±0,02	0,59±0,01	0,34±0,05
Word2vec + *Naïve Bayes*	0,39±0,05	0,83±0,01	0,67±0,03	0,49±0,04	0,78±0,05	0,76±0,06	0,72±0,03	0,37±0,06
FastText + *Naïve Bayes*	0,37±0,06	0,82±0,01	0,63±0,09	0,46±0,05	0,77±0,04	0,75±0,05	0,70±0,04	0,35±0,07
InferSent + *Naïve Bayes*	0,37±0,05	0,82±0,01	0,63±0,08	0,46±0,05	0,77±0,04	0,75±0,05	0,70±0,04	0,34±0,06
BETO + *Naïve Bayes*	0,41±0,07	0,86±0,00	0,79±0,04	0,53±0,06	0,78±0,08	0,76±0,09	0,77±0,03	0,44±0,07
Word2vec + *Logistic Regression*	0,63±0,11	0,83±0,01	0,26±0,05	0,36±0,02	0,82±0,01	0,84±0,02	0,61±0,01	0,33±0,04
FastText + *Logistic Regression*	0,63±0,12	0,83±0,01	0,26±0,09	0,35±0,04	0,81±0,02	0,84±0,03	0,61±0,02	0,32±0,04
InferSent + *Logistic Regression*	0,63±0,12	0,83±0,01	0,26±0,09	0,35±0,05	0,81±0,02	0,84±0,03	0,61±0,02	0,33±0,04
BETO + *Logistic Regression*	0,70±0,15	0,86±0,02	0,46±0,08	0,53±0,05	0,85±0,05	0,85±0,07	0,70±0,01	0,49±0,08
BETO *fine-tuning*	0,64±0,13	0,87±0,01	0,61±0,09	0,60±0,07	0,86±0,05	0,86±0,07	0,76±0,03	0,53±0,09

TABLE IV: Resultados de evaluación con algoritmo de validación cruzada K-Fold (k=10) para el conjunto de *features* de aprendizaje profundo.

Modelo	*Precision*	*Weighted Precision*	*Recall*	*F-measure*	*Weighted F-measure*	*Accuracy*	*Balanced Accuracy*	*MCC*
Voting	0,75±0,16	0,87±0,02	0,43±0,09	0,52±0,04	0,85±0,04	0,86±0,06	0,69±0,01	0,49±0,07
Bagging	0,70±0,15	0,86±0,01	0,45±0,08	0,52±0,04	0,85±0,04	0,86±0,06	0,69±0,01	0,48±0,07
Stacking	0,73±0,15	0,87±0,02	0,48±0,08	0,55±0,05	0,85±0,05	0,86±0,06	0,71±0,01	0,51±0,08
Boosting (AdaBoost)	0,50±0,10	0,88±0,01	0,79±0,04	0,61±0,08	0,83±0,09	0,81±0,10	0,80±0,04	0,53±0,09
Boosting (GradientBoosting)	0,75±0,15	0,86±0,02	0,37±0,08	0,47±0,04	0,84±0,03	0,85±0,05	0,66±0,01	0,45±0,07

TABLE V: Resultados de evaluación con algoritmo de validación cruzada K-Fold (k=10) con clasificadores ensamblados.

el peor desempeño lo tuvo *Gradient Boosting* con diferencias significativamente desfavorables en términos de las métricas más relevantes.

> Los mejores resultados fueron obtenidos por *AdaBoost*, técnica que alcanzó el mejor balance en la clasificación de ambas clases, lo que refleja un compromiso bien fundamentado entre alcanzar una cobertura exhaustiva de las frases críticas y mantener un nivel aceptable de falsos positivos.

D. Clasificación basada en LLMs

La Tabla VI presenta los resultados obtenidos para los LLMs seleccionados en sus configuraciones *zero-shot* y *few-shot*. Como se observa, los LLMs presentaron resultados muy variables, reflejando así la complejidad de la tarea. Dado que los LLMs se encuentran entrenados en dominios generales, su capacidad de generalización a dominios específicos puede ser limitada. De esta manera, los resultados revelan tanto el potencial como las limitaciones inherentes a estos modelos.

Llama2 obtuvo un *recall* casi perfecto, a expensas de una *precision* muy baja, lo que genera un gran volumen de falsos positivos. En contraste, `Mistral` obtuvo una *precision* alta, a expensas de un *recall* bajo, lo que implica una capacidad limitada para detectar las frases críticas. En lo que respecta a la diferencia entre las estrategias de *zero-shot* y *few-shot*, tanto para `Mistral` como `Gemma`, la inclusión de ejemplos en el prompt causó una reducción del *recall* y un aumento de la *precision*. Esto resulta en clasificaciones mucho menos exhaustivas, pero más precisas. Finalmente, `GPT-3.5` obtuvo un buen desempeño para las distintas métricas, alcanzando un buen balance entre *precision* y *recall*, y el mejor resultado de *MCC*. Es interesante destacar que para este modelo no se observaron diferencias significativas entre las alternativas *zero-shot* y *few-shot*.

> Considerando el balance entre *precision* y *recall* (tanto para la clase positiva como para la ponderada), *GPT-3.5* y *Gemma* resultan los modelos más adecuados, en sus variantes *few-shot* y *zero-shot*, respectivamente.

Modelo	Aprendizaje	*Precision*	*Weighted Precision*	*Recall*	*F-measure*	*Weighted F-measure*	*Accuracy*	*Balanced Accuracy*	*MCC*
GPT-3.5 TurboGPT	*zero-shot*	0,157	0,78	0,70	0,25	0,49	0,42	0,54	0,05
	few-shot	0,153	0,78	0,76	0,25	0,42	0,36	0,53	0,05
Gemma 2B	*zero-shot*	0,13	0,74	0,47	0,20	0,55	0,47	0,47	-0,03
	few-shot	0,20	0,76	0,07	0,11	0,79	0,82	0,51	0,04
Llama 2 7B	*zero-shot*	0,14	0,86	0,99	0,25	0,05	0,15	0,50	0,03
	few-shot	0,14	0,83	0,98	0,25	0,09	0,16	0,51	0,04
Mistral 7B	*zero-shot*	0,27	0,77	0,01	0,03	0,79	0,85	0,50	0,03
	few-shot	0,6	0,82	0,008	0,01	0,79	0,85	0,50	0,05

TABLE VI: Resultados de evaluación para clasificación mediante LLMs.

E. Análisis final

La Figura 2 presenta la comparación entre las mejores alternativas obtenidas para cada uno de los grupos analizados. Como puede observarse, en general, estas alternativas superaron los *baselines* simples presentados en la Tabla II, observándose las mayores diferencias en términos de *recall* y *MCC*. La excepción es una de las alternativas basadas en LLMs, la cual mostró resultados ligeramente inferiores para casi todas las métricas, excepto *MCC*.

En términos de *recall*, los mejores resultados fueron obtenidos por *AdaBoost* (0,798), seguido por `GPT-3.5` *few-shot* (0,768). En cuanto a *precision*, *AdaBoost* se destacó entre los mejores resultados (0,508), mientras que `GPT-3.5` *few-shot* obtuvo uno de resultados más bajos (0,154). Por el contrario, TF-IDF + *SVM* obtuvo los mejores resultados de precisión, a expensas del peor *recall*, es decir, no identificando la gran mayoría de las frases críticas. BETO *fine-tuning*, obtuvo un muy buen balance entre *precision* y *recall*, ocupando el segundo lugar en *precision* (0,641) y el tercero en *recall* (0,612). En contraste, `Gemma` *zero-shot* no obtuvo buenos resultados en la comparación, ubicándose en último lugar en *precision* (0,130) y en penúltimo lugar en *recall* (0,477).

En lo que respecta a las métricas ponderadas, AdaBoost, BETO *fine-tuning* y TF-IDF + *SVM* destacaron en *precision*, con resultados de 0,881, 0,877 y 0,856 respectivamente. Las diferencias entre TF-IDF + *SVM* y *AdaBoost* y BETO *fine-tuning* fueron estadísticamente significativas. Esta misma tendencia de resultados se observó también para *recall*, *F-measure* y *MCC*. A diferencia de los estudios previos de McDonald et al. [16] y Tarannum et al. [17], la evaluación realizada mostró la superioridad de las representaciones de *embeddings* semánticas por sobre las representaciones basadas en léxico (TF-IDF). En ninguno de los casos las diferencias entre BETO *fine-tuning* y *AdaBoost* resultaron estadísticamente significativas, reforzando la buena performance que alcanzaron ambas alternativas.

La evaluación realizada mostró que tanto BETO *fine-tuning* como *AdaBoost* resultaron modelos altamente competitivos para la identificación de frases críticas para verificar. El primero de ellos se muestra como la mejor opción debido a su buen desempeño en términos de *recall* y *precision*, mostrando así una combinación óptima con suma exhaustividad sin descuidar los falsos positivos, acompañados por resultados satisfactorios para el resto de las métricas analizadas. Finalmente, si bien los LLMs resultan un enfoque prometedor para la tarea sin la necesidad de contar con una gran cantidad de datos de entrenamiento, la evaluación mostró que su performance todavía puede ser mejorada.

V. Conclusiones

Este trabajo exploró la identificación de frases críticas para ser verificadas en idioma español, mediante la evaluación de diferentes combinaciones de representaciones, clasificadores y LLMs. De esta forma, se presentó una visión amplia de las herramientas computacionales y fácilmente al alcance en el estado del arte y su aplicabilidad para el *fact-checking*. Los resultados obtenidos mostraron que las técnicas basadas en *embeddings* poseen un gran potencial para mejorar la eficiencia de los procesos de verificación, al lograr priorizar efectivamente las afirmaciones más críticas y relevantes para su revisión. Por su parte, si bien los LLMs resultan un enfoque prometedor (especialmente dada la baja disponibilidad de datos de entrenamiento), la evaluación mostró que su performance todavía puede ser mejorada.

Como limitaciones a abordar en trabajos futuros pueden mencionarse el tamaño limitado de la colección de datos utilizada en la evaluación, y los desafíos asociados con la fiabilidad de los resultados de los LLMs. Además, sería beneficioso explorar la inclusión de características adicionales que capturen las dependencias sintácticas de las frases, la cantidad de entidades mencionadas, la viralidad de los recursos mencionados (por ejemplo, URLs o *hashtags*), e incluso, indicadores de credibilidad de las fuentes. En relación con el uso de LLMs, sería interesante realizar un análisis más completo del efecto del *prompt* en la tarea, y la sensibilidad a los ejemplos elegidos para la estrategia *few-shot*. En este sentido, se podría considerar un escenario de selección dinámica de los ejemplos, explorando arquitecturas como RAG (*Retrieval Augmented Generation*) [29], la cual ha demostrado ser prometedora en la generación de texto y podría mejorar la capacidad de los modelos para evaluar y contextualizar cada caso de clasificación. Finalmente, sería interesante también explorar la combinación de las técnicas de ensamble con LLMs para potenciar ambos enfoques. Asimismo, los LLMs podrían explorarse como complemento de las técnicas tradicionales

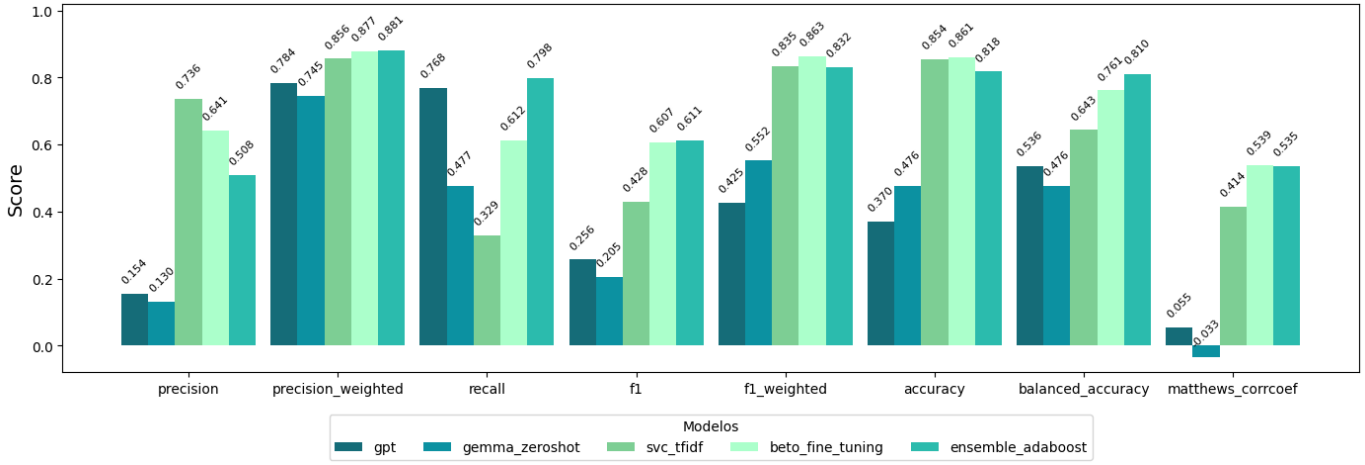

Fig. 2: Resumen comparativo de los resultados más competitivos obtenidos con los modelos más destacados.

para la generación de explicaciones de las clasificaciones de forma de guiar a los verificadores humanos en una mejor selección de las frases a verificar.

REFERENCES

[1] P. Nakov, D. Corney, M. Hasanain, F. Alam, T. Elsayed, A. Barrón-Cedeño, P. Papotti, S. Shaar, and G. D. S. Martino, "Automated fact-checking for assisting human fact-checkers," *arXiv preprint arXiv:2103.07769*, 2021.

[2] S. Singla, "Checking fact worthiness using sentence embeddings," *arXiv preprint arXiv:2012.09263*, 2020.

[3] S. Shaar, G. D. S. Martino, N. Babulkov, and P. Nakov, "That is a known lie: Detecting previously fact-checked claims," *arXiv preprint arXiv:2005.06058*, 2020.

[4] E. Ferrara, "Measuring social spam and the effect of bots on information diffusion in social media," *Complex spreading phenomena in social systems: Influence and contagion in real-world social networks*, pp. 229–255, 2018.

[5] N. Vo and K. Lee, "Where are the facts? searching for fact-checked information to alleviate the spread of fake news," *arXiv preprint arXiv:2010.03159*, 2020.

[6] M. J. Moon, "Can it help government to restore public trust? declining public trust and potential prospects of it in the public sector," in *36th Annual Hawaii International Conference on System Sciences, 2003. Proceedings of the*. IEEE, 2003, pp. 8–pp.

[7] V. S. Zúñiga and M. P. Torres, "Confianza en instituciones políticas: factores que explican la percepción de confianza en chile," *Revista Temas Sociológicos*, no. 25, pp. 231–258, 2019.

[8] J. Pomares and N. Guzmán, "Measuring the impact of fact-checking."

[9] N. Hassan, B. Adair, J. T. Hamilton, C. Li, M. Tremayne, J. Yang, and C. Yu, "The quest to automate fact-checking," in *Proceedings of the 2015 computation+ journalism symposium*. Citeseer, 2015.

[10] A. Patwari, D. Goldwasser, and S. Bagchi, "Tathya: A multi-classifier system for detecting check-worthy statements in political debates," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, 2017, pp. 2259–2262.

[11] S. Shaar, M. Hasanain, B. Hamdan, Z. S. Ali, F. Haouari, A. Nikolov, M. Kutlu, Y. S. Kartal, F. Alam, G. Da San Martino *et al.*, "Overview of the clef-2021 checkthat! lab task 1 on check-worthiness estimation in tweets and political debates." in *CLEF (working notes)*, 2021, pp. 369–392.

[12] F. Arslan, N. Hassan, C. Li, and M. Tremayne, "A benchmark dataset of check-worthy factual claims," in *Proceedings of the International AAAI Conference on Web and Social Media*, vol. 14, 2020, pp. 821–829.

[13] A. Merpert, M. Furman, M. V. Anauti, and L. Zommer, "Chequeando el discurso," 2015.

[14] N. Kotonya and F. Toni, "Explainable automated fact-checking: A survey," *arXiv preprint arXiv:2011.03870*, 2020.

[15] N. Hassan, F. Arslan, C. Li, and M. Tremayne, "Toward automated fact-checking: Detecting check-worthy factual claims by claimbuster," in *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, 2017, pp. 1803–1812.

[16] T. McDonald, Z. Dong, Y. Zhang, R. Hampson, J. Young, Q. Cao, J. L. Leidner, and M. Stevenson, "The university of sheffield at checkthat! 2020: Claim identification and verification on twitter." in *CLEF (Working Notes)*, 2020.

[17] P. Tarannum, F. Alam, M. A. Hasan, and S. R. H. Noori, "Z-index at checkthat! lab 2022: Check-worthiness identification on tweet text," *arXiv preprint arXiv:2207.07308*, 2022.

[18] A. Nikolov, G. D. S. Martino, I. Koychev, and P. Nakov, "Team alex at clef checkthat! 2020: Identifying check-worthy tweets with transformer models," *arXiv preprint arXiv:2009.02931*, 2020.

[19] X. Zhou, B. Wu, and P. Fung, "Fight for 4230 at checkthat! 2021: Domain-specific preprocessing and pretrained model for ranking claims by check-worthiness." in *CLEF (Working Notes)*, 2021, pp. 681–692.

[20] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[21] W. Yu, D. Iter, S. Wang, Y. Xu, M. Ju, S. Sanyal, C. Zhu, M. Zeng, and M. Jiang, "Generate rather than retrieve: Large language models are strong context generators," *arXiv preprint arXiv:2209.10063*, 2022.

[22] S. Agresti, S. A. Hashemian, and M. J. Carman, "Polimi-flatearthers at checkthat!-2022: Gpt-3 applied to claim detection." in *CLEF (Working Notes)*, 2022, pp. 422–427.

[23] P. Nakov, A. Barrón-Cedeño, G. Da San Martino, F. Alam, R. Míguez, T. Caselli, M. Kutlu, W. Zaghouani, C. Li, S. Shaar *et al.*, "Overview of the clef-2022 checkthat! lab task 1 on identifying relevant claims in tweets," in *2022 Conference and Labs of the Evaluation Forum, CLEF 2022*. CEUR Workshop Proceedings (CEUR-WS. org), 2022, pp. 368–392.

[24] F. Alam, A. Barrón-Cedeño, G. S. Cheema, S. Hakimov, M. Hasanain, C. Li, R. Míguez, H. Mubarak, G. K. Shahi, W. Zaghouani *et al.*, "Overview of the clef-2023 checkthat! lab task 1 on check-worthiness in multimodal and multigenre content," *Working Notes of CLEF*, 2023.

[25] D. Opitz and R. Maclin, "Popular ensemble methods: An empirical study," *Journal of artificial intelligence research*, vol. 11, pp. 169–198, 1999.

[26] J. Zamfirescu-Pereira, R. Y. Wong, B. Hartmann, and Q. Yang, "Why johnny can't prompt: how non-ai experts try (and fail) to design llm prompts," in *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, 2023, pp. 1–21.

[27] A. Deshpande, V. Murahari, T. Rajpurohit, A. Kalyan, and K. Narasimhan, "Toxicity in chatgpt: Analyzing persona-assigned language models," *arXiv preprint arXiv:2304.05335*, 2023.

[28] D. Chicco and G. Jurman, "The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation," *BMC genomics*, vol. 21, no. 1, pp. 1–13, 2020.

[29] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel *et al.*, "Retrieval-augmented generation for knowledge-intensive nlp tasks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 9459–9474, 2020.