Simplificación léxica en español con aproximaciones basadas en modelos de IA: experimentos y resultados

Spanish lexical simplification based on AI models: experiments and results

Juan Sixto, Ana Garcia-Serrano
ETSI Informática-UNED
jsixto10@alumno.uned.es, agarcia@lsi.uned.es

Resumen: En este artículo se aborda el problema de la simplificación automática de textos en español, con el propósito de transformar documentos de texto en versiones que faciliten su comprensión a diferentes tipos de usuarios. La investigación se desarrolla afrontando la tarea de generación de sustitutos y su clasificación a través de herramientas recientes en el ámbito del aprendizaje automático profundo o modelos de Inteligencia Artificial (IA). La propuesta se realiza en el marco de la tarea TSAR2022, diseñando e implementando varias aproximaciones con modelos GPT disponibles y explorando nuevas opciones de parametrización e ingeniería de instrucciones. Finalmente, se concluye con un análisis de los resultados obtenidos de las diversas aproximaciones en español, cuyos resultados superan a los presentados.
Palabras clave: Simplificación léxica multilingüe, GPT, simplificación de texto.

Abstract: This paper addresses the problem of automatic simplification of Spanish texts, with the goal of transforming text documents into user-friendly versions that facilitate their comprehension by different types of users. Our research face the task of generating substitutes and their classification through recent tools in the field of deep learning and artificial intelligence (AI) models. Our proposal is conducted within the TSAR2022 task, designing and implementing several approaches with models available and exploring new options for parameterization and instruction engineering. Finally, we conclude with an analysis of the results obtained from the various approaches in Spanish. The results exceed the previous ones.
Keywords: Spanish Lexical Simplification, GPT, Text Simplification.

1 Introducción

La tarea de simplificación automática de textos consiste, en términos generales, en transformar documentos de texto en otros más sencillos para facilitar su comprensión por diferentes usuarios finales con distintas dificultades para la lectura, como la afasia (Carroll et al., 1998) o la dislexia (Rello et al., 2013). Existen diferentes aproximaciones técnicas a la simplificación automática de textos, debido tanto a la subjetividad implícita en el concepto de simplificación, como a la falta de consenso respecto a varias áreas del mismo (Grabar y Saggion, 2022). No obstante, dentro de la tarea hay varios procesos comunes a la mayoría de aproximaciones lo que permite comparar y mejorar en el desempeño general que las aplicaciones.

La tarea de simplificación a nivel léxico, de acuerdo con la definición de Shardlow (Shardlow, 2014) consta de cinco subtareas que pueden afrontarse por separado: (1) identificación de palabras complejas (*Complex Word Identification* (CWI)); (2) generación de posibles sustitutos (*Substitute Generation* (SG)); (3) selección de sustitutos que se ajusten al contexto y preserven el significado original (*Substitute Selection* (SS)); (4) clasificación de sustitutos (*Substitute Ranking* (SR)) para decidir entre los generados y (5) adaptación de las formas simplificadas de las palabras al contexto de la frase *Morphological generation and context adaptation*.

Trabajos previos han utilizado modelos de incrustación de palabras (*word-embeddings*), que necesitan un corpus de texto sin anotar para ser entrenados, dejando atrás el uso de la mayoría de recursos que se usaban en las secuencias de procesos habituales en aplicaciones en las que intervie-

ne el Procesamiento del Lenguaje Natural (PLN) (*pipeline*) (Paetzold y Specia, 2017) (Martínez-Fernández et al., 2006) (Calle-Gómez, García-Serrano, y Martínez, 2006). Estos modelos no solo mejoraban el rendimiento de los modelos anteriores, sino que su implementación era más sencilla al no depender de recursos externos. En los últimos años, el uso de LLMs (*Large Language Models* en inglés) pre-entrenados ha dado lugar a los Modelos de Lenguaje Enmascarados (MLMs, de sus siglas en inglés) para la generación de sustitutos como en TSAR-2022 (*Text Simplification, Accessibility, and Readability*) (Saggion et al., 2015).

Por otro lado, el aprendizaje mediante instrucciones o *prompts* (*Prompt learning*) ha surgido con fuerza para simplificación léxica y aporta en este momento (principios del 2024) los mejores resultados (Vásquez-Rodríguez et al., 2022). Consiste en introducir en un LLM una serie de entradas de texto con la descripción de la tarea específica para obtener los resultados devueltos por el modelo. Esto provoca que los *prompts* utilizados sean una de las claves en el rendimiento del modelo y uno de los campos donde más se puede experimentar. Por ello, en la mayoría de artículos sobre estos modelos, los autores experimentan con varios *prompts*, variando su sintaxis y su léxico para seleccionar aquellas variantes con mejor rendimiento.

En esta investigación la simplificación léxica para textos en español parte del uso de modelos GPT a partir de los trabajos de Aumiller y Gertz (Aumiller y Gertz, 2023), que obtienen los mejores resultados para el español en el marco de la tarea TSAR, incorpora los últimos modelos disponibles y explora nuevas opciones de parametrización y *prompt engineering*. Los resultados muestran que la aproximación final obtenida supera los resultados, en español, de aproximaciones anteriores.

A lo largo de la sección 2 se exponen los conceptos básicos vinculados con la tarea de simplificación automática de textos. La sección 3 detalla el caso de estudio, centrado en la generación de sustitutos de palabras complejas en español para múltiples candidatos de acuerdo con el enunciado de la tarea compartida TSAR-2022. La sección 4 incluye los experimentos realizados, se detalla la solución técnica realizada y las diferentes aproximaciones estudiadas. Además, se presentan los resultados obtenidos durante los mismos y se extraen conclusiones sobre los resultados, las contribuciones de la experimentación realizada y, finalmente se realizan algunas propuestas de trabajo futuro.

2 Contextualización

En los primeros tiempos de la simplificación léxica, la tarea se afrontaba mediante el uso de sinónimos que se decidían a partir de la frecuencia de palabras, asumiendo que a mayor uso de una palabra, más comprensible era esta (Devlin, 1998). Estos primeros modelos empezaron a combinarse con aproximaciones centradas en alteraciones sintácticas para tratar de simplificar la estructura de las frases sin perder información. No contaban con un proceso de identificación de palabras complejas (CWI), sino que consideraban la simplificación de todas las palabras. No sería hasta más tarde, cuando comenzarían a utilizarse clasificadores para seleccionar qué palabras simplificar, evitando en gran medida la pérdida de información, por ejemplo mediante el uso de límites (*thresholds*) sobre las frecuencias de las palabras (Shardlow, 2013).

Estos modelos evolucionaron hasta los que trataban de afinar el proceso de identificación de palabras complejas a partir del uso de lexicones de complejidad. En su mayoría, estos lexicones se construían manualmente. Estos modelos, así como otros basados en reglas o frecuencias estadísticas, fueron la norma hasta la llegada de las técnicas de aprendizaje automático. Estos funcionaban fundamentalmente en base a la idea de entrenar clasificadores binarios que discernían entre las palabras simples y complejas, aunque también se empleaban técnicas de regresión para entrenar modelos que medían el grado de complejidad de las palabras, de una forma más aproximada a los límites anteriores. Los algoritmos más frecuentes en estos modelos eran los árboles de decisiones, los modelos supervisados basados en vectores (SVM, de sus siglas en inglés) y algo después las redes neurales, en muchas ocasiones combinadas con recursos léxicos o morfológicos (Alarcon, Moreno, y Martínez, 2021).

Desde entonces se utilizan modelos de incrustación de palabras (*word-embeddings*), que solo necesitan un corpus de texto sin anotar para ser entrenados, dejando atrás el uso de la mayoría de recursos que se usaban previamente, simplificando la implemen-

tación de aplicaciones. En estos modelos, cada palabra del vocabulario del corpus está representada por un vector único que contiene un número n de valores reales que describen la palabra en un espacio de características semánticas distribuido (Paetzold y Specia, 2017). Son capaces de detectar palabras sinónimas a otras seleccionando aquellas cuyo vector tenga una mayor similitud con el vector de la palabra compleja con operaciones como la (*similitud coseno*) ((Lastra-Díaz, Lara-Clares, y Garcia-Serrano, 2022)).

Los modelos basados en incrustación de palabras mejoraban el rendimiento de los modelos anteriores y como parte de su evolución, estos empezaron a combinarse con modelos generados por los LLM o por conjuntos de puntuaciones de predicción (*prediction scores*) generados previamente a través de los modelos LLM. Así, se utilizan modelos desarrollados con anterioridad, como Word2Vec, Sense2Vec o FastText en combinación con LLMs preentrenados. Sin embargo, estas aproximaciones no superaban en rendimiento a los enfoques más tradicionales preexistentes cuando estos se aplicaban directamente sobre léxicos para generar las palabras sustitutas de una compleja (North et al., 2023).

En este campo, se han incorporado recientemente avances en *deep learning*, y en particular los (*LLMs*) pre-entrenados dando lugar a los Modelos de Lenguaje Enmascarados (MLMs, de sus siglas en inglés) para la generación de sustitutos en la simplificación léxica. Para el español, (Qiang et al., 2020) desarrollan por primera vez un MLM para la tarea en español a través del lenguaje de modelo BERT (*Bidirectional Encoder Representations from Transformers*). Este modelo será llamado BERT-LS por los autores, y se usará en la mayoría de los sistemas desarrollados para esta tarea, y en especial dentro del ámbito del TSAR-2022 (Saggion et al., 2015). BERT-LS, se entrena a través de un proceso que enmascara aleatoriamente un porcentaje de los tokens de entrada, para predecir la palabra enmascarada a partir de su contexto. De esta forma, si se enmascara la palabra compleja en una frase, este MLM actúa como generador de candidatos de la palabra compleja para la simplificación léxica. Otros trabajos para el español son los de (Navarrate-Parra, Uc-Cetina, y Reyes-Magaña, 2023), (de la Rosa et al., 2023), (De la Rosa et al., 2022), (Serrano et al., 2022) o (Gutiérrez-Fandiño et al., 2022).

Por último, el aprendizaje a través de *prompts* (North et al., 2023) ha surgido con fuerza en las aproximaciones más recientes de simplificación léxica. De hecho, aportan los mejores resultados para las tareas de sustitución de palabras en español. Este tipo de aprendizaje consiste en introducir en un LLM una serie de entradas de texto (*prompts*)con la descripción de la tarea específica y obtener los resultados devueltos por el modelo. Esto provoca que los *prompts* utilizados sean una de las claves en el rendimiento de un modelo y s donde más se puede experimentar para desarrollar aproximaciones más precisas. Estos *prompts*, también llamados plantillas en ocasiones porque se aplican sobre los diferentes objetos de un conjunto de datos, pueden combinarse con distintos parámetros y configuraciones en los casos en los que el LLM lo permite. Entre las propuestas de modelos de aprendizaje mediante *prompts* para español destacan PromptLS (Vásquez-Rodríguez et al., 2022), generado a partir del dataset EASIER corpus (Alarcon, Moreno, y Martínez, 2023), y la aproximación UniHD (Aumiller y Gertz, 2023) basada en GPT-3 de OpenAI.

Dentro del ámbito de la simplificación léxica, estos modelos basados en aprendizaje profundo trabajan de forma asimétrica respecto a las diferentes subtareas existentes. Normalmente tienen una etapa de selección de sustitutos (SS) mínima, que se realiza simultáneamente durante la propia generación de los sustitutos (SG). Esto se debe fundamentalmente a que las predicciones de un LLM preentrenado ya incluyen un filtro de palabras que normalmente serían descartadas en pasos posteriores. De la misma forma, no suelen implemnetarse las técnicas para ordenar las listas de candidatos (SR), utilizando las referencias del propio LLM. También el el caso de las aproximaciones MLM, u otros modelos, alteran la importancia entre las diferentes subtareas o sustituyen a varias de ellas por un único paso.

En comparación con los modelos previos existentes, el uso de instrucciones (*prompting*) es más sencillo, ya que no introduce grandes cantidades de parámetros adicionales ni requiere la inspección directa de las representaciones de un modelo. Es lógico pensar que los modelos de aprendizaje mediante *prompts* y otros MLM serán aún más utiliza-

dos en el futuro inmediato ya que aún cuentan con un amplio margen de mejora. Sin embargo, esta técnica se basa en instrucciones creadas manualmente a partir de la intuición del experimentador. Y estas instrucciones manuales pueden no ser óptimas, debido a que los LLM pueden haber aprendido de contextos sustancialmente diferentes durante su creación (aprendizaje) respecto al momento actual (Jiang et al., 2020).

3 Descripción de la tarea TSAR

Para probar sistemas y comparar los resultados con nuevas aproximaciones en problemas concretos (denominados tareas) se organizan las denominadas "tareas compartidas", como en los foros CLEF, IberLEF (para el español) u otros (Moreno-Sandoval, Gisbert, y Montoro, 2020), (Ermakova et al., 2022).

En la tarea compartida TSAR-2022 los participantes deben crear un sistema de simplificación léxica que devuelva una lista ordenada (sin empates) de un máximo de 10 posibles palabras candidatas a sustituir cada palabra compleja objetivo (ver Tabla 1) para que mejore la comprensión de la frase.

Con la excepción de algunas diferencias a la hora de generar el conjunto de datos, la tarea funciona de la misma forma para los tres idiomas (inglés, portugués (brasileño) y español). A diferencia de otras tareas similares, no se proporciona a los participantes un conjunto de entrenamiento, pero sí varios ejemplos de prueba en cada lengua, pues se espera que los participantes hagan uso de recursos externos para crear sus sistemas, en lugar de entrenar modelos.

Frase/Contexto: Floreció en la época clásica y tenía una reputada escuela de filosofía.
Palabra objetivo: reputada
Sustitutos ordenados (gold): prestigiosa:6, famosa:4, reconocida:2, afamada:2, conocida:2, renombrada:2, respetada:2, prestigioso:1, muy reconocida:1, valorada:1, acreditada:1, prestigiada:1

Tabla 1: Ejemplo de simplificación léxica extraído del conjunto de datos TSAR

Durante el proceso de creación de los conjuntos de datos en inglés y español, las frases/contextos y las palabras (complejas) objetivo se seleccionaron a partir de conjuntos de datos anteriores, concretamente los presentados en la tarea compartida BEA-2018 (Yimam et al., 2018) sobre identificación de palabras complejas (CWI). Ante cada una de las frases y palabra objetivo, el sistema debe generar un conjunto de sinónimos candidatos, donde no se permiten palabras complejas. Además, estos candidatos deben tener la misma inflexión morfológica que la palabra compleja en la frase original, o se considerarán como erróneos.

En la tarea compartida TSAR-2022 (Saggion et al., 2023). las métricas utilizadas permiten una comparación más justa de los sistemas que proponen un número diferente de candidatos para la sustitución. Las métricas son las siguientes: ACC@1 (Exactitud), potential@k, accuracy@n@top1 y MAP@k, donde k $\in$ 3, 5, 10 y n $\in$ 1, 2, 3.

El **Potencial**@k se define como el porcentaje de casos en los que al menos uno de los **k** sustitutos mejor clasificados también está presente en los datos de referencia.

La **Exactitud**@k@top1 se define como el porcentaje de casos en los que al menos uno de los **k** sustitutos mejor clasificados coincide con el sinónimo sugerido con más frecuencia en los datos de referencia. Es importante señalar que Exactitud@1@top1 se denominó Exactitud@1 en (Stajner et al., 2022).

La métrica **MAP** en el contexto de la simplificación léxica, indica que en una lista ordenada de sustitutos generados, estos pueden coincidir (relevantes) o no (irrelevantes) con el conjunto de los sustitutos *gold-standard*. A diferencia de Precision@k, que sólo mide qué porcentaje de los k sustitutos mejor clasificados puede encontrarse entre los sustitutos *gold-standard*, **MAP@k** tiene en cuenta además la posición de los sustitutos relevantes entre los **k** primeros candidatos generados (es decir, si los candidatos relevantes están o no en las primeras posiciones).

4 Propuesta, experimentación y evaluación

De todas las aproximaciones realizadas en TSAR-2022, la publicada por Aumiller y Gertz (2023) es la que obtiene los mejores resultados tanto en inglés como en español, y la técnica propuesta es la mas original y sencila con respecto al resto. Consiste en un sistema basado en instrucciones a través de un modelo GPT parametrizado, combinado con un conjunto de seis plantillas, con distintos niveles de contexto, mediante las que generar los *prompt* en inglés para GPT-3.

Dada la complejidad en los diferentes contextos de las palabras a sustituir y con la intención de no sesgar las predicciones del sistema, los autores optaron por una primera aproximación que se basa en una única consulta de *disparo cero*. En este caso, al modelo GPT de OpenAI se le proporciona únicamente la frase contextual y la palabra compleja, preguntando al modelo por diez sinónimos simplificados de la palabra compleja en el contexto dado. Como en esta aproximación no se proporciona ningún conocimiento adicional al modelo, se considera un límite inferior razonable para la configuración de la tarea.

Existe una gran variabilidad en las generaciones de las respuestas por parte del modelo al cambiar la plantilla o la configuración del contexto. Para tratar de moderar este efecto, realizan una segunda aproximación con predicciones conjuntas, utilizando un conjunto de varias plantillas diferentes para ampliar el espectro de posibles generaciones y garantizar que un número mínimo de sugerencias supere el proceso de filtrado. Por último, también se varía la temperatura de generación (creatividad en la generación), ya que un valor más alto aumenta la probabilidad de una predicción más creativa, para tener una lista de respuestas más diversas. Los seis *prompts* utilizados por Aumiller y Gertz están descritos en su publicación.

Pese a que su trabajo fundamental es con el conjunto de datos en inglés, aplican sus experimentos al español y portugués trasladando las instrucciones al español o al portugués de forma directa, cambiando la pregunta del *prompt* por: "*Given the above context, list ten alternative* ***Spanish*** *words for 'complex_word' that are easier to understand.*".

4.1 Parámetros GPT

La API de OpenAI se basa en un conjunto de modelos, cada uno desarrollado con un ajuste específico de los parámetros, que se describen a continuación, y con los que se han experimentado durante la presente investigación.

El parámetro *max_tokens* controla el número máximo de tokens a generar en las respuestas del modelo. Durante esta investigación hemos mantenido el valor en 256 para asegurar suficiente espacio para las salidas generadas, aunque en la práctica, la mayoría de las terminaciones están muy por debajo de este límite, siguiendo lo establecido en Aumiller y Gertz (2023).

Los parámetros *frequency and presence penalties* pueden tener valores comprendidos entre -2,0 y 2,0. En el caso del *penalizador de frecuencia*, los valores positivos penalizan los tokens nuevos en función de si aparecen en el texto hasta el momento, lo que aumenta la probabilidad de que el modelo tienda a abordar temas nuevos en lugar de repetirse. De forma parecida, los valores positivos del *penalizador de presencia* penalizan los nuevos tokens en función de su frecuencia en el texto hasta el momento, lo que disminuye la probabilidad de que el modelo repita textualmente la misma línea. Estos se utilizan para reducir la probabilidad de secuencias repetitivas de tokens, y funcionan modificando directamente los *logits* (probabilidades logarítmicas no normalizadas) mediante una contribución aditiva. En (Aumiller y Gertz, 2023) losautores establecen unos valores de *frequency penalty=0,5*, así como *presence penalty=0,3*, con el objetivo de penalizar conjuntamente los tokens presentes y las repeticiones de tokens. Indican también que estos valores están muy por debajo del máximo, ya que los tokens de sub-palabras individuales pueden estar presentes varias veces en múltiples predicciones y ser válidas. Durante el desarrollo de este trabajo esos valores se han mantenido estables.

En los modelos ChatGPT, GPT-3 y GPT-4 el parámetro *temperature* rige la aleatoriedad del sistema y, por tanto, la creatividad de sus respuestas. Se trata de un valor entre 0.0 y 1.0 en donde los valores más altos hacen que la salida sea más aleatoria, mientras que los valores más bajos generan salidas más centradas y deterministas. En este trabajo se han establecido tres valores para este parámetro, 0.3 (bajo) , 0.5 (medio) y 0.7 (alto) para experimentar e intentar establecer su influencia sobre el rendimiento de los modelos.

Los resultados obtenidos en la tarea TSAR-2022 evidenciaron las posibilidades de los nuevos modelos de instrucciones para afrontar la simplificación a partir de palabras complejas previamente etiquetadas, sin embargo, el uso de instrucciones generadas manualmente para las tareas automáticas es un campo reciente y donde muy a menudo se publican nuevas técnicas y métodos para la generación de instrucciones (*prompt tuning*) (Lester, Al-Rfou, y Constant, 2021), que pueden mejorar el rendimiento de estos mode-

los. Además, se ha observado que las instrucciones manuales adolecen frecuentemente de un alto grado de inestabilidad (Liu et al., 2023), aunque este problema puede mitigarse a través de varias técnicas. También se echa en falta un análisis más detallado de la influencia de algunos parámetros en el rendimiento final de las tareas, así como las diferencias existentes a la hora de generar las instrucciones en un idioma o en otro. Por último, dada la velocidad a la que se publican nuevos modelos de lenguaje, ya existen nuevas variantes de los modelos GPT cuyo rendimiento es necesario evaluar. Por ello se ha planteado la experimentación siguiente.

4.2 Implementación

Se ha desarrollado un *pipeline* de varios procesos encadenados que abarcan todo el proceso, desde la generación inicial de las instrucciones hasta la evaluación de los resultados. Todos los procesos han sido implementados expresamente para este trabajo utilizando el lenguaje de programación Python 3, con la excepción del evaluador TSAR, que está desarrollado por los autores de la tarea y cuyo contenido no se ha modificado. Los procesos son los siguientes:

Generador de instrucciones: Este proceso transforma todas las frases en *prompts* para poder enviarse al modelo GPT.

Conector a GPT: Este programa hace uso de la API de OpenAI para acceder a los modelos GPT. Configura los parámetros del modelo a partir de un fichero externo, y todos los *prompts* generados en el paso anterior, recogiendo sus respuestas en un documento.

Post procesador de respuestas: Una vez obtenidas las respuestas del modelo GPT, es necesario un proceso de adaptación al formato definitivo para la tarea de evaluación.

Evaluador: Utiliza dos ficheros de entrada, el documento de respuestas correctas (*gold standard*) y otro documento de similar formato, pero con las respuestas del modelo a evaluar. A partir de ellos, genera los resultados de todas las métricas de la tarea.

4.3 Post-procesado de las respuestas

Uno de los problemas de los modelos basados en instrucciones es que devuelven sus respuestas de forma libre, con formatos variables. Coincidiendo con otros trabajos similares (Aumiller y Gertz, 2023), durante este trabajo se ha comprobado que las salidas del modelo GPT no cumplen un patrón estable, incluso aunque se especifique en las instrucciones. Ante este problema de respuestas no deterministas, es necesario este post-procesado para no penalizar su rendimiento.

A continuación, se detallan los filtros que se han implementado durante esta etapa, así como una explicación de los errores que afrontan.

- **Numeración de listas:** La mayoría de las respuestas del modelo GPT son en formato de lista numerada. Por ejemplo "*1. adaptarse 2. sumarse 3. incorporarse 4. obedecer 5. ...*". Esto obliga a eliminar la numeración, así como a asegurar que todas las palabras se sitúan en la misma línea del documento. Sin embargo, también hay que tener en cuenta que la salida no siempre es así, por lo que primero es necesario identificar si esto ocurre.
- **Signos de puntuación:** Fundamentalmente comas (,), puntos y comas (;) y dos puntos (:) que deben eliminarse del texto.
- **Mayúsculas:** Todas las palabras del conjunto de respuestas de la tarea (*gold-standard*) están en minúsculas, así que es necesario asegurarse de que las respuestas mantienen ese formato.
- **Explicaciones entre paréntesis:** Se ha detectado que en algunas ocasiones, el modelo incluye en su respuesta explicaciones entre paréntesis, formadas por la traducción en inglés. Este contenido adicional no debe ser tenido en cuenta a la hora de generar respuestas válidas para la evaluación.
- **Espacios:** Las palabras candidatas deben separarse mediante tabulaciones, no espacios sencillos, así que también hay que asegurarse de adaptarlo.

4.4 Resultados

Los experimentos se han clasificado en secciones, que enfrentan distintas estrategias de transferencia de conocimiento (*transfer learning)*, como el tipo de contexto utilizado o el lenguaje de las instrucciones.

En la fase experimental de este trabajo se utiliza el enfoque de transferencia de tareas (*task transfer*, en inglés), donde al modelo se

le proporcionan ejemplos de la tarea en el contexto (Brown et al., 2020; Radford et al., 2019). El uso de estos ejemplos de entrenamiento se describen habitualmente como *one-shot*, en el caso de utilizar un único ejemplo, o *few-shot* cuando se utilizan varios ejemplos. Estos casos se comparan con la aproximación denominada *zero-shot*, que únicamente utiliza una descripción o invocación de la tarea en lenguaje natural, sin utilizar ejemplos.

Los modelos GPT suelen mejorar su rendimiento con el uso de ejemplos respecto a la falta de ellos (zero-shot), lo que sugiere que estos son meta-aprendices en los que el aprendizaje intrínseco a los LLMs se combina con el aprendizaje rápido *en contexto* a través de las instrucciones (Brown et al., 2020).

Los experimentos se han dividido en diferentes secciones que afrontan distintos aspectos, y junto con una explicación de los mismos, cada sección incluye una tabla resumen con los resultados obtenidos, a los que se le ha incluido con fines comparativos los resultados de (Aumiller y Gertz, 2023), por ser el modelo de mejor rendimiento de la tarea TSAR-2022, y del modelo *LSBert-baseline*, utilizado como punto de referencia por los autores de la tarea (Saggion et al., 2023).

4.4.1 Text-DaVinci-003 y GPT-3.5-turbo

Como primera aproximación, se han comparado dos recientes modelos de lenguaje GPT-3, para conocer si existe una diferencia relevante entre sus rendimientos, de cara a realizar el resto de los experimentos (ver Tabla 2). Los dos modelos escogidos son los siguientes;

text-davinci-003: La última versión del modelo text-davinci-002, utilizado por (Aumiller y Gertz, 2023) para la tarea. Se ha entrenado en una amplia gama de tareas y tiene una mayor capacidad de aprendizaje a partir de menos ejemplos en comparación con otros modelos.

gpt-3.5-turbo: El modelo GPT-3.5 más reciente. Optimizado para el uso conversacional, también es más preciso en tareas de clasificación y en aprendizaje con contexto. Además, sus respuestas suelen ser más extensas. Se ha incluido también una aproximación de su versión 16k, que tiene las mismas capacidades que el modelo gpt-3.5-turbo estándar pero ha sido entrenado con 4 veces más contexto.

Para esta comparación se ha utilizado una aproximación sencilla de zero-shot en inglés para los tres modelos, variando también los valores de temperatura para obtener un espectro más amplio de resultados. Al evaluar el modelo da-vinci-003 se obtienen unos valores de ACC@1 entre el 61 % y el 67 %, muy similares a los resultados que obtiene UniHD con la versión da-vinci-002. En cambio, al evaluar el modelo gpt-3.5-turbo, obtenemos valores de ACC@1 entre el 68 % y el 72 %, muy superiores a da-vinci y al estado del arte actual. También obtiene unos resultados superiores en MAP@k y en Potencial@k, aunque resulta significativo que la aproximación de UniHD obtiene mejores resultados para algunas secciones de Potencial@k. Respecto a la variable temperatura, ambos modelos obtienen sus mejores resultados de ACC@1 con valores bajos (0.3), aunque en el resto de métricas resultan más dispares. Por su parte, la versión de 16k obtiene peores resultados que su versión estándar. Podemos concluir que el modelo gpt-3.5-turbo tiene un mejor rendimiento que el resto de versiones, [1], por tratarse del último modelo publicado. En experimentos posteriores se ha utilizado gpt-3.5-turbo en combinación con otras técnicas de *prompting* para mejorar su rendimiento.

4.4.2 Idioma de los prompts

Los modelos GPT son multilingües y capaces de gestionar varios idiomas a la vez, pudiendo recibir las instrucciones en un idioma u otro. Esta es una de sus capacidades más útiles, pero el funcionamiento exacto y cómo influye a su rendimiento específico no se ha evaluado aún. Por esta razón, se ha considerado realizar un experimento alterando únicamente el idioma de los *prompts* para extraer conclusiones de su rendimiento. Aunque GPT-3 acostumbra a responder en el mismo idioma de la instrucción recibida, es posible solicitarle que responda en otro idioma. En el experimento se le han solicitado diez sinónimos en español para la palabra clave, aunque la instrucción se ha redactado en los dos idiomas: **Prompt en inglés:** Give me ten simplified Spanish synonyms for the following word: "propiciado" **Prompt en español:** Dame diez sinónimos más sencillos en español para la palabra: "propiciado"

Los resultados obtenidos en esta prueba se presentan en la tabla 3. Lo primero que podemos observar es que, pese a utilizarse *prompts* sin contexto, los resultados en inglés son si-

[1] https://platform.openai.com/docs

Aprox.	Temp.	ACC@1	Acc@k@Top1			MAP@k			Potencial@k		
			k=1	k=2	k=3	k=3	k=5	k=10	k=3	k=5	k=10
UniHD (davinci-002)		0.6521	0.3505	0.5108	**0.5788**	0.4281	0.3239	0.1967	0.8206	**0.8885**	**0.9402**
gpt-3.5-t	0.3	**0.7201**	**0.3831**	0.5135	0.5733	**0.5182**	**0.3843**	**0.2243**	**0.8532**	0.8831	0.9021
gpt-3.5-t	0.5	0.6959	0.3804	0.4891	0.5489	0.5055	0.3719	0.2163	0.8233	0.8586	0.8668
gpt-3.5-t	0.7	0.6820	0.3777	**0.5163**	**0.5788**	0.5045	0.3739	0.2170	0.8315	0.8478	0.8722
gpt-3.5-t-16k	0.3	0.6847	0.3722	0.4782	0.5380	0.4987	0.3660	0.2129	0.8070	0.8423	0.8505
davinci-003	0.3	0.6739	0.3586	0.4375	0.5625	0.4038	0.3005	0.1839	0.8369	0.8858	0.9293
davinci-003	0.5	0.6467	0.3369	0.4619	0.5543	0.4364	0.3275	0.1914	0.8260	0.8722	0.9211
davinci-003	0.7	0.6195	0.3125	0.4266	0.5108	0.4101	0.3033	0.1772	0.7853	0.8315	0.8777

Tabla 2: Resultados de los modelos Text-Da-Vinci-003 y GPT-3.5-turbo.

Aprox.	Temp.	ACC@1	Acc@k@Top1			MAP@k			Potencial@k		
			k=1	k=2	k=3	k=3	k=5	k=10	k=3	k=5	k=10
UniHD		0.6521	0.3505	0.5108	**0.5788**	0.4281	0.3239	0.1967	0.8206	0.8885	**0.9402**
gpt-3.5-t-NC	0.3	**0.7336**	0.3750	**0.5163**	0.5706	**0.5294**	**0.3916**	**0.2294**	0.8641	**0.8967**	0.9130
gpt-3.5-t-NC	0.5	0.7201	0.3641	0.4972	0.5543	0.5058	0.3743	0.2153	0.8641	**0.8967**	0.9076
gpt-3.5-t-NC	0.7	0.7146	**0.3777**	0.5054	0.5760	0.4992	0.3710	0.2160	0.8668	0.8940	0.9157
gpt-3.5-t-NC-ES	0.3	0.7146	0.3641	**0.5163**	0.5543	0.5123	0.3777	0.2212	**0.8695**	0.8913	0.9048
gpt-3.5-t-NC-ES	0.5	0.6902	0.3559	0.5108	0.5543	0.5033	0.3631	0.2136	0.8505	0.8777	0.8994
gpt-3.5-t-NC-ES	0.7	0.6956	0.3451	0.5000	0.5570	0.5025	0.3765	0.2207	0.8559	0.8940	0.9130

Tabla 3: Resultados para las aproximaciones con prompts en inglés y español.

milares a los mostrados en la sección anterior, donde se ha utilizado un zero-shot. Este tema se aborda en más detalle a continuación, en la sección 4.4.3. Respecto al lenguaje utilizado en los *prompts*, se observa en los resultados que la versión en inglés obtiene resultados superiores, con ACC@1 en una franja entre 71 % y 73 %, mientras que la versión en español oscila entre 69 % y 71 %. Aunque no es una diferencia elevada, y puede deberse a la variabilidad del modelo, se puede concluir que el uso de instrucciones en español no mejora el rendimiento del modelo, al menos en lo respectivo a esta tarea.

4.4.3 Aproximaciones Zero-shot

En las aproximaciones de zero-shot, el modelo no recibe ningún ejemplo en las instrucciones, sino que debe generar su respuesta sin información adicional, aunque sí se comparte la frase contextual como parte de la instrucción, junto con la palabra objetivo. Adicionalmente, se le añade una instrucción sencilla en la que se pide al modelo que aporte diez palabras alternativas para la palabra objetivo. Tampoco se incluyen detalles sobre el formato de salida, ni otras especificaciones. Las respuestas del modelo se consideran en orden de clasificación a efectos de la tarea, y no se utiliza ningún otro proceso para reordenarlas.

No obstante, es importante diferenciar esta aproximación zero-shot, de una aproximación sin contexto. En el caso de las instrucciones sin contexto, no solo no se aporta un ejemplo de solución al modelo, sino que tampoco se le incluye el contexto de la frase en la que se encuentra la palabra a sustituir. Los resultados obtenidos en este experimento se presentan en la tabla 4.

4.4.4 Aproximaciones One-shot

De forma similar a las aproximaciones zero-shot, en las aproximaciones one-shot, se solicita al modelo que genere una lista de diez sinónimos de la palabra clave que sean más sencillos. Pero en esta ocasión, al modelo se le aporta un ejemplo de pregunta y respuesta correcta, como parte del *prompt*, previamente a la solicitud para que haga lo mismo con una nueva frase. El ejemplo se ha extraído de las frases de prueba que los autores de la tarea TSAR ofrecen junto al conjunto de datos de evaluación.

Estas aproximaciones mediante ejemplos son extensamente utilizadas en la experimentación con modelos GPT-3, ya que este es capaz de extraer detalles de una tarea a realizar a través de unos pocos ejemplos expresados en lenguaje natural (Brown et al., 2020). Este proceso sustituye a los procesos clásicos de ajuste fino en los modelos previos, que requerían una gran cantidad de datos. Los resultados de esta aproximación se recogen en la tabla 5. Se observa una subida general respecto a las aproximaciones de zero-shot en torno al 3-4 % en todas las métricas, estableciendo además un nuevo máximo de ACC@1 de 77,71 %. De aquí podemos con-

			Acc@k@Top1			MAP@k			Potencial@k		
Aprox.	**Temp.**	**ACC@1**	k=1	k=2	k=3	k=3	k=5	k=10	k=3	k=5	k=10
UniHD		0.6521	0.3505	0.5108	**0.5788**	0.4281	0.3239	0.1967	0.8206	0.8885	**0.9402**
gpt-3.5-t-0s	0.3	0.7201	**0.3831**	0.5135	0.5733	0.5182	0.3843	0.2243	0.8532	0.8831	0.9021
gpt-3.5-t-0s	0.5	0.6959	0.3804	0.4891	0.5489	0.5055	0.3719	0.2163	0.8233	0.8586	0.8668
gpt-3.5-t-0s	0.7	0.6820	0.3777	**0.5163**	**0.5788**	0.5045	0.3739	0.2170	0.8315	0.8478	0.8722
gpt-3.5-t-NC	0.3	**0.7336**	0.3750	**0.5163**	0.5706	**0.5294**	**0.3916**	**0.2294**	0.8641	**0.8967**	0.9130
gpt-3.5-t-NC	0.5	0.7201	0.3641	0.4972	0.5543	0.5058	0.3743	0.2153	0.8641	**0.8967**	0.9076
gpt-3.5-t-NC	0.7	0.7146	0.3777	0.5054	0.5760	0.4992	0.3710	0.2160	**0.8668**	0.8940	0.9157

Tabla 4: Resultados para las aproximaciones Zero-shot.

			Acc@k@Top1			MAP@k			Potencial@k		
Aprox.	**Temp.**	**ACC@1**	k=1	k=2	k=3	k=3	k=5	k=10	k=3	k=5	k=10
UniHD		0.6521	0.3505	0.5108	0.5788	0.4281	0.3239	0.1967	0.8206	0.8885	**0.9402**
gpt-3.5-t-1s	0.3	**0.7771**	**0.4266**	**0.5597**	0.6195	**0.5590**	**0.4187**	**0.2471**	**0.8777**	**0.8994**	0.9048
gpt-3.5-t-1s	0.5	0.7635	0.4184	0.5516	**0.6304**	0.5394	0.4022	0.2371	0.8614	0.8967	0.9048
gpt-3.5-t-1s	0.7	0.7527	0.4239	0.5434	0.6141	0.5357	0.4025	0.2358	0.8722	**0.8994**	0.9157

Tabla 5: Resultados para las aproximaciones One-Shot

cluir que el modelo está transfiriendo conocimiento (*transfer learning*) a partir del ejemplo incluido en el *prompt*.

4.4.5 Aproximaciones Two-shots

La configuración *two-shots*, también llamada *few-shots*, es similar a los *one-shot*, con una única excepción de que recibe dos ejemplos previos extraídos de las frases de prueba. Durante este experimento, se han replicado las aproximaciones *one-shot*, incluyendo un segundo ejemplo extraído del conjunto de datos de prueba de la tarea. Un ejemplo de aprendizaje *two-shots* en nuestro modelo es el siguiente:

Los resultados de este experimento se resumen en la tabla 6. En estos se observan pocas diferencias respecto a los modelos one-shot, quedando incluso ligeramente por debajo en la métrica de ACC@1. Esto puede deberse a que el nuevo ejemplo no es relevante para el modelo, pudiendo ser suficiente con uno solo, y las diferencias deberse únicamente a la variabilidad entre ejecuciones. No obstante, para valores altos de k, sus resultados mejoran los del one-shot, lo que puede revelar alguna mejora secundaria en la que el modelo incluye más palabras correctas, aunque esto requeriría más investigación para poder corroborarlo. Puede estar relacionado también con el hecho de que los mejores resultados se obtengan con un valor alto de temperatura, a diferencia del resto de experimentos, pues esto indica una mayor creatividad en las respuestas del modelo.

5 Discusión final

Tras presentar los distintos experimentos realizados para la simplificación léxica en español mediante modelos de aprendizaje basados en instrucciones, las aproximaciones con un mayor rendimiento se han recogido en la tabla 7, donde se han ordenado según su puntuación de ACC@1, al considerarse la métrica más relevante para la tarea (Saggion et al., 2023).

Estos experimentos ofrecen una imagen panorámica del funcionamiento de los modelos GPT-3 en el desempeño de la tarea, así como el peso específico de las posibles variaciones en los modelos. De estas pruebas podemos concluir que los nuevos modelos lingüisticos incorporados a GPT-3, especialmente el modelo *gpt-3.5-turbo*, mejora sensiblemente el rendimiento de los anteriores. Nuestras aproximaciones llegan a obtener un 77 % de exactitud (ACC@1) por un 65 % de UniHD(Aumiller y Gertz, 2023), a pesar de no incorporar combinaciones de *prompts*. Este resultado implica que el primer sustituto propuesto por el modelo se encuentra en la lista de resultados correctos en el 77 % de los casos, un resultado significativo y prometedor para el desarrollo de posibles aplicaciones de simplificación de textos. También son reseñables las altas puntuaciones que los modelos obtienen en la métrica de potencial, que hace referencia al ratio de candidatos presentes en la lista de resultados correctos. Alcanzando incluso en ocasiones el 90 %, que representa una capacidad elevada de generar los sustitutos correctos por parte del mode-

			Acc@k@Top1			MAP@k			Potencial@k		
Aprox.	**Temp.**	**ACC@1**	k=1	k=2	k=3	k=3	k=5	k=10	k=3	k=5	k=10
UniHD		0.6521	0.3505	0.5108	0.5788	0.4281	0.3239	0.1967	0.8206	0.8885	**0.9402**
gpt-3.5-t-2s	0.3	0.7608	0.4184	0.5489	0.6195	0.5543	**0.4172**	0.2449	0.8722	0.8913	0.9021
gpt-3.5-t-2s	0.5	0.7581	0.4211	**0.5706**	0.6467	0.5455	0.4145	0.2429	0.8804	0.8994	0.9076
gpt-3.5-t-2s	0.7	**0.7717**	**0.4320**	0.5570	**0.6630**	**0.5628**	0.4155	**0.2450**	**0.8940**	**0.9048**	0.9130

Tabla 6: Resultados para las aproximaciones Two-shots.

			Acc@k@Top1			MAP@k			Potencial@k		
Aprox.	**Temp.**	**ACC@1**	k=1	k=2	k=3	k=3	k=5	k=10	k=3	k=5	k=10
gpt-3.5-t-1s	0.3	**0.7771**	0.4266	**0.5597**	0.6195	0.5590	**0.4187**	**0.2471**	0.8777	0.8994	0.9048
gpt-3.5-t-2s	0.7	0.7717	**0.4320**	0.5570	**0.6630**	**0.5628**	0.4155	0.2450	**0.8940**	**0.9048**	**0.9130**
gpt-3.5-t-1s	0.5	0.7635	0.4184	0.5516	0.6304	0.5394	0.4022	0.2371	0.8614	0.8967	0.9048
gpt-3.5-t-2s	0.3	0.7608	0.4184	0.5489	0.6195	0.5543	0.4172	0.2449	0.8722	0.8913	0.9021
gpt-3.5-t-2s	0.5	0.7581	0.4211	0.5706	0.6467	0.5455	0.4145	0.2429	0.8804	0.8994	0.9076
gpt-3.5-t-0s	0.3	0.7201	0.3831	0.5135	0.5733	0.5182	0.3843	0.2243	0.8532	0.8831	0.9021
UniHD		0.6521	0.3505	0.5108	0.5788	0.4281	0.3239	0.1967	0.8206	0.8885	0.9402
LSBert-baseline		0.3262	0.1577	0.2326	0.2860	0.1904	0.1313	0.0775	0.4946	0.5802	0.6737

Tabla 7: Comparativa de resultados más significativos para el TSAR en Español.

lo y deja la puerta abierta a una importante mejora en su rendimiento si se añade un sistema de ordenación (*ranking*) de candidatos que logre diferenciarlos.

Respecto al papel del parámetro *temperatura*, se observa que los valores más bajos obtienen mejores resultados en la mayoría de los experimentos, pudiendo estimar que un valor más estable de la creatividad del modelo provoca más palabras correctas. Sin embargo, la diferencia es reducida, siempre menor al 5 %, así que su relevancia es menor a otros elementos de la configuración. Esta tendencia se invierte en el caso de las aproximaciones two-shots. También resulta significativo el rendimiento similar de las aproximaciones one-shot y two-shots, que obtienen resultados muy similares en todas las métricas. Esto puede deberse a que el modelo apenas infiere información de refuerzo de múltiples contextos, más allá del formato de salida.

Aun así, los resultados obtenidos son susceptibles de mejorar en varias áreas, lo que permite suponer que en el futuro estos modelos lograrán obtener resultados aún más altos. En otras aproximaciones del estado del arte (Chiu, Collins, y Alexander, 2021; Aumiller y Gertz, 2023), la combinación de múltiples *prompts* a través de diversas fórmulas ha generado una mejora significativa en el rendimiento con modelos GPT-3. Las diferencias de resultados en exactitud, MAP y potencial sugieren que los resultados también son susceptibles de mejora cuando se trabaja con listas de varios sustitutos, algo que podría ser útil a la hora de trabajar con entornos de simplificación orientados a usuarios finales con necesidades concretas. Esto es coherente con lo observado durante los experimentos, donde muchas de las respuestas del modelo incluían palabras repetidas, palabras más complejas que el original o combinaciones de dos palabras, para alcanzar los diez sustitutos solicitados. De hecho, algunas aproximaciones (Aumiller y Gertz, 2023) han optado por borrar estas palabras durante el proceso, aunque eso suponga realizar la evaluación con un menor número de palabras, lo que puede explicar sus resultados con valores altos de k, más cercanos a nuestros experimentos.

Además de los problemas con el formato de salida que ya se ha explicado anteriormente, se ha observado otro un error recurrente en los experimentos, consistente en que las palabras candidatas aportadas por el modelo no están conjugadas correctamente respecto a la frase de contexto. Aunque son casos minoritarios, este error invalida la totalidad de las palabras. Una posible solución de futuro sería la inclusión de un etiquetador gramatical que conjugue las palabras erróneas.

Bibliografía

Alarcon, R., L. Moreno, y P. Martínez. 2021. Lexical simplification system to improve web accessibility. *IEEE Access*, 9:58755–58767.

Alarcon, R., L. Moreno, y P. Martínez. 2023. Easier corpus: A lexical simplification resource for people with cognitive impairments. *Plos one*, 18(4):e0283622.

Aumiller, D. y M. Gertz. 2023. Unihd at tsar-2022 shared task: Is compute all we need for lexical simplification. *arXiv preprint arXiv:2301.01764*.

Brown, T., B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, y others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Calle-Gómez, J., A. García-Serrano, y P. Martínez. 2006. Intentional processing as a key for rational behaviour through natural interaction. *INTERACTING WITH COMPUTERS*, 18:1419–1446.

Carroll, J., G. Minnen, Y. Canning, S. Devlin, y J. Tait. 1998. Practical simplification of english newspaper text to assist aphasic readers. En *Proceedings of the AAAI-98 Workshop on Integrating Artificial Intelligence and Assistive Technology*, páginas 7–10. Citeseer.

Chiu, K.-L., A. Collins, y R. Alexander. 2021. Detecting hate speech with gpt-3. *arXiv preprint arXiv:2103.12407*.

De la Rosa, J., P. Eduardo, R. Manuel, V. Paulo, G. d. P. Pablo, y G. María. 2022. Bertin: Efficient pre-training of a spanish language model using perplexity sampling. *Procesamiento del Lenguaje Natural*, 68(0):13–23.

de la Rosa, J., P.-P. Álvaro, R. Salvador, y G.-B. Elena. 2023. Alberti, a multilingual domain specific language model for poetry analysis. *Procesamiento del Lenguaje Natural*, 71(0):215–225.

Devlin, S. 1998. The use of a psycholinguistic database in the simplification of text for aphasic readers. *Linguistic databases*.

Ermakova, L., E. Sanjuan, J. Kamps, S. Huet, I. Ovchinnikova, D. Nurbakova, S. Araújo, R. Hannachi, E. Mathurin, y P. Bellot. 2022. Overview of the clef 2022 simpletext lab: Automatic simplification of scientific texts. En *International Conference of the Cross-Language Evaluation Forum for European Languages*, páginas 470–494. Springer.

Grabar, N. y H. Saggion. 2022. Evaluation of automatic text simplification: Where are we now, where should we go from here. En *Actes de la 29e Conférence sur le Traitement Automatique des Langues Naturelles. Volume 1 : conférence principale*, páginas 453–463, Avignon, France, 6. ATALA.

Gutiérrez-Fandiño, A., A.-E. Jordi, P. Marc, L.-P. Joan, S.-O. Joaquin, P. C. Casimiro, A.-O. Carme, R.-P. Carlos, G.-A. Aitor, y V. Marta. 2022. Maria: Spanish language models. *Procesamiento del Lenguaje Natural*, 68(0):39–60.

Jiang, Z., F. F. Xu, J. Araki, y G. Neubig. 2020. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438.

Lastra-Díaz, J. J., A. Lara-Clares, y A. Garcia-Serrano. 2022. Hesml: a real-time semantic measures library for the biomedical domain with a reproducible survey. *BMC bioinformatics*, 23(1):23.

Lester, B., R. Al-Rfou, y N. Constant. 2021. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*.

Liu, X., Y. Zheng, Z. Du, M. Ding, Y. Qian, Z. Yang, y J. Tang. 2023. Gpt understands, too. *AI Open*.

Martínez-Fernández, J. L., J. V. Román, A. García-Serrano, y J. C. González-Cristóbal. 2006. Combining textual and visual features for image retrieval. En C. Peters F. C. Gey J. Gonzalo H. Müller G. J. F. Jones M. Kluck B. Magnini, y M. de Rijke, editores, *Accessing Multilingual Information Repositories*, páginas 680–691.

Moreno-Sandoval, A., A. Gisbert, y H. Montoro. 2020. Fint-esp: A corpus of financial reports in spanish. *Fuster, et al., editors, Multiperspectives in analysis and corpus design*, páginas 89–102.

Navarrate-Parra, O., V. Uc-Cetina, y J. Reyes-Magaña. 2023. Aligning a medium-size gpt model in english to a small closed domain in spanish. *Procesamiento del Lenguaje Natural*, 71(0):87–95.

North, K., T. Ranasinghe, M. Shardlow, y M. Zampieri. 2023. Deep learning approaches to lexical simplification: A survey. *arXiv preprint arXiv:2305.12000*.

Paetzold, G. H. y L. Specia. 2017. A survey on lexical simplification. *Journal of Artificial Intelligence Research*, 60:549–593.

Qiang, J., Y. Li, Y. Zhu, Y. Yuan, y X. Wu. 2020. Lexical simplification with pretrained encoders. En *Proceedings of the AAAI Conference on Artificial Intelligence*, volumen 34, páginas 8649–8656.

Radford, A., J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, y others. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Rello, L., C. Bayarri, A. Górriz, R. Baeza-Yates, S. Gupta, G. Kanvinde, H. Saggion, S. Bott, R. Carlini, y V. Topac. 2013. Dyswebxia 2.0! more accessible text for people with dyslexia. En *Proceedings of the 10th International Cross-Disciplinary Conference on Web Accessibility*, páginas 1–2.

Saggion, H., S. Štajner, S. Bott, S. Mille, L. Rello, y B. Drndarevic. 2015. Making it simplext: Implementation and evaluation of a text simplification system for spanish. *ACM Transactions on Accessible Computing (TACCESS)*, 6(4):1–36.

Saggion, H., S. Štajner, D. Ferrés, K. C. Sheang, M. Shardlow, K. North, y M. Zampieri. 2023. Findings of the tsar-2022 shared task on multilingual lexical simplification. *arXiv preprint arXiv:2302.02888*.

Serrano, A. V., G. G. Subies, H. M. Zamorano, N. A. Garcia, D. Samy, D. B. Sanchez, A. M. Sandoval, M. G. Nieto, y A. B. Jimenez. 2022. Rigoberta: A state-of-the-art language model for spanish.

Shardlow, M. 2013. A comparison of techniques to automatically identify complex words. En *51st annual meeting of the association for computational linguistics proceedings of the student research workshop*, páginas 103–109.

Shardlow, M. 2014. A survey of automated text simplification. *International Journal of Advanced Computer Science and Applications*, 4(1):58–70.

Stajner, S., D. Ferrés, M. Shardlow, K. North, M. Zampieri, y H. Saggion. 2022. Lexical simplification benchmarks for english, portuguese, and spanish. *arXiv preprint arXiv:2209.05301*.

Vásquez-Rodríguez, L., N. Nguyen, M. Shardlow, y S. Ananiadou. 2022. Uom&mmu at tsar-2022 shared task: Prompt learning for lexical simplification. En *Proceedings of the Workshop on Text Simplification, Accessibility, and Readability (TSAR-2022)*, páginas 218–224.

Yimam, S. M., C. Biemann, S. Malmasi, G. H. Paetzold, L. Specia, S. Štajner, A. Tack, y M. Zampieri. 2018. A report on the complex word identification shared task 2018. *arXiv preprint arXiv:1804.09132*.