MÁSTER EN INGENIERÍA INDUSTRIAL

TRABAJO FIN DE MÁSTER

Revisión de modelos de lenguaje y aplicación a la automatización de tareas de oficina en empresas industriales

Autor: Jacobo Zamora Fraguas

Director: Álvaro López López

Madrid

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

Revisión de modelos de lenguaje y aplicación a la automatización de tareas de oficina en empresas industriales

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el

curso académico 2022/23 es de mi autoría, original e inédito y

no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido

tomada de otros documentos está debidamente referenciada.

Fdo.: Jacobo Zamora Fraguas Fecha: 29 / 08 / 2023

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Álvaro Jesús López López

Firmado digitalmente por Álvaro Jesús López López
Fecha: 2023.08.30 16:19:20 +02'00'

Fdo.: Álvaro López López Fecha: ……/ ……/ ……

MÁSTER EN INGENIERÍA INDUSTRIAL

TRABAJO FIN DE MÁSTER

Revisión de modelos de lenguaje y aplicación a la automatización de tareas de oficina en empresas industriales

Autor: Jacobo Zamora Fraguas

Director: Álvaro López López

Madrid

REVISIÓN DE MODELOS DE LENGUAJE Y APLICACIÓN A LA AUTOMATIZACIÓN DE TAREAS DE OFICINA EN EMPRESAS INDUSTRIALES

Autor: Zamora Fraguas, Jacobo.
Director: López López, Álvaro.
Entidad Colaboradora: ICAI – Universidad Pontificia Comillas

RESUMEN DEL PROYECTO

Este proyecto se enfoca en la evaluación y aplicación de *Large Language Models* (LLMs) en el procesamiento de lenguaje natural (PLN). Se analizan las propiedades técnicas, ventajas y desventajas de los LLMs, resaltando su amplio uso en traducción automática, asistentes virtuales y más. Se consideran también aspectos éticos y de privacidad en su aplicación. El proyecto se extiende a la automatización de tareas de oficina mediante *Robotic Process Automation* (RPA). Se identifican tareas aptas para la automatización, como análisis de texto y generación de informes, y se evalúa su impacto en la eficiencia empresarial. Luego, se presenta una demostración práctica utilizando *GPT-3.5 Turbo* de *OpenAI* para generar respuestas de correo electrónico. Esto ilustra cómo los LLMs pueden agilizar tareas cotidianas.

Palabras clave: Modelos de Lenguaje, Automatización, Inteligencia Artificial, Tareas de Oficina.

1. Introducción

El procesamiento del lenguaje natural (PLN) ha ido evolucionado en los últimos años gracias a los avances en el campo del *Deep Learning*, especialmente con la aparición de los *Large Language Models* (LLMs). Estos modelos representan un hito significativo en la capacidad de las máquinas para comprender y generar lenguaje humano de manera coherente y significativa. A medida que los LLMs evolucionan y se perfeccionan, su potencial para cambiar la forma en que interactuamos con la tecnología y la información se vuelve cada vez más evidente. [1]

Los primeros sistemas de PLN se centraban en comprender patrones y estructuras gramaticales dentro del lenguaje, pero la verdadera revolución llegó con la aparición del *Deep Learning* y las redes neuronales. Gracias a las redes neuronales se pudo empezar a analizar grandes cantidades de datos y aumentar la capacidad de computación, lo que dio paso a sistemas de PLN más avanzados. Con estos nuevos algoritmos de aprendizaje, los

LLMs se convirtieron en modelos de inteligencia artificial capaces de procesar información compleja y generar respuestas coherentes basadas en contextos dados. [2]

Hoy en día, los LLMs como BERT y T5 están siendo ampliamente utilizados en una variedad de aplicaciones, desde la síntesis de textos y traducción automática hasta la asistencia virtual y la creación de contenido. Su versatilidad y eficiencia han llamado la atención de diversas industrias, desde la atención médica y la educación hasta el comercio electrónico. La rapidez con la que estos modelos han evolucionado ha llevado a especular que en el futuro cercano, tareas complejas en diferentes campos se realizarán o mejorarán significativamente mediante el uso de LLMs.[3]

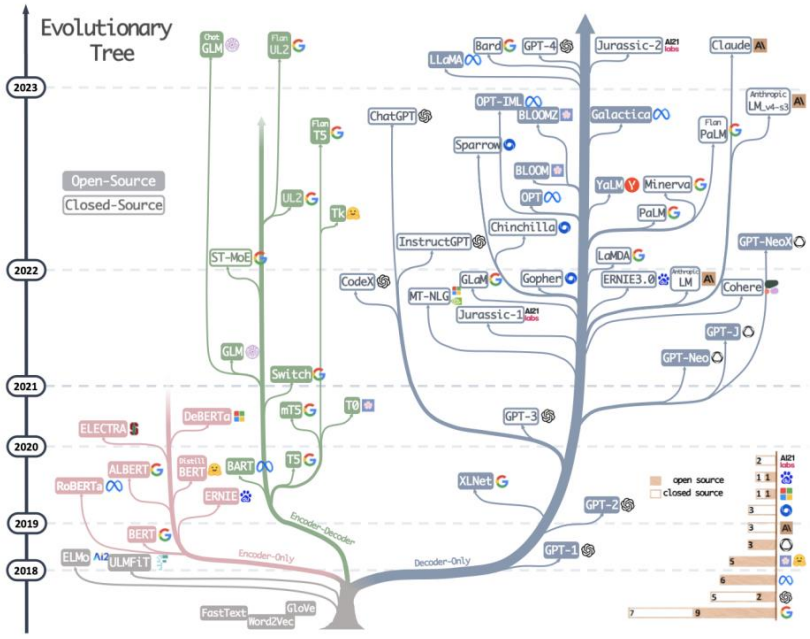

Figura 1. Árbol de evolución de los LLMs [3]

Sin embargo, a pesar de los avances prometedores, los LLMs también enfrentan desafíos importantes. Temas como la privacidad de los datos y la igualdad en el acceso a la tecnología deben abordarse con responsabilidad y ética. Es crucial considerar el impacto social y potenciales sesgos al desarrollar y aplicar estos modelos para garantizar que se utilicen de manera responsable y beneficiosa para la sociedad.

2. Definición del Proyecto

En este trabajo se aborda lograr una comprensión completa y exhaustiva de los LLMs disponibles en la literatura, evaluando sus propiedades, ventajas y desventajas. Se han analizado las tendencias actuales y futuras de estos modelos, prestando atención a aspectos

técnicos como eficiencia computacional, tiempos de respuesta y posibilidad de ejecución local. Además, se considerarán las barreras legales, de privacidad y los marcos éticos asociados con su uso.

Se ha podido comprobar que ha habido una gran evolución en el desarrollo de modelos de lenguaje. Estos modelos aprovechan las técnicas de Deep Learning, en particular las arquitecturas de modelo transformer, para procesar y generar texto similar al humano a una escala sin precedentes. Al entrenarse con grandes cantidades de datos de texto, estos modelos adquieren una comprensión contextual y la capacidad de generar respuestas coherentes y contextualmente relevantes. Se puede resumir la gran mayoría de los LLMs en dos categorías. *Encoder-Decoder* y *Decoder only*. [4][5]

Para poder realizar un análisis exhaustivo de los modelos se han seleccionado las que se han considerado las 13 principales características que permiten definir por completo cada uno de los LLMs seleccionados. Son las siguientes: Desarrollador; Año; Tipo de Modelo; Tamaño del modelo; Como ha sido entrenado; Usos y Tareas para los que se desarrolló; Precisión en dichas tareas; APIs que ofrece; Localización del Sistema y Donde se puede ejecutar; Privacidad de los datos; Propiedad Intelectual; Trasparencia del desarrollador y del modelo; y Regulación específica.

Los modelos analizados han sido los siguientes: BERT, T5, ELECTRA, REFORMER, GPT-4, RoBERTa, LLAMA, ELMO, XLNET, DistilBERT, UniLM, Megatron-Turing NLG, LaMDa, Jurassic-1, Gopher y Chinchilla.

Posteriormente tras el análisis de los modelos más relevantes de la literatura, se realiza un análisis exhaustivo de las posibles tareas de back office en empresas industriales que puedan ser automatizadas mediante LLMs. Este proceso de automatización se conoce como *Robotic Process Automation* (RPA). [6]

Son sistemas que permiten automatizar tareas muy repetitivas que normalmente representan tareas de escaso valor añadido para las personas. En muchos sectores, estos sistemas suelen emplearse para automatizar tareas de back office, donde pueden aportar un alto retorno a la inversión. Además, son especialmente útiles en casos de uso que implican sistemas muy dispares y poco integrados entre sí, donde pueden utilizarse para eliminar la necesidad de intervención humana en cada paso del flujo de trabajo y sustituirla por un operador automático, que se limita a simular exactamente lo que hace el ser humano, pero de forma

automatizada. El resultado es un flujo de trabajo más eficiente que requiere menos intervenciones humanas, lo que se traduce en un menor riesgo de errores manuales y una reducción de los costes operativos.

Para evaluar la eficacia de cualquier tarea a RPA, nos vamos a centrar en dos aspectos. El primero de ellos se centrará en valorar si la tarea es rutinaria o no rutinaria y el segundo de ellos si requiere el uso de funciones manuales o capacidades cognitivas. Las tareas altamente cognitivas requieren pensamiento creativo, al igual que las tareas no rutinarias con pocos o ningún patrón que tienen una gran variabilidad. Este tipo de tareas no son adecuadas para la automatización. Una regla práctica para determinar si una tarea es apta para la automatización es determinar si se pueden escribir con precisión todos los pasos del proceso, teniendo en cuenta todos los posibles acontecimientos y resultados a lo largo del camino. Aunque los avances en Inteligencia Artificial han permitido automatizar algunas tareas no rutinarias, el principio general que utilizaremos para determinar la eficacia seguirá siendo el mismo. [7]

Para determinar si una tarea es adecuada para RPA, hay que tener en cuenta más factores. Además de la necesidad de que una tarea sea manual y rutinaria, una empresa dispuesta a utilizar la RPA debe considerar si es viable reemplazar a los humanos por robots para determinadas tareas y cuáles serían las implicaciones a largo plazo de tal decisión. Estos criterios también influyen en las decisiones estratégicas de los proveedores de RPA en relación con la comercialización y el marketing de la tecnología. Se han utilizado estos criterios para asignar una puntación a las tareas que hemos seleccionado para determinar así su eficacia a la automatización. Entre ellos están los siguientes: Elevado número de transacciones, Tareas consideradas para RPA son tareas que se realizan de manera frecuente o incluyen un gran volumen de sub-tareas; o Necesidad de acceso a múltiples sistemas, El que haya que acceder a múltiples sistemas es un factor relevante. Por ejemplo, copiar datos de un Excel a una base de datos. [8]

Tras establecer los criterios de RPA, se identificaron las tareas más relevantes y se analiza cómo podrían llevarse a cabo. Se analizan las posibles repercusiones de su automatización, evaluando el impacto en la eficiencia y los procesos empresariales. Las tareas analizadas son las siguientes: Tareas de análisis de textos; Extracción de datos, análisis de datos no estructurados y su organización; Clasificación automática de documentos; Generación automática de informes y resúmenes; Traducción; Generación automática de sugerencias de

respuestas de mensajes y correos electrónicos; Asistentes virtuales; Controles y verificaciones de *compliance*; y Sistemas de análisis y prevención de riesgos.

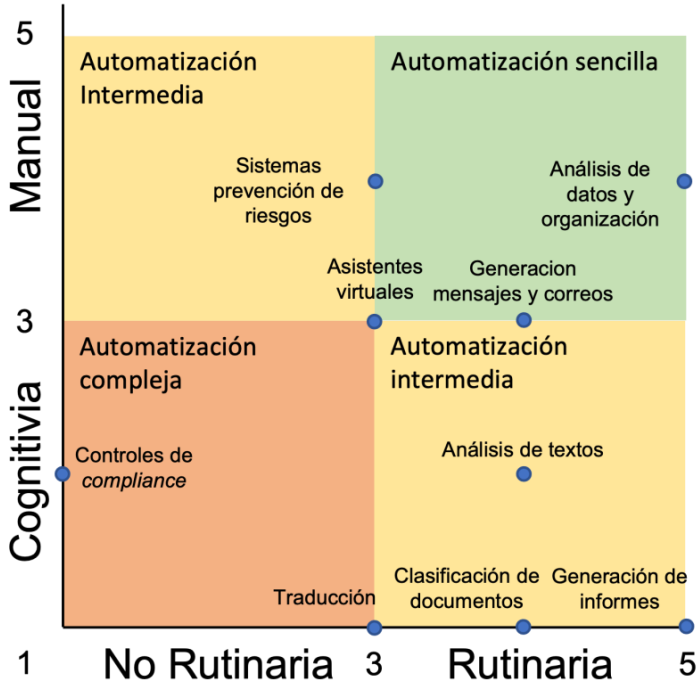

Figura 2. Gráfica de automatización con las tareas [8]

Finalmente se realizó un cruce de los resultados de la revisión de modelos y el análisis de tareas para evaluar la eficiencia actual de los modelos en la realización de dichas tareas. Se ha propuesto un caso de aplicación concreto y se ha desarrollado una demo. Se ha seleccionado el modelo más adecuado para el caso, teniendo en cuenta las consideraciones técnicas, legales y éticas.

Se escogió concertarse directamente a la API de *Openai* para acceder a sus modelos, para simplificar al máximo la demo y a su vez utilizar uno de los modelos más relevantes de la actualidad. Se utilizará por lo tanto *GPT 3.5 Turbo* para la demo. En cuanto a la tarea que se quiere automatizar, los modelos GPT se especializan en la generación de texto y respuestas a preguntas por lo tanto se seleccionará una tarea en estos ámbitos para la demo. A la hora de la selección se ha tenido en cuanta intentar automatizar una tarea que resulte útil y de realización diaria para todo el mundo. Es por ello por lo que se selecciona la generación automática de correos electrónicos, ya que en el día a día se reciben una gran cantidad de ellos y el poder partir de una posible respuesta facilitará la vida a quienes utilicen la aplicación.

Se desarrolló un sencillo programa haciendo uso de *Python*. En esta pequeña demo se genera una sugerencia a un correo electrónico entrante. Los parámetros de entrada serán conversaciones de correo electrónico pasadas para que el modelo pueda analizar el estilo de escritura del usuario para utilizarlo en su sugerencia de respuesta.

3. Resultados

El programa analiza el historial de correos del usuario para posteriormente solicitar la introducción de un nuevo correo electrónico para generar con la información historia una posible respuesta utilizando el estilo de escritura del usuario. Dado que es complicado acertar por completo que hubiese querido responder el usurario, se le da la opción de en la misma interfaz editar el correo antes de enviarlo.

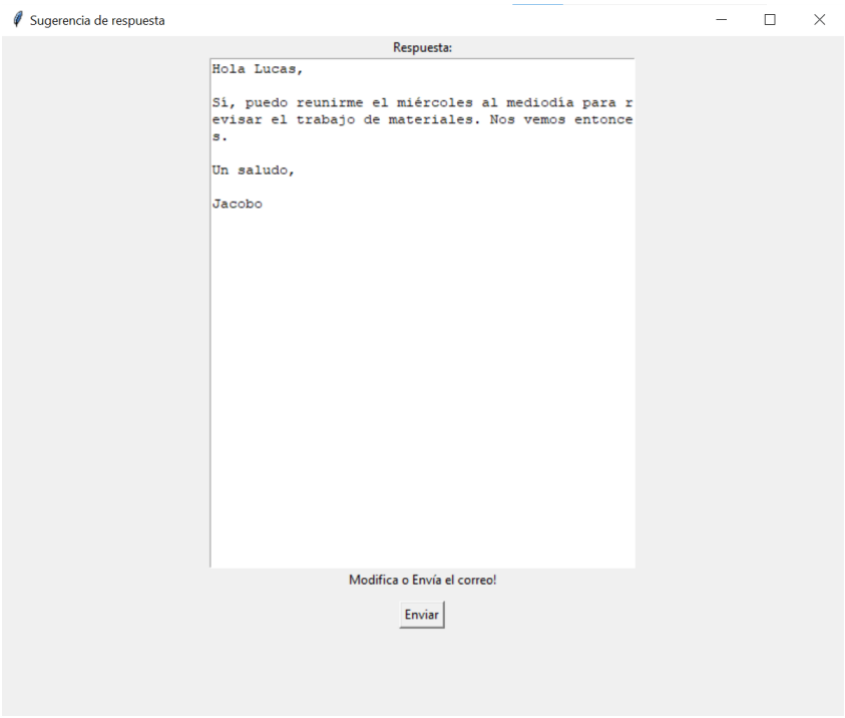

Figura 3. Correo Automáticamente Generado

A su vez, una vez el usuario está satisfecho con la respuesta sugerida y editada, el correo electrónico se almacena junto con el resto de históricos para que la siguiente vez que uso el programa, tenga la información histórica actualizada.

4. Conclusiones

En este trabajo se ha realizado un análisis exhaustivo de los modelos de lenguaje más importantes, evaluando su capacidad para abordar tareas de back office y su eficiencia en la automatización inteligente de procesos. Se destacó la importancia de contar con LLMs

eficientes y precisos para el procesamiento del lenguaje natural, y se exploraron diversas áreas de aplicación, como análisis de textos, extracción de datos y generación de informes. La comparativa entre modelos y tareas resaltó el papel fundamental de los LLMs en la mejora de la eficiencia y la reducción de la carga de trabajo manual en el ámbito empresarial. Se destacó el potencial en constante evolución de estos modelos, así como su combinación adecuada con tareas para la automatización inteligente y la transformación digital.

La demostración final evidenció la facilidad con la que ciertas tareas pueden ser automatizadas, demostrando cómo la selección adecuada de modelos y aplicaciones permite iniciar rápidamente la automatización de tareas diarias de oficina. Este trabajo sienta las bases para futuras investigaciones y aplicaciones en la utilización de LLMs en el back office, enfatizando el potencial de estas tecnologías para revolucionar y mejorar el ámbito laboral mediante la automatización inteligente y la optimización de procesos. La combinación de modelos avanzados con tareas optimizadas abre un camino prometedor hacia un futuro en el que la eficiencia empresarial y la productividad se vean impulsadas por la inteligencia artificial y el procesamiento del lenguaje natural.

5. Referencias

[1] S. R. Joseph, H. Hlomani, K. Letsholo, F. Kaniwa, and K. Sedimo, "Natural Language Processing: A Review," *International Journal of Research in Engineering and Applied Sciences*, vol. 6, no. 3, 2016, Accessed: Aug. 05, 2023.

[2] M. Rahman Minar and J. Naher, "Recent Advances in Deep Learning: An Overview," 2018.

[3] J. Yang, H. Jin, and X. Han, "Harnessing the Power of LLMs in Practice: A Survey on ChatGPT and Beyond; Harnessing the Power of LLMs in Practice: A Survey on ChatGPT and Beyond," 2023.

[4] K. Aitken, "Understanding How Encoder-Decoder Architectures Attend," 2021.

[5] J. Lin, X. Mao, Y. Chen, L. Xu, Y. He, and H. Xue, "D 2 ETR: Decoder-Only DETR with Computationally Efficient Cross-Scale Attention," 2022.

[6] R. Syed *et al.*, "Robotic Process Automation: Contemporary themes and challenges," *Comput Ind*, vol. 115, p. 103162, Feb. 2020.

[7] Commission de Surveillance du Secteur Financier, “OPPORTUNITIES, RISKS AND RECOMMENDATIONS FOR THE FINANCIAL SECTOR,” 2018.

[8] A. Asatiani and E. Penttinen, “Turning robotic process automation into commercial success - Case OpusCapita,” *Journal of Information Technology Teaching Cases*, vol. 6, no. 2, pp. 67–74, Nov. 2016.

REVIEW OF LANGUAGE MODELS AND APPLICATION TO THE AUTOMATION OF OFFICE TASKS IN INDUSTRIAL COMPANIES

Author: Zamora Fraguas, Jacobo.
Supervisor: López López, Álvaro.
Collaborating Entity: ICAI – Universidad Pontificia Comillas

ABSTRACT

This project focuses on the evaluation and application of Large Language Models (LLMs) in natural language processing (NLP). The technical properties, advantages, and disadvantages of LLMs are analysed, highlighting their wide use in machine translation, virtual assistants and more. Ethical and privacy aspects of their application are also considered. The project extends to the automation of office tasks using Robotic Process Automation (RPA). Tasks suitable for automation, such as text analysis and report generation, are identified and their impact on business efficiency is assessed. A practical demonstration is then presented using OpenAI's GPT-3.5 Turbo to generate email responses. This illustrates how LLMs can streamline everyday tasks.

Keywords: Language Models, Artificial Intelligence, Automation, Office Tasks.

1. Introduction

Natural language processing (NLP) has been evolving in recent years thanks to advances in the field of Deep Learning, especially with the emergence of Large Language Models (LLMs). These models represent a significant milestone in the ability of machines to understand and generate human language in a coherent and meaningful way. As LLMs evolve and are refined, their potential to change the way we interact with technology and information becomes increasingly apparent. [1]

Early PLN systems focused on understanding patterns and grammatical structures within language, but the real revolution came with the emergence of Deep Learning and neural networks. Thanks to neural networks, it was possible to start analysing large amounts of data and increase computational capacity, which led to more advanced PLN systems. With these new learning algorithms, LLMs became artificial intelligence models capable of processing complex information and generating coherent answers based on given contexts. [2]

Today, LLMs such as BERT and T5 are being widely used in a variety of applications, from text summarization and machine translation to virtual assistance and content creation. Their versatility and efficiency have attracted the attention of diverse industries, from healthcare and education to e-commerce. The speed with which these models have evolved has led to

speculation that in the near future, complex tasks in different fields will be performed or significantly improved through the use of LLMs [3].

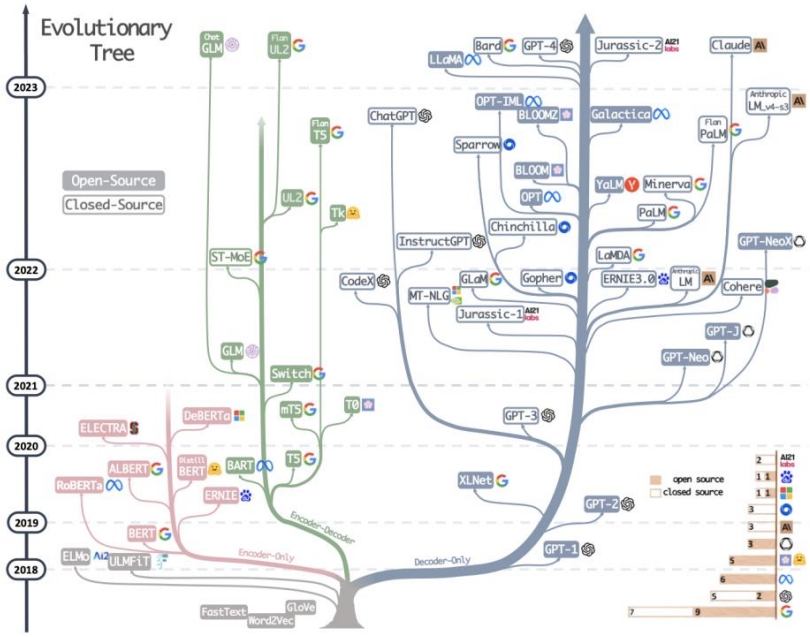

Figure 1. Evolutionary tree of LLMs [3]

However, despite promising developments, LLMs also face significant challenges. Issues such as data privacy and equal access to technology must be addressed responsibly and ethically. It is crucial to consider the social impact and potential biases when developing and applying these models to ensure that they are used in a responsible and socially beneficial way.

2. Project Definition

This paper aims to achieve a complete and comprehensive understanding of the LLMs available in the literature, evaluating their properties, advantages, and disadvantages. Current and future trends of these models have been analysed, paying attention to technical aspects such as computational efficiency, response times and local runnability. In addition, the legal barriers, privacy, and ethical frameworks associated with their use will be considered.

It has been found that there has been a great evolution in the development of language models. These models leverage Deep Learning techniques, in particular transformer model architectures, to process and generate human-like text on an unprecedented scale. By training on large amounts of text data, these models acquire contextual understanding and the ability

to generate coherent and contextually relevant responses. The vast majority of LLMs can be summarised in two categories. Encoder-Decoder and Decoder only. [4][5]

In order to carry out an exhaustive analysis of the models, what are considered to be the 13 main characteristics that allow each of the selected LLMs to be fully defined have been selected. They are the following: Developer; Year; Type of Model; Size of the model; How it has been trained; Uses and Tasks for which it was developed; Accuracy in these tasks; APIs it offers; Location of the System and Where it can be executed; Privacy of the data; Intellectual Property; Transparency of the developer and of the model; and Specific Regulation.

The models analysed were the following: BERT, T5, ELECTRA, REFORMER, GPT-4, RoBERTa, LLAMA, ELMO, XLNET, DistilBERT, UniLM, Megatron-Turing NLG, LaMDa, Jurassic-1, Gopher and Chinchilla.

Following the analysis of the most relevant models in the literature, an exhaustive analysis of the possible back office tasks in industrial companies that can be automated by means of LLMs is carried out. This automation process is known as Robotic Process Automation (RPA). [6]

They are systems that automate highly repetitive tasks that typically represent low value-added tasks for people. In many industries, these systems are often used to automate back office tasks, where they can provide a high return on investment. They are also particularly useful in use cases involving disparate and poorly integrated systems, where they can be used to remove the need for human intervention at every step of the workflow and replace it with an automated operator, which simply simulates exactly what humans do, but in an automated way. The result is a more efficient workflow that requires less human intervention, resulting in a reduced risk of manual errors and reduced operational costs.

To assess the effectiveness of any RPA task, we will focus on two aspects. The first of these will focus on assessing whether the task is routine or non-routine and the second on whether it requires the use of manual functions or cognitive abilities. Highly cognitive tasks require creative thinking, as do non-routine tasks with little or no pattern and high variability. These types of tasks are not suitable for automation. A rule of thumb for determining whether a task is suitable for automation is to determine whether all the steps in the process can be accurately scripted, taking into account all possible events and outcomes along the way.

Although advances in Artificial Intelligence have made it possible to automate some non-routine tasks, the general principle we will use to determine effectiveness will remain the same. [7]

To determine whether a task is suitable for RPA, more factors need to be taken into account. In addition to the need for a task to be manual and routine, a company willing to use RPA must consider whether it is feasible to replace humans with robots for certain tasks and what the long-term implications of such a decision would be. These criteria also influence the strategic decisions of RPA providers regarding the commercialisation and marketing of the technology. These criteria have been used to score the tasks we have selected to determine their effectiveness for automation. These include the following: High number of transactions, Tasks considered for RPA are tasks that are performed frequently or include a large volume of sub-tasks; o Need to access multiple systems, having to access multiple systems is a relevant factor. For example, copying data from an Excel to a database. [8]

After establishing the RPA criteria, the most relevant tasks were identified and how they could be carried out is analysed. The potential impact of their automation is analysed, assessing the impact on efficiency and business processes. The tasks analysed are the following: Text analysis tasks; Data extraction, analysis of unstructured data and its organisation; Automatic document classification; Automatic generation of reports and summaries; Translation; Automatic generation of suggestions for message and email replies; Virtual assistants; Compliance checks and controls; and Risk analysis and prevention systems.

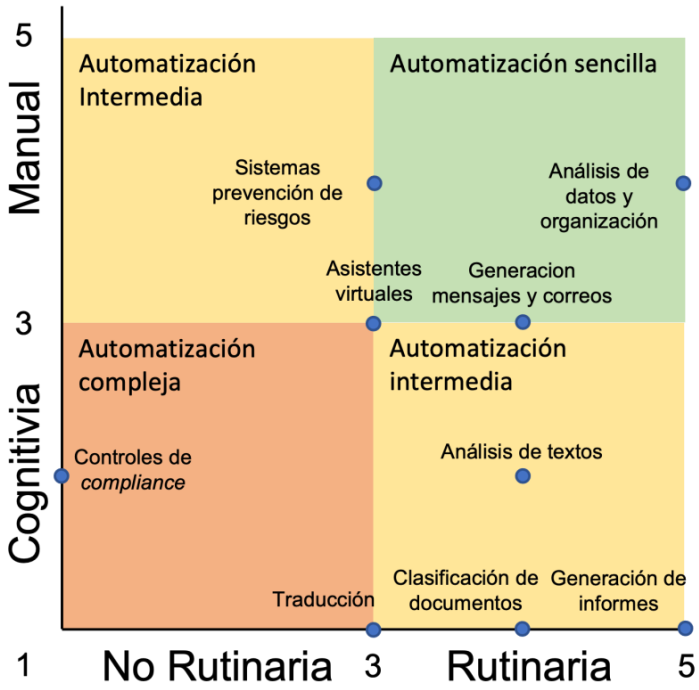

Figure 2. Automation chart with the tasks [8]

Finally, the results of the model review and the task analysis were cross-checked to assess the current efficiency of the models in performing these tasks. A concrete application case has been proposed and a demo has been developed. The most appropriate model for the case has been selected, taking into account technical, legal and ethical considerations.

A direct connection to the Openai API to access its models was chosen in order to simplify the demo as much as possible and at the same time use one of the most relevant models currently available. GPT 3.5 Turbo will therefore be used for the demo. As for the task to be automated, GPT models specialise in text generation and question answering, therefore a task in these areas will be selected for the demo. In the selection process, the aim was to automate a task that would be useful and that everyone could perform on a daily basis. This is why the automatic generation of e-mails has been selected, as a large number of e-mails are received on a daily basis and being able to start from a possible answer will make life easier for those who use the application.

A simple program was developed using Python. In this small demo a suggestion is generated to an incoming email. The input parameters will be past email conversations so that the model can analyse the user's writing style for use in its suggested response.

3. Results

The programme analyses the user's email history and then prompts the user to enter a new email to generate a possible reply using the user's writing style. Since it is difficult to completely guess what the user would have wanted to reply, the user is given the option to edit the email in the same interface before sending it.

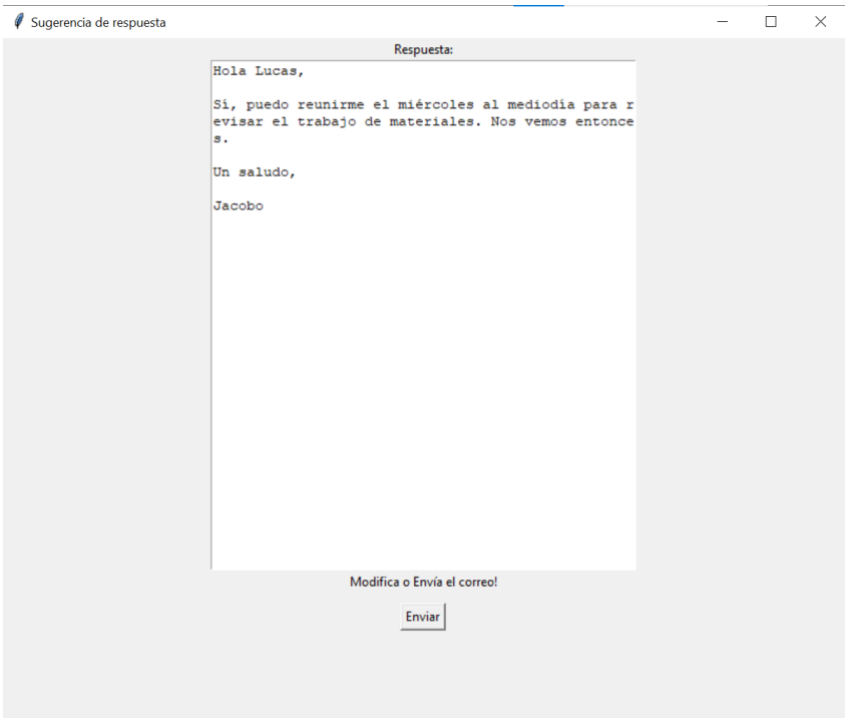

Figure 3. Automatically Generated Mail

In turn, once the user is satisfied with the suggested and edited response, the e-mail is stored along with the rest of the history so that the next time the programme is used, the user has the updated historical information.

4. Conclusions

In this paper, a comprehensive analysis of the most important language models has been carried out, assessing their ability to address back office tasks and their efficiency in intelligent process automation. The importance of efficient and accurate LLMs for natural language processing was highlighted, and several application areas, such as text analysis, data mining and report generation, were explored. The comparison between models and tasks highlighted the key role of LLMs in improving efficiency and reducing manual workload in the business environment. The constantly evolving potential of these models was highlighted, as well as their appropriate combination with tasks for intelligent automation and digital transformation.

The final demonstration showed the ease with which certain tasks can be automated, demonstrating how the right choice of models and applications can quickly initiate the automation of everyday office tasks. This work lays the foundation for future research and applications in the use of LLMs in the back office, emphasising the potential of these technologies to revolutionise and improve the workplace through intelligent automation and process optimisation. The combination of advanced models with optimised tasks opens a promising path towards a future where business efficiency and productivity are driven by artificial intelligence and natural language processing.

5. References

[1] S. R. Joseph, H. Hlomani, K. Letsholo, F. Kaniwa, and K. Sedimo, "Natural Language Processing: A Review," *International Journal of Research in Engineering and Applied Sciences*, vol. 6, no. 3, 2016, Accessed: Aug. 05, 2023.

[2] M. Rahman Minar and J. Naher, "Recent Advances in Deep Learning: An Overview," 2018.

[3] J. Yang, H. Jin, and X. Han, "Harnessing the Power of LLMs in Practice: A Survey on ChatGPT and Beyond; Harnessing the Power of LLMs in Practice: A Survey on ChatGPT and Beyond," 2023.

[4] K. Aitken, "Understanding How Encoder-Decoder Architectures Attend," 2021.

[5] J. Lin, X. Mao, Y. Chen, L. Xu, Y. He, and H. Xue, "D 2 ETR: Decoder-Only DETR with Computationally Efficient Cross-Scale Attention," 2022.

[6] R. Syed *et al.*, "Robotic Process Automation: Contemporary themes and challenges," *Comput Ind*, vol. 115, p. 103162, Feb. 2020.

[7] Commission de Surveillance du Secteur Financier, "OPPORTUNITIES, RISKS AND RECOMMENDATIONS FOR THE FINANCIAL SECTOR," 2018.

[8] A. Asatiani and E. Penttinen, "Turning robotic process automation into commercial success - Case OpusCapita," *Journal of Information Technology Teaching Cases*, vol. 6, no. 2, pp. 67–74, Nov. 2016.

Índice de la memoria

Índice de figuras

Índice de tablas

1. INTRODUCCIÓN

El procesamiento del lenguaje natural (PLN) es un campo que ha experimentado un gran impulso en los últimos años en particular gracias al uso de modelos de *Deep Learning*. El PLN es un campo que se centra en la interacción entre los ordenadores y el lenguaje humano. Se basa en aplicaciones, algoritmos, modelos y técnicas que permiten a los ordenadores comprender, interpretar y generar lenguaje humano de forma que tenga sentido y sobre todo que nos sea útil. [1]

La evolución de los modelos nos lleva hasta los *Large Language Models* (LLMs) que son el claro ejemplo de cómo el *Deep Learning* ha revolucionado el campo del PLN. El desarrollo de LLMs ha sido un proceso evolutivo que se remonta varias décadas atrás. Los primeros sistemas de PLN se centraron en la comprensión de patrones y la identificación de estructuras gramaticales dentro del lenguaje. La verdadera revolución del *Deep Lear*ning llegó en la década de 2010. Se empezó a poder utilizar grandes cantidades de datos para los análisis y aumentó la capacidad de computación de los ordenadores. El desarrollo de algoritmos de aprendizaje, permitieron la creación de sistemas de procesamiento de lenguaje natural más avanzados. [2]

Estos avances y la rápida evolución que en los últimos años están teniendo está cambiando la manera con la que interactuamos con la tecnología y la información. Estos nuevos modelos de inteligencia artificial, como BERT y T5, son capaces de generar texto coherente y comprensible a partir de una entrada de texto. Se han empezado a implantar la utilización de modelos de lenguaje en una gran cantidad de sectores y para diversas aplicaciones. Los LLMs, hoy en día, se utilizan para aplicaciones que van, desde la síntesis de textos y traducción automática, hasta la asistencia virtual y la creación de contenido. Empresas como *Google*, *Microsoft*, *OpenAI*, *Anthropic* y *Amazon* están invirtiendo en la investigación y desarrollo de LLMs, y las empresas emergentes están explorando nuevas aplicaciones para esta tecnología en campos como la atención médica, la educación y el comercio electrónico. [3]

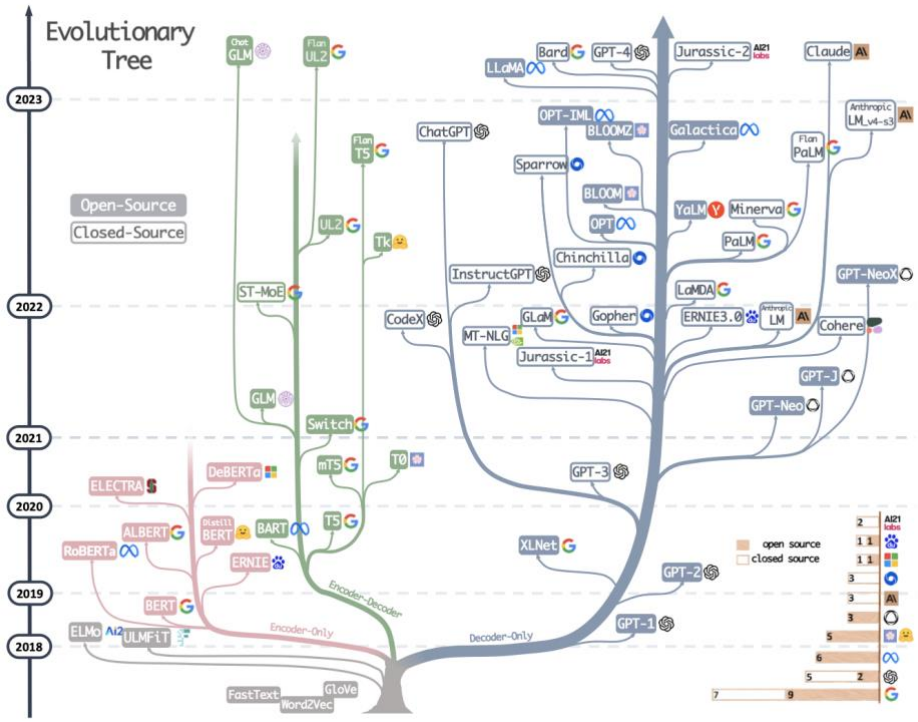

Figura 1. Árbol de evolución de los LLMs [3]

Como podemos ver en la figura a partir de 2021 aparecen cada vez más y más modelos. La gran mayoría de ellos de arquitectura *Decoder-Only*. Se espera que esta rápida evolución vaya a tener un gran impacto en la sociedad, hasta llegar al punto en el que tareas complejas en una gran variedad de campos se realicen por estos sistemas o se utilicen para mejorar la accesibilidad para aquellos con discapacidades del habla o la escritura y muchas otras aplicaciones.

Es importante tener en cuenta que los LLMs todavía enfrentan desafíos significativos en áreas como la igualdad y la privacidad de los datos. Sin embargo, con un enfoque ético y responsable hay un gran potencial para utilizar los LLMs de manera efectiva para mejorar nuestras vidas.

1.1 MOTIVACIÓN Y OBJETIVOS DEL PROYECTO

Como hemos podido ver, los LLMs son un tema de gran relevancia en la actualidad. Su rápido desarrollo nos hace plantearnos hacia donde nos van a llevar y como va a ser su evolución.

A su vez, el hecho de que puedan llegar a suplir tareas complicadas y monótonas realizadas por seres humanos nos hacen ver la importancia que van a tener en un futuro en el día a día de las personas.

Es por ello por lo que surge la necesidad de analizar cómo está actualmente esta cuestión y las capacidades que tienen los LLMs. Para poder entender cómo se van a seguir desarrollando y especializando en distintas tareas es necesario entender los modelos y sus características para no solo ver su potencial, sino también qué implicaciones tienen para nosotros. Actualmente la privacidad y la ética son dos temas de gran importancia los cuales merece la pena analizar ya que, por ejemplo, el aumento de ciberataques que suceden día a día podría tener un gran impacto en este mundo ya que la posible dependencia de nuestros datos y sistemas de uso diario en aplicaciones basadas de inteligencia artificial podrían ser comprometidos.

Es esencial, por lo tanto, poder comprender las diferencias entre los diversos modelos y sus aplicaciones prácticas. Al conocer sus fortalezas y limitaciones, los desolladores pueden tomar decisiones informadas sobre qué modelo utilizar en función de los requisitos específicos de los procesos que se pretendan automatizar. Buscamos, por lo tanto, poder proporcionar una visión completa y actualizada del panorama de modelos de lenguaje e inteligencia artificial, y como su aplicación en la automatización puede ser una herramienta poderosa para mejorar la eficiencia y la productividad en un mundo cada vez más impulsado por la tecnología.

A continuación, se hace una enumeración de los objetivos que se pretende cumplir para ejecutar correctamente el proyecto y que permitirán garantizar una absoluta comprensión de todo lo realizado y la consecución del fin que se persigue con el proyecto:

- Realizar una revisión exhaustiva de los LLMs disponibles en la literatura, teniendo en cuenta sus propiedades, ventajas y desventajas. Además, se reflexionará sobre las tendencias en estos modelos para poder tener una visión de cómo podrían desarrollarse en los próximos años. Se prestará atención a aspectos técnicos como la eficiencia computacional, los tiempos de respuesta y la posibilidad de ejecutar los modelos en local. También se tendrán en cuenta las barreras legales y de privacidad, así como los marcos éticos asociados al uso de estos modelos.
- Realizar un análisis de las posibles tareas de back office en empresas industriales que puedan ser susceptibles de ser automatizadas mediante LLMs. Se identificarán las tareas más relevantes, se explicará cómo se podrían llevar a cabo y se analizarán las posibles repercusiones de su automatización.
- Cruzar los resultados de la revisión de modelos y del análisis de tareas para ver como de eficientes serian actualmente esos los modelos realizando estas tareas. Se propondrá un caso concreto de aplicación y se desarrollará una pequeña demo. Se tendrá en cuenta la elección del modelo más adecuado para el caso concreto, teniendo en cuenta las consideraciones técnicas, legales y éticas mencionadas anteriormente. También se identificarán los requisitos para la implementación del caso de aplicación propuesto, como el hardware y software necesario y el costo asociado.

2. DESCRIPCIÓN DE LAS TECNOLOGÍAS

El *Deep Learning* y el *Machine Learnin*g son dos tecnologías fundamentales que han impulsado el desarrollo de la inteligencia artificial. En el *Machine Learning*, se crean algoritmos que permiten a los sistemas informáticos aprender de los datos, hacer predicciones y tomar decisiones sin ser programados explícitamente. Al analizar patrones y extraer información de grandes conjuntos de datos, estos algoritmos pueden mejorar su rendimiento de manera automática a lo largo del tiempo. [4]

El *Deep Learning* es una parte del *Machine Learning* que se enfoca en entrenar redes neuronales artificiales con múltiples capas para aprender automáticamente de datos complicados. Usando grandes conjuntos de datos, los algoritmos de *Deep Learning* pueden descubrir patrones complejos y hacer predicciones o clasificaciones precisas. Esto ha causado una revolución en áreas como la robótica, el reconocimiento de voz y sistemas generación de texto. [5]

La evolución de las tecnologías de IA condujo a la aparición del procesamiento del lenguaje natural (PLN), que implica la interacción entre los ordenadores y el lenguaje humano. Inicialmente estaban basados en sistemas que seguían reglas, pero el PLN progresó con la llegada de los métodos estadísticos y el *Machine Learning*, en particular con las técnicas de *Deep Learning*. Este progreso permitió a las máquinas comprender, analizar y generar textos en lenguaje humano. Los algoritmos de PLN pueden realizar tareas como el análisis de sentimientos, la extracción de información, la traducción automática y los sistemas de respuesta a preguntas, lo que los convierte en un componente crucial en las aplicaciones de IA que implican la comprensión del lenguaje.

Uno de los avances transformadores en el campo de la PLN es el auge de los *large language models* (LLMs). Estos modelos aprovechan las técnicas de *Deep Learning*, en particular las arquitecturas de modelo *transformer*, para procesar y generar texto similar al humano a una escala sin precedentes. Al entrenarse con grandes cantidades de datos de texto, estos modelos

adquieren una comprensión contextual y la capacidad de generar respuestas coherentes y contextualmente relevantes. Los LLMs han recibido una gran atención recientemente debido a sus aplicaciones en *chatbots*, asistentes virtuales, generación de contenidos y traducción de idiomas. Han hecho avanzar significativamente las capacidades de comprensión y generación de lenguaje natural, permitiendo interacciones más sofisticadas entre humanos y ordenadores. [6]

Como ya hemos mencionado, la evolución desde, el *Deep Learnin*g y el *Machine Learnin*g hasta, el PNL y los LLMs ha impulsado las tecnologías de IA en el ámbito de la comprensión del lenguaje natural. Estos avances han facilitado el desarrollo de sistemas capaces de comprender y generar lenguaje con una precisión y sofisticación cada vez mayores. La investigación y el desarrollo en curso en este campo encierran un inmenso potencial para potenciar aún más los sistemas de IA en la comprensión y el procesamiento del lenguaje natural en diversas aplicaciones del mundo real. A pesar de sus impresionantes capacidades, los LLM se enfrentan a retos relacionados con la parcialidad, la ética y la seguridad. Los investigadores y desarrolladores están explorando activamente formas de abordar estos problemas y garantizar el uso responsable de los LLM en aplicaciones del mundo real.

De cara al futuro, continúa el desarrollo de LLMs aún más grandes y sofisticados. Estos modelos se esfuerzan por mejorar no sólo en términos de comprensión y generación del lenguaje, sino también en su capacidad para razonar, inferir y realizar tareas más complejas. A medida que avanzan las tecnologías de los LLMs, tienen el potencial de revolucionar las interacciones entre humanos y ordenadores. [7]

Con todos estos avances surge la posibilidad de utilizar estos modelos para la realización de tareas rutinarias. Empiezan a surgir aplicaciones impulsadas por LLMs que realizan tareas como resumir textos, clasificar diálogos, traducciones, etcétera. El siguiente paso en esta evolución se convirtió en el proceso de automatizar todas estas actividades y aplicaciones, lo que dio lugar al *Robotic Process Automation*. [8]

2.1 ROBOTIC PROCESS AUTOMATION Y LA AUTOMATIZACIÓN INTELIGENTE DE PROCESOS

Robotic Process Automation (RPA) consiste en sistemas que permiten automatizar tareas muy repetitivas que normalmente representan tareas de escaso valor añadido para las personas. En muchos sectores, estos sistemas suelen emplearse para automatizar tareas de *back office*, donde pueden aportar un alto retorno a la inversión. Además, son especialmente útiles en casos de uso que implican sistemas muy dispares y poco integrados entre sí, donde pueden utilizarse para eliminar la necesidad de intervención humana en cada paso del flujo de trabajo y sustituirla por un operador automático, que se limita a simular exactamente lo que hace el ser humano, pero de forma automatizada. El resultado es un flujo de trabajo más eficiente que requiere menos intervenciones humanas, lo que se traduce en un menor riesgo de errores manuales y una reducción de los costes operativos.

Algunos ejemplos de tareas que pueden automatizarse mediante RPA son:

- Crear documentos de Excel/Word.
- Exportar datos de una aplicación de oficina a otra.
- Dar formato a los datos.
- Generar tablas y diagramas.
- Ejecutar macros de MS Excel o códigos.
- Guardar un archivo en una carpeta específica.
- Abrir correos electrónicos y exportar los archivos adjuntos.
- Enviar correos electrónicos.
- Iniciar sesión en una aplicación web introduciendo las credenciales a través de la interfaz web.
- Y muchos más

La automatización de las tareas a menudo consiste simplemente en simular mediante un agente de software (*bot*) la interacción del usuario para lanzar el proceso, pasando como

parámetros de entrada los resultados de la tarea anterior. Como tal, la automatización pura mediante RPA es sobre todo un sistema basado en reglas que no requiere IA avanzada.

Sin embargo, hoy en día la RPA ha evolucionado hacia la Automatización Inteligente de Procesos (IPA), combinando funcionalidades de IA y aprendizaje automático, como la PLN y el análisis de textos. Por ejemplo, los modelos basados en LLM pueden analizar un documento escaneado y clasificarlo automáticamente según su categoría, lo que permite automatizar partes enteras de procesos de *middle* y *back office*.

Es importante señalar que la RPA puede automatizar algunos pasos manuales, pero no debe utilizarse para sustituir por una máquina los pasos críticos que requieren validación humana. Además, deben aplicarse controles de seguridad estándar para restringir y asegurar el acceso a las credenciales utilizadas por el RPA para lanzar los distintos pasos del flujo de trabajo, y garantizar un seguimiento y una auditabilidad adecuados de todas las acciones. Las acciones realizadas por los *bots* deben estar claramente marcadas como tales en el registro de auditoría.

Por lo general, la RPA se presenta en forma de ofertas predefinidas de proveedores que proporcionan la herramienta y la experiencia para su implantación. Empresas como *UI Path* ayudan a sus clientes en todas las fases del proceso de RPA; analizando, explicando e implantando los procesos.[9] Sin embargo, no debe subestimarse la dependencia de los proveedores externos para el mantenimiento del RPA y la institución también debe evaluar la necesidad de formar al personal interno para el mantenimiento continuo y la evolución del proceso de RPA.

Por último, es importante señalar que, aunque la RPA puede mejorar mucho un proceso operativo al automatizarlo y, por tanto, hacerlo más rápido y menos propenso a errores manuales, estas tecnologías no mejoran el proceso en sí. A veces, se podría ganar más eficiencia mediante la reingeniería de todo el proceso, es decir, revisando todas las fases e interacciones que componen el proceso e implementando un STP (*Straight Through Processing*). [10]

Según estimaciones, alrededor del 47% del empleo total de EE.UU. se encuentra en la categoría de alto riesgo. Se denominan empleos de riesgo a aquellos que se prevé que podrían automatizarse relativamente pronto, quizá en la próxima década o dos. Se estima que la mayoría de los trabajadores de los sectores del transporte y la logística, junto con el grueso de los trabajadores de apoyo administrativo y de oficina, y la mano de obra de los sectores productivos, están en riesgo. [11]

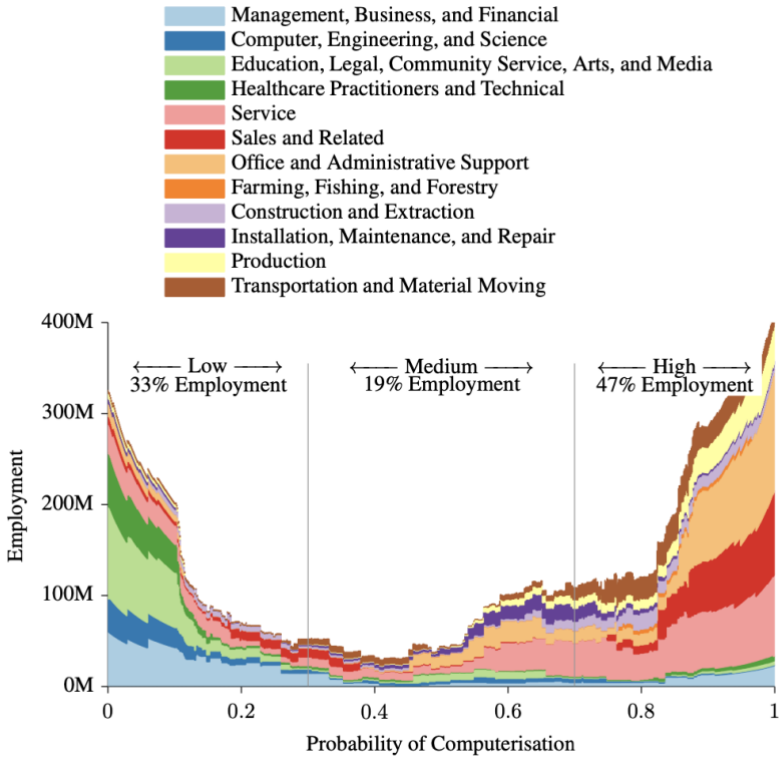

Figura 2. Grafica comparativa entre empleo y probabilidad de automatización. [11]

Se puede observar en la gráfica ese 47% de los empleos que tienen una probabilidad de automatización por encima del 70%. Estos resultados son coherentes con los recientes avances tecnológicos documentados en la literatura. Los rápidos avances de la IA implican que los trabajadores poco cualificados se reasignarán a tareas que no sean susceptibles de informatización, es decir, tareas que requieran inteligencia creativa y social.

3. ESTADO DE LA CUESTIÓN

Los LLMs han ido evolucionando progresivamente a lo largo de los años, y no fue hasta 2017, con la publicación del *paper*: "*Attention is all you need*" [12], donde se marcó un hito en la historia de la tecnología de procesamiento de lenguaje natural. Fue publicado por un equipo de investigadores de Google, y mostraron por primera vez la arquitectura *Transformer* para LLMs. La aparición de los Transformers supuso un gran salto con respecto a las tecnologías previas (redes neuronales recurrentes con memoria corto plazo y largo plazo (LSTMs), se eliminó por ejemplo la recurrencia o se minimizo la distancia entre elementos de las secuencias de las redes.

Aparecerían en 2018 modelos como BERT [13] y GPT-1 [14], utilizando esta arquitectura. Desde entonces, los LLMs han experimentado una rápida evolución, con el lanzamiento de un gran número de modelos de lenguaje desarrollados por una gran variedad de empresas.

Las capacidades de memorización combinadas con la versatilidad hacen que los LLMs sean capaces de ejecutar varias tareas, como la comprensión del lenguaje o la generación de texto condicional y no condicional, a un nivel de rendimiento muy avanzado, dando paso a una interacción con las personas muy avanzada.

A pesar de estos avances, los LLMs siguen teniendo limitaciones que impiden su despliegue más amplio. Siguen cometiendo errores y pueden incluir tendencias a ciertos sesgos que vengan derivadas de los datos utilizados para entrenarlos. Por ello se sigue investigando para abordar sus limitaciones.

Se están explorando nuevas técnicas para su aprendizaje exprimiendo las técnicas de *Deep Learning*, como la utilización de *unsupervised learning* y aprendizaje colaborativo. Con ellas se pretende mejorar la precisión y la eficiencia de los LLM. Además, se están investigando enfoques para el diseño de conjuntos de datos de entrenamiento que reduzcan los sesgos y prejuicios.

Actualmente hay una gran variedad de modelos de lenguaje disponibles. El más conocido es GPT-3 de *OpenAI* [15], evolución de GPT-1 de 2018, también hay otros LLMs importantes en el mercado, como T5 de Google [16], Turing-NLG de Microsoft [17] y ELMo del *Allen insitute for articial inteligence* [18]. Estos modelos tienen sus propias fortalezas y debilidades, y cada uno está diseñado para abordar diferentes problemas en el procesamiento del lenguaje natural

Se podría decir que los desarrollos de estos modelos se han realizado con un enfoque para especializarlos en un área determinada. T5 de Google es especialmente conocido por su capacidad para realizar tareas de comprensión del lenguaje en múltiples idiomas, mientras que Turing-NLG de *Microsoft* ha demostrado ser excepcionalmente efectivo en la creación de diálogos coherentes y en el razonamiento lógico.

Las posibilidades son cada vez mayores en cuanto a lo que se puede lograr en términos de automatización y mejora de la eficiencia en el procesamiento del lenguaje natural.

4. MODELOS Y CARACTERÍSTICAS ANALIZADOS

Como hemos podido comprobar ha habido una gran evolución en el desarrollo de modelos de lenguaje. Podríamos resumir la gran mayoría de los LLMs en dos categorías. *Encoder-Decoder* [19]y *Decoder only*[20].

Ambos tipos de arquitectura se diseñaron intentando resolver el mismo problema desde dos ángulos distintos. Para determinadas tareas realizan un trabajo tan eficiente que es difícil determinar qué tipo de arquitectura es mejor.

Tabla 1. Tabla resumen de los LLMs

	Características	**Ejemplos de LLMs**
Encoder-Decoder o solo Encoder	Tipo de entrenamiento: *Masked Language Models* Estilo del modelo: Discriminativo Tarea: Predecir la palabra o *token* encubierta en la frase	ELMo [18], BERT [13], RoBERTa [21], DistilBERT [22], BioBERT [23], XLM [24], XLNET [25], ALBERT [26], ELECTRA [27], T5 [16], GLM [28], XLM-E [29], ST-MoE [29], AlexaTM [30].
Decoder only	Tipo de entrenamiento: *Autoregressive Language Models* Tipo de modelo: Generativo Tarea: Predecir la próxima palabra o *token* de la frase.	GPT-3 [15] OPT [31], PaLM [32], BLOOM [33], MT-NLG [17], GLaM [34], Gopher [35], Chinchilla [36], LaMDA [37], LLaMA [38], GPT-4 [39], BloombergGPT [40].

Antes de pasar a seleccionar que modelos se han considerado los más relevantes para analizar se analizarán las principales características que se pueden utilizar para definir y comparar los LLMs que se seleccionaran. Con ellas se tendrá un mayor conocimiento de los modelos para posteriormente compararlos.

4.1 CARACTERÍSTICAS ANALIZADAS

Estas han sido las 13 principales características que se ha decidido analizar y que permiten por lo tanto definir todo LLMs.

1. Desarollador
2. Año
3. Tipo de Modelo
4. Tamaño del Modelo
5. Como ha sido entrenado
6. Usos y Tareas para los que se desarrollo
7. Precision en dichas tareas
8. APIs que ofrece
9. Localizacion del sistema y donde se puede ejecutar
10. Privacidad de los datos
11. Propiedad intelectual
12. Trasparencia del desarrollador y del modelo
13. Regulación específica

Desarrollador

Cuando comparamos varios modelos de lenguaje, es de vital importancia conocer al desarrollador o entidad detrás de cada uno debido a diversas razones cruciales. A partir de quien lo desarrolla vamos a ser capaces de extraer el resto de las características delo modelo. Además, la reputación y prestigio del desarrollador influyen en la credibilidad y confianza que podemos tener en el modelo. Desarrolladores reconocidos suelen seguir prácticas sólidas

de desarrollo y ética, lo que aumenta la confianza en la calidad y seguridad del modelo. Serán los propios desarrolladores los que nos permitan acceder a detalles importantes como la documentación técnica, el código fuente y los datos utilizados para entrenar el modelo, lo que nos ayuda a evaluar su transparencia y adecuación para casos específicos de uso.

A su vez, conocer al desarrollador proporciona información valiosa sobre la ética y responsabilidad asociada al modelo, como las medidas para evitar sesgos y cumplir con regulaciones relevantes. Por lo tanto, saber quién desarrolló el modelo es esencial para tomar decisiones informadas y responsables al utilizar inteligencia artificial y procesamiento del lenguaje natural.

Año

Es importante conocer el año en que se desarrolló un modelo de lenguaje por varias razones. En primer lugar, los avances tecnológicos y mejoras en inteligencia artificial hacen que los modelos más recientes tengan un rendimiento y resultados más precisos. Además, la actualidad del modelo es relevante para garantizar que se base en datos y conocimientos actualizados para abordar problemas actuales de manera efectiva.

Otro aspecto clave es la posibilidad de correcciones y mejoras. Los modelos más nuevos pueden haber abordado errores conocidos y realizado actualizaciones para ser más confiables. Asimismo, las leyes y regulaciones sobre el uso de datos y privacidad pueden cambiar con el tiempo, y un modelo más antiguo puede no estar alineado con las últimas normativas, lo que podría tener implicaciones legales o éticas.

Conocer el año de desarrollo es esencial para comparar modelos y comprender las diferencias de rendimiento. Los modelos más recientes pueden haber superado limitaciones y ofrecer resultados más prometedores en comparación con modelos más antiguos.

Tipo de Modelo

Es esencial tener en cuenta el tipo de modelo y la arquitectura utilizada, ya que estos aspectos desempeñan un papel crucial en el rendimiento y las capacidades del modelo. Cada tipo de

modelo, como *Transformers* [12], *Recurring Neural Networks* (RNN) o *Long Short-Term Memory* (LSTM) [41], entre otros, se han desarrollado con enfoques distintos para procesar el lenguaje natural.

Cada tipo de modelo tiene sus propias fortalezas y debilidades. Por ejemplo, los Transformers han demostrado un excelente rendimiento en tareas de procesamiento de lenguaje, gracias a su capacidad para capturar relaciones complejas y patrones en el texto.[12] Por otro lado, las RNNs son conocidas por manejar eficientemente secuencias largas y mantener la memoria a largo plazo, lo que las hace útiles en tareas donde el contexto histórico es relevante. [42]

La elección de la arquitectura también es un factor determinante. Algunas arquitecturas pueden ser más efectivas para tareas específicas, como la traducción automática o la generación de texto creativo. Por ejemplo, los modelos de lenguaje pre-entrenados como han demostrado excelentes resultados en la generación de texto, mientras que los modelos basados en LSTM se han destacado en tareas de procesamiento de lenguaje estructurado.

Además, la arquitectura también afecta la eficiencia computacional y los requisitos de recursos. Algunas arquitecturas pueden requerir más tiempo de entrenamiento, lo que puede ser un factor crítico a considerar, especialmente en aplicaciones con recursos limitados o en tiempo real. [43]

Otro aspecto importante es la disponibilidad de recursos adicionales. Algunos tipos de modelos o arquitecturas pueden estar más respaldados por la comunidad de investigación y desarrollo, lo que puede resultar en una mayor cantidad de recursos, tutoriales y herramientas disponibles para su implementación y uso.

Conocer el tipo de modelo y la arquitectura afecta directamente al rendimiento, la adaptabilidad a diferentes tareas, la eficiencia computacional y la disponibilidad de recursos adicionales.

Tamaño del modelo

La cantidad de parámetros que componen los modelos determinan su tamaño. Los tres tipos principales de parámetros que contribuyen al tamaño de los modelos de lenguaje son:

1. **Número de Capas (L)**: La arquitectura de muchos modelos de lenguaje está compuesta por múltiples capas apiladas. Cada capa contiene conjuntos de pesos que se utilizan para aprender representaciones progresivamente más complejas del lenguaje. A más capas, más grande será por lo tanto el modelo.
2. **Dimensión Oculta (H)**: La dimensión oculta hace referencia al tamaño de capa del modelo. Cuanto mayor es la dimensión oculta, más capacidad tiene el modelo para capturar características y relaciones complejas en los datos de entrada. Un mayor valor de H aumenta la cantidad de parámetros del modelo.
3. **Número de Cabezas de Atención (A)**: Las cabezas de atención son una parte esencial de los modelos, se utilizan para capturar diferentes patrones y relaciones en el texto. Permiten que el modelo se enfoque en diferentes aspectos de la entrada, mejorando el procesamiento del lenguaje natural. Cada cabeza de atención tiene su propio conjunto de parámetros. Aumentar el número de cabezas de atención suele mejorar el rendimiento del modelo.

Es importante tener en cuenta el tamaño del modelo debido a que afecta directamente a diversos aspectos.

El rendimiento y la eficiencia se ven afectados ya que los modelos más grandes tienden a tener un mejor rendimiento en tareas complejas de procesamiento del lenguaje, porque pueden capturar más información y patrones. Sin embargo, modelos más grandes también requieren más recursos computacionales para su entrenamiento e inferencia, lo que puede ser un factor limitante en entornos con recursos limitados.

Hay que saber buscar el equilibrio entre la generalización y sobreajuste. Un modelo grande suele sobre ajustarse a los datos de entrenamiento, lo que podría afectar su habilidad para generalizar a datos nuevos o desconocidos. En ciertos casos, modelos más pequeños y menos complejos pueden generalizar mejor.

Y por último el tamaño de los modelos suele afectar considerablemente a el coste y mantenimiento. Los modelos más grandes suelen requerir más almacenamiento y capacidad computacional, lo que implica costes adicionales.

Como ha sido entrenado

Una característica muy importante de los modelos es como este ha sido entrenado. Ya que el proceso de entrenamiento puede afectar significativamente el rendimiento y la calidad del modelo de lenguaje. Un modelo que ha sido entrenado con una cantidad suficiente y diversa de datos puede tener un mejor desempeño en una amplia variedad de tareas de procesamiento del lenguaje natural.

El entrenamiento va a determinar cómo representa el modelo el lenguaje. Un modelo bien entrenado debería ser capaz de capturar relaciones complejas y patrones en el lenguaje, lo que afecta su habilidad para comprender y generar texto coherente y significativo.

Asimismo, la calidad y diversidad de los datos de entrenamiento pueden afectar la presencia de sesgos y discriminación. Un entrenamiento cuidadoso y una selección adecuada de datos pueden ayudar a mitigar estos problemas y producir modelos más justos y equitativos.

Se puede por lo tanto analizar el rendimiento de un modelo basándonos en parte en cómo ha sido entrenado.

Usos y Tareas para los que se desarrolló

Para tener una mejor visión de como de eficaz es un modelo debemos también analizar para que tareas se pensó originariamente que se podría utilizar el modelo.

Podríamos, por lo tanto, evaluar su adecuación a la tarea que nosotros queremos abordar. Además, comprender sus usos previstos permite optimizar la selección de modelos para cualquier aplicación, al ser capaces de identificar los modelos líderes en un campo específico y detectar posibles limitaciones en tareas fuera de su dominio.

Precisión en dichas tareas

La precisión va a permitir comprender la efectividad y eficiencia de cada modelo en abordar esas tareas con resultados confiables. Al conocer la precisión de los modelos, podemos tomar decisiones informadas y seleccionar el más adecuado para nuestras necesidades.

La precisión también proporciona una visión clara de las fortalezas y debilidades en los diferentes contextos. Esto ayuda a entender las fortalezas y debilidades que pueden llegar a tener. Se pueden aprovechar las ventajas de cada modelo y utilizarlos de manera más efectiva en aplicaciones prácticas.

APIs que ofrece

Las Interfaces de Programación de Aplicaciones (APIs) que tienen los desabolladores para sus modelos son esenciales. Cada modelo puede tener diferentes APIs con funcionalidades específicas, lo que afecta su facilidad de integración, la disponibilidad de herramientas de desarrollo y la flexibilidad para adaptarse a nuestras necesidades.

Evaluar las APIs de los modelos permite identificar qué funcionalidades y capacidades están disponibles para la interacción con el modelo. Podemos determinar si el modelo es compatible con nuestras aplicaciones y si ofrece las herramientas necesarias para obtener los resultados deseados. Por lo tanto, revisar las APIs que ofrece cada uno nos permite seleccionar el modelo más adecuado para nuestras aplicaciones y garantizar una integración exitosa y eficiente.

Localización del Sistema y Donde se puede ejecutar

Es importante tener en cuenta si los modelos son públicos o privados, ya que esto afecta a dónde y cómo se pueden ejecutar. Los modelos de lenguaje públicos suelen estar disponibles para su descarga y uso en cualquier ordenador, lo que brinda la flexibilidad de ejecutarlos localmente en nuestros propios ordenadores. Esto puede ser beneficioso para aplicaciones que requieran privacidad, autonomía o para evitar la dependencia de servicios externos.

Por otro lado, si los modelos son privados o de acceso limitado, es probable que estén alojados en los sistemas del desarrollador o en plataformas específicas. En este caso, se requeriría acceso a los servidores o infraestructura del desarrollador para ejecutar el modelo. Esto puede implicar la necesidad de comunicarse con el proveedor del modelo para solicitar permisos de uso o adquirir licencias para acceder a su funcionalidad. Aunque la privacidad de nuestros datos pueda estar afectada por lo general son infraestructuras más fáciles de escalar lo cual puede ser altamente beneficioso.

Privacidad de los datos

Es esencial analizar si el desarrollador hace uso de nuestros datos, ya que la privacidad de los datos es un aspecto muy relevante en el contexto actual de la tecnología. Será importante saber qué tipo de datos se están recopilando, cómo se utilizan y si existe algún riesgo de que nuestra información personal sea comprometida o utilizada de manera inapropiada.

La privacidad de nuestros datos es un derecho fundamental que debe ser respetado. Al permitir que un desarrollador acceda y utilice nuestros datos, estamos confiando en que estos serán tratados de manera adecuada y segura. Hay que analizar por lo tanto que el desarrollador tenga políticas claras de privacidad y protección de datos, y que esté cumpliendo con las regulaciones y leyes aplicables.

Además, la privacidad de nuestros datos también puede tener implicaciones éticas. Al utilizar modelos de lenguaje entrenados con datos sensibles o personales, podríamos estar contribuyendo a la propagación de sesgos o discriminación. Por lo tanto, es crucial evaluar si el desarrollador tiene medidas para abordar estos problemas y garantizar un uso ético de los datos.

Propiedad Intelectual

La propiedad intelectual protege los derechos legales de los desarrolladores sobre sus creaciones, como algoritmos, arquitecturas y datos, fomentando la innovación y la inversión

en nuevas tecnologías. Al garantizar que los creadores tengan derechos exclusivos sobre sus modelos, se promueve el intercambio de conocimientos y el avance.

La propiedad intelectual también puede afectar el acceso y la distribución de los modelos de lenguaje. Algunos modelos pueden estar sujetos a licencias y restricciones, lo que puede influir en su disponibilidad y uso.

Trasparencia del desarrollador y del modelo

La transparencia del desarrollador nos permite entender cómo se construyó el modelo y qué datos se utilizaron para su entrenamiento. Esto es esencial para evaluar la calidad y la validez del modelo y asegurarnos de que esté bien fundamentado y libre de sesgos. Además, la transparencia nos proporciona información sobre las limitaciones y las áreas en las que el modelo puede no ser adecuado.

La transparencia también es crucial para la confianza y la credibilidad. Cuando el desarrollador proporciona información detallada y accesible sobre el modelo, sus métodos y resultados, se fomenta la confianza de los usuarios y la comunidad en la aplicación del modelo.

Regulación especifica

Es importante analizar si los modelos tienen regulación específica porque diferentes regiones y países pueden tener leyes y normativas que afecten el uso y la aplicación de modelos de lenguaje. Al conocer la regulación específica aplicable a un modelo, podemos garantizar su cumplimiento y evitar posibles consecuencias legales o éticas.

Además de las regulaciones específicas, también es importante tener en cuenta las regulaciones globales, ya que el uso de modelos de lenguaje puede trascender fronteras y afectar a usuarios y comunidades en diferentes países. Considerar las regulaciones globales nos permite adoptar un enfoque más amplio y asegurarnos de que el modelo sea aplicable y seguro en una escala internacional. Una de ellas que afecta a todos ellos es la Regulación para la IA de la unión europea. [44]

4.2 MODELOS A ANALIZAR

Con las características ya definidas se pasa al análisis de los modelos. Primero de todo seleccionamos los modelos que han resultado más interesantes para analizar. Se han seleccionado un total de 16 modelos. Las razones detrás de la selección de estos son la relevancia que tienen actualmente, sus tamaños, sus desarrolladores. De esta manera se pretende tener diversidad con la selección.

1. BERT [13]
2. T5 [16]
3. ELECTRA[27]
4. REFORMER [45]
5. GPT-4 [15], [39]
6. RoBERTa [21]
7. LLAMA [46]
8. ELMO [18]
9. XLNET [25]
10. DistilBERT [22]
11. UniLM [47]
12. Megatron-Turing NLG [17]
13. LaMDa [37]
14. Jurassic-1 [48]
15. Gopher [35]
16. Chinchilla [49]

4.3 TABLA RESUMEN DE CARACTERÍSTICAS Y MODELOS

A continuación, se muestra la tabla resumen de todos los modelos analizados y sus características.

Tabla 2. Tabla Resumen análisis de modelos

	Modelo	Desarrollador	Año	Tipo de Modelo	Descripción	Tamaño del Modelo	Cómo se ha Entrenado	Usos y Tareas que puede procesar	Precisión	APIs que ofrece	Localización del Sistema/ Donde se puede ejecutar	Privacidad de los datos	Propiedad Intelectual	Transparencia	Regulación especifica
1	**BERT**	Google AI	2017	Bidirectional Transformer	-	Existen dos tamaños principales de modelos BERT, el BERT $_{BASE}$ (L=12, H=768, A=12, Numero Total de Parameteros=110M) y BERT LARGE (L=24, H=1024, A=16, Numero Total de Parametros=340M).	Para entrenar estos modelos bidireccionales se ocultan un porcentaje de los datos de entrada de manera aleatoria y posteriormente se intentará predecir esos datos. Se ocultan el 15% de los datos. Los datos de entrnamiento están compuestos por dos sets de datos, BookCorpus (800M de palabras) y los pasajes de texto de la Wikipedia en inglés (2.500M de palabras).	Las 4 tareas principales son: Tareas de clasificación de parejas de frases, tareas de clasificación de frases individuales, tareas de responder preguntas y tareas de etiquetado de frases.	90%	Adhoc	Open Source	Privados	Derechos de autor propios de google ai	Total	Ninguna
2	**T5**	Google AI	2020	Encoder - Decoder Transformer	-	Las redes feed-forward de cada bloque constan de una capa densa con una dimensionalidad de salida seguida de una función de activación no linear ReLU y otra capa densa. Todos los mecanismos de atención tienen una dimensionalidad interna de 64 y todos los mecanismos de atención tienen 12 cabezas. Todas las demás subcapas e incrustaciones tienen una dimensionalidad de 768. En total, esto da como resultado un modelo con unos 220 millones de parámetros. Esto es aproximadamente el doble del número de parámetros de BERT BASE, ya que nuestro modelo base contiene dos capas en lugar de una. Para la regularización, utilizamos una probabilidad de abandono de 0,1 en todos los lugares en los que se aplica el abandono en el modelo.	Se utilizan grandes conjuntos de datos no etiquetados para el aprendizaje no supervisado "Colossal Clean Crawled Corpus" (C4), un conjunto de datos formado por cientos de gigabytes de texto. Preentrenamos cada modelo durante 219 en C4 antes de afinarlo. Utilizamos una longitud máxima de secuencia de 512 y un tamaño de lote de 128 secuencias. Siempre que es posible, "empaquetamos" varias secuencias en cada entrada del lote10. En total, este tamaño de lote y número de pasos corresponde a un preentrenamiento en 235 ≈ 34B tokens. Esto es considerablemente menos que BERT, que utilizó aproximadamente 137B tokens, o RoBERTa, que utilizó aproximadamente 2,2T tokens. El uso de solo 235 tokens resulta en un presupuesto computacional razonable al tiempo que proporciona una cantidad suficiente de pre-entrenamiento para un rendimiento aceptable. Se realiza también ajuste fino para tareas específicas.	Respuesta a preguntas, resumen de documentos y clasificación de sentimientos, por nombrar algunas.	86%	Adhoc	Open Source	Privados	Derechos de autor propios de google ai	Total	Ninguna
3	**ELECTRA**	Google AI	2020	Two Transformers models	Lenguaje enmascarado modificado. En lugar de enmascarar la entrada, su método la modifica sustituyendo algunos tokens por alternativas probables extraídas de una pequeña red de generación. Entonces, en lugar de entrenar un modelo que prediga las identidades originales de los tokens modificados, entrenamos un modelo discriminatorio que predice si cada token de la entrada modificada ha sido sustituido por una muestra de un generador o no.	Existen varios tamaños del modelo, el más pequeño; L=12, H=256, A=64; diseñado con la intención de ser corrido por una sola GPU. Y el modelo grande; L=24, H=1024, A=64.	Se realizaron una seria de modificaciones al modelo Base para su entrenamiento. Se propuso mejorar la eficacia del preentrenamiento compartiendo los pesos entre el generador y el discriminador. Se implementa un generador más pequeño y se utilizan las incrustaciones del tamaño de los estados ocultos del discriminador. Las incrustaciones de token de "entrada" y "salida" del generador siempre están vinculadas, como en BERT. Al utilizar un generador más pequeño se reducen los tiempos de cálculo, se consigue este generador más pequeño disminuyendo el tamaño de las capas.	Las 4 tareas principales son: Tareas de clasificación de parejas de frases, tareas de clasificación de frases individuales, tareas de responder preguntas y tareas de etiquetado de frases.	91%	Adhoc	Open Source	Privados	Derechos de autor propios de google ai	Total	Ninguna
4	**Reformer**	Google AI	2020	Large Transformer	Los modelos Transformer de gran tamaño suelen obtener los mejores resultados en una serie de tareas, pero su entrenamiento puede resultar muy costoso, sobre todo en secuencias largas. Se han introducido dos técnicas para mejorar la eficiencia. Las capas reversibles, permiten almacenar una sola copia de las funciones de activación en todo el modelo, por lo que el factor N desaparece. Se dividen las activaciones dentro de las secuencias feed-forward para ahorrar memoria. El cálculo aproximado de los puntos basado en hashing sensible a la localidad lo que permite trabajar con secuencias largas. El modelo resultante, el Reformer, tiene un rendimiento similar al de los modelos Transformer, pero es mucho más eficiente en memoria y mucho más rápido en secuencias largas.	En todos los experimentos se utilizó un modelo L = 1024, H = 4096, A = 8 y un tamaño total de lote de 8 secuencias. Utilizamos el optimizador Adafactor para entrenar estos modelos.	Hemos realizado nuestros experimentos en imagenet64 y enwik8-64K. Utilizamos modelos de 3 capas para nuestras depuraciones con el fin de poder compararlo con el Transformer normal, que tiene un alto consumo de memoria.	La capacidad de manejar secuencias largas abre el camino para el uso del Reformer en muchas tareas generativas. Además de generar textos coherentes muy largos, el Reformer puede llevar la potencia de los modelos Transformer a otros dominios como la previsión de series temporales o la generación de música, imágenes y vídeos.		Adhoc	Open Source	Privados	Derechos de autor propios de google ai	Total	Ninguna

5	**GPT- 4**	Open Ai	2023	Casual Unidirac tional Transfor mer, large multimo dal model	Desarollado a partir de GPT-3 pero permite como entrada tanto texto como imágenes.	Al igual que el modelo GPT-3 mad grande el tamaño maximo de GPT-4 es: L = 96, H = y A = 12288.	Al igual que para GPT-3. Se utilizo CommonCrawl con modificaciones para el entrenamiento: 1. Se filtró una versión de CommonCrawl basada en la similitud con una serie de corpus de referencia de alta calidad. 2. Se realizó una deduplicación difusa a nivel de documento para evitar la redundancia y preservar la integridad de nuestro conjunto 3. Se añadió un corpus de referencia de alta calidad a la mezcla de entrenamiento para aumentar CommonCrawl e incrementar su diversidad. Se añadieron varios conjuntos de datos curados de alta calidad, incluida una versión ampliada del conjunto de datos WebText. Los datos de CommonCrawl constituyen 45 TB de texto plano comprimido antes del filtrado y 570 GB después del filtrado, lo que equivale aproximadamente a 400.000 millones de tokens codificados por pares de bytes. Los modelos más grandes pueden utilizar normalmente un tamaño de lote mayor, pero requieren una tasa de aprendizaje menor. Se midió la escala de ruido de gradiente durante el entrenamiento y se utilizó para guiar nuestra elección del tamaño del lote. Todos los modelos se entrenaron en GPUs V100 que formaban parte de un cluster de gran ancho de banda proporcionado por Microsoft.	Toto tipo de tareas de generacion de texto, actividades de razonamiento, respuesta de preguntas.	96%	Propias de OpenA i como un ChatG PT	Plataform a Open AI	Accesibles por Open AI	Derechos propios de Open AI	Total	Ninguna
6	**RoBERT a**	Facebook AI	2019	A partir de BERT	Diferenciándose en que se entrenó con batches más grandes, no se entrena para predicción de siguiente frase y utiliza unos patrones de ocultación de datos de entrenamiento distintos.	El tamaño y los parámetros son los mismos utilizados en BERT y se desarrollaron también dos modelos, Base y Large. BASE (L=12, H=768, A=12, Numero Total de Parameteros=110M) y LARGE (L=24, H=1024, A=16, Numero Total de Parametros=340M).	Se modifica la estructura de entrenamiento para la ocultación de parámetros de entrada, en BERT se realizaba de manera estática y aleatoria con un 15% de ocultación y con RoBERTa se realiza de manera dinámica. A su vez, se realizaron más pasos en el entrenamiento. RoBERTa mejoró el entrenamiento de BERT utilizando 160 GB de datos de texto, frente a los 16 GB de BERT, eliminó la predicción de la siguiente frase, entrenó con lotes más grandes y aumentó el tamaño del modelo a 355 millones de parámetros, lo que mejoró el rendimiento y la comprensión del lenguaje. Los datos utilizados en el entrenamiento fueron: BookCorpus, la Wikipedia en Ingles, Common Crawl, OpenWebText y Stories. Todos ellos de acceso público.	Las 4 tareas principales son: Tareas de clasificación de parejas de frases, tareas de clasificación de frases individuales, tareas de responder preguntas y tareas de etiquetado de frases.	92%	Adhoc	Open Source	Privados	Derechos de autor propios de meta ai	Total	Ninguna
7	**LLAMA**	Meta AI	2023	Transfor mer con modifica ciones	Prenormalización: En lugar de normalizar la salida de cada capa transformadora, aplicamos la normalización a la entrada. Esto mejora la estabilidad del entrenamiento de nuestro modelo. Función de activación SwiGLU: En vez de utilizar la función de activación ReLU, empleamos la función SwiGLU. Esta función, ha demostrado mejorar el rendimiento del modelo al proporcionar una no linealidad más adecuada para nuestras necesidades. Incrustaciones rotativas: En lugar de utilizar incrustaciones posicionales absolutas, las cuales asignan una posición fija a cada palabra en la secuencia, empleamos incrustaciones posicionales rotativas (RoPE). Las RoPE, asignan una incrustación posicional que varía dependiendo de la posición relativa de las palabras en cada capa de la red. Esto ayuda al modelo a capturar mejor las relaciones entre las palabras en diferentes posiciones de la secuencia.	El modelo LLAMA más grande tiene 65,2 billones de parámetros con 8192 dimensiones, 64 self attention heads y 80 capas.	Datos de carácter publico para el entrenamiento de estos modelos como puede ser Common Crawl, C4, GitHub, Wikipedia, ArXiv.	Las tareas típicas para las que se utiliza ese modelo son tareas de tareas de razonamiento de sentido común, generación de textos, Respuesta a preguntas y generación de código.	Zero shot performanc e en torno a 80%	Adhoc	Open Source	Privados	Derechos de autor propios de meta ai	Total	Ninguna
8	**ELMO**	Allen Institute for Artificial Intelligenc e	2018	Bidirecti onal Transfor er	-	2 capas bidireccionales con 512 dimensiones y 4096 unidades. Conexión residual entre las dos capas	Los modelos biLM preentrenados se han modificado para permitir el entrenamiento conjunto en ambas direcciones y añadir una conexión residual entre las capas LSTM. Después de entrenar durante 10 épocas en el 1B Word Benchmark, el promedio de perplejidades hacia adelante y hacia atrás es de 39,7. En general, encontramos que las perplejidades hacia delante y hacia atrás son aproximadas. Para realizar el ajuste fino en una tarea determinada, se ignoraron temporalmente las variables de supervisión, y se ajustó durante una época en la división de entrenamiento y se evaluó en la división de desarrollo. Una vez ajustados, los pesos biLM se fijaron durante el entrenamiento de la tarea.	Vinculación textual, tarea de determinar si una "hipótesis" es cierta, dada una "premisa". Etiquetado de funciones semánticas Resolución de correferencias, tarea de agrupar referencias en un texto que corresponden a las mismas cosas. Respuesta a preguntas Análisis de sentimiento cinco niveles desde muy negativo a muy positivo	En torno al 85%, salvo analisis de sentimiento s al 55% (por enncima de la media)	El modelo en si mismo es una API para mejora r otros modelo s en las tareas que realiza	Open Source	Privados	Derechos de autor propios de google ai	Total	Ninguna

9	**XLNET**	Google AI	2019	Modelo autorregresivo de lenguaje basando en el Transformer XL.	-	Igual que BERTLARGE (L=24, H=1024, A=16, Numero Total de Parametros=340M).	Data sets: BooksCorpus , Wikipedia en inglés, Giga5 (16 GB de texto), ClueWeb 2012-B y Common Crawl para el preentrenamiento. En total, 32,89 GB de texto. El método de entrenamiento se basa en la arquitectura de este modelo ya que a diferencia de un Transformer normal, utiliza aparte de atención bidireccional, predicción parcial. La atención bidireccional consiste en tener en cuenta tanto la información de la izquierda como la de la derecha del dato oculto que estamos intentando predecir. La predicción parcial, no intenta predecir solo la siguiente palabra de una secuencia, sino que considera todas las combinaciones de palabras dentro de la frase para hacerlo.	Este modelo se utiliza para responder preguntas, comprensión lectora y tareas de relación de frases	92%	Adhoc	Open Source	Privados	Derechos de autor propios de google ai	Total	Ninguna
10	**DistilBERT**	Huggin Face	2020	Modelo simplificado de BERT: más pequeño, rápido, barato y ligero.	Se aprovechó la extracción de conocimientos durante la fase de preentrenamiento y demostramos que es posible reducir el tamaño de un modelo BERT en un 40%, conservando al mismo tiempo el 97% de sus capacidades de comprensión del lenguaje y siendo un 60% más rápido.	40% deBERT	El entrenamiento se basa en el aprendizaje maestro-alumno, una técnica de compresión donde un modelo compacto, DistilBERT actúa como el alumno, se entrena para replicar el comportamiento de un modelo más grande, BERT. DistilBERT conserva la misma estructura general que BERT, pero se realizan modificaciones para reducir su tamaño y complejidad. Se eliminan las incrustaciones de tipo token y el agrupador, y se reduce a la mitad el número de capas. Estas modificaciones se basan en la optimización de las operaciones utilizadas en la arquitectura Transformer, como las capas lineales y la normalización de capas, que están altamente optimizadas en los marcos de álgebra lineal modernos. Las investigaciones han demostrado que las variaciones en la última dimensión del tensor (tamaño oculto) tienen un impacto menor en la eficiencia computacional en comparación con otros factores, como el número de capas, por lo que se enfoca en reducir este último. Además de las consideraciones de optimización y arquitectura, es crucial encontrar una inicialización adecuada para lograr una convergencia óptima. DistilBERT se entrena en lotes de gran tamaño, aprovechando la acumulación de gradientes y utilizando enmascaramiento dinámico. A diferencia de BERT, DistilBERT prescinde del objetivo de predicción de la siguiente oración. El corpus utilizado para entrenar DistilBERT es el mismo que el del modelo BERT original, que consiste en una concatenación de Wikipedia en inglés y el Toronto Book Corpus.	Las 4 tareas principales son: Tareas de clasificación de parejas de frases, tareas de clasificación de frases individuales, tareas de responder preguntas y tareas de etiquetado de frases.	97% de BERT	Adhoc	Open Source	Privados	Derechos de autor propios de google ai	Total	Ninguna
11	**UniLM**	Microsoft	2019	Modelo Transformer multicapa	Pre-entrenado conjuntamente sobre grandes cantidades de texto, optimizado para tres tipos de objetivos de modelado lingüístico no supervisado: predicción unidireccional, bidireccional y de secuencia a secuencia. El modelado unificado se consigue empleando una red Transformer compartida y utilizando máscaras de autoatención específicas para controlar en qué contexto condiciona la predicción.	La arquitectura del modelo de UNILM sigue la de BERTLARGE para una comparación justa. La función de activación gelu se utiliza como en GPT. Específicamente, usamos un Transformador de 24 capas con H = 1.024, y A = 16, que contiene alrededor de 340M de parámetros. La matriz de pesos del clasificador softmax está ligada con token embeddings.	El objetivo general del entrenamiento es la suma de los distintos tipos de objetivos descritos anteriormente. Específicamente, dentro de un lote de entrenamiento, 1/3 del tiempo se utiliza para el objetivo bidireccional, 1/3 para secuencia-a-secuencia, y ambos objetivos izquierda-derecha y derecha-izquierda se muestrean con una tasa de 1/6. UNILM es inicializado como BERTLARGE, y luego pre-entrenado usando Wikipedia en inglés y BookCorpus. El tamaño del vocabulario es de 28.996. La longitud máxima de la secuencia de entrada es 512. La probabilidad de enmascaramiento de tokens es del 15%. En este enmascaramiento el 80% de las veces se elimina el valor de entrada, el 10% se sustituye por un token aleatorio y el otro 10% se mantiene original. Además, el 80% de las veces enmascaramos aleatoriamente un token cada vez, y el 20% de las veces enmascaramos un bigrama o un trigrama.	Utilizado para tareas de comprensión y generación de lenguaje natural	90%	Adhoc	Open Source	Privados	Derechos de autor propios de google ai	Total	Ninguna
12	**MT-NLG 530B**	Microsoft y NVIDIA	2022	Modelo de lenguaje basado en transformadores generativos autorregresivos de izquierda a derecha	Tiene dos componentes clave: Megatron-LM, para el modelado lingüístico a gran escala, y el modelo Turing NLG, para tareas de generación de lenguaje natural.	El número de capas, dimensiones ocultas y cabezas de atención son L = 105, H = 20480 y A = 128, respectivamente. Con 530.000 millones de parámetros	Para compilar nuestro conjunto de datos de entrenamiento, utilizamos un trabajo reciente destinado a recopilar un conjunto de entrenamiento diverso para el modelado del lenguaje, The Pile: Un conjunto de datos de 800 GB de texto diverso para el modelado lingüístico. Además, incluimos RealNews, Common Crawl y CC-Stories, que ya se han utilizado para el preentrenamiento de grandes LM. La longitud de la secuencia es de 2048 y el tamaño global del lote es de 1920. Usamos mil millones de tokens para el aprendizaje lineal. Usamos el decaimiento del coseno para que la tasa de aprendizaje alcance el 10% de su valor en 340.000 millones de tokens. Durante los primeros 12.000 millones de tokens, empezamos con un tamaño de lote de 32 y se fue aumentando gradualmente el tamaño de lote en incrementos de 32, hasta alcanzar el tamaño de lote final de 1920. Se utilizó el optimizador Adam.	Las tareas que se realizan con este modelo son: Predicción de palabras completas, comprensión lectora, razonamiento de sentido común, deducción del lenguaje natural, distinción del sentido de las palabras.	92%	APIs propias de NVIDIA que realizan las tareas para las que ha sido entrenado el modelo	En la plataforma de NVIDIA	Accesibles por NVIDIA	Derechos de autor propios de NVIDIA	Total	Ninguna

13	**LaMDA**	Google AI	2022	LaMDA es una familia de modelos de lenguaje basados en Transformer y especializados en el diálogo	LaMDA utiliza un único modelo para realizar múltiples tareas: genera posibles respuestas, que luego se filtran para comprobar su seguridad, se respaldan en una fuente de conocimiento externo y se vuelven a calificar para encontrar la respuesta de mayor calidad. LaMDA (Language Model for Dialogue Applications) está diseñado específicamente para el diálogo. A diferencia de los modelos lingüísticos tradicionales, que generan texto basándose en el contexto previo, LaMDA se centra en comprender el significado que subyace a la conversación y generar respuestas más relevantes desde el punto de vista contextual. Su objetivo es captar los matices y el contexto de las conversaciones, permitiendo interacciones más interactivas y dinámicas.	El modelo LaMDA más grande tiene 137B de parametros. Utilizamos un modelo de lenguaje Transformer sólo decodificador como arquitectura del modelo para LaMDA. El Transformer tiene L = 64 , H = 8192, A = 128.	LaMDA utiliza una arquitectura Transformer a gran escala y se entrena con una amplia gama de datos de diálogo para mejorar sus capacidades conversacionales. Google ha destacado que LaMDA está diseñado para tener una mayor sensibilidad a las indicaciones y proporcionar respuestas más detalladas en escenarios de diálogo. Para mejorar la calidad, seguridad y fundamentación del modelo se realizaron interacciones diversas entre LaMDA y crowdworkers, según la característica se pidió que se llevaran las conversaciones de determinada manera, y se les pidió que evaluaran las respuestas para posteriormente incluir los resultados dentro del propio modelo. LaMDA se preentrenó para predecir la siguiente palabra en un texto. A diferencia de otros modelos de diálogo entrenados únicamente con datos de diálogo, LaMDA se preentrenó con un conjunto de datos creado a partir de datos de diálogo públicos y otros documentos web públicos. Por lo tanto, LaMDA puede utilizarse como modelo lingüístico general antes de su ajuste. El conjunto de datos de preentrenamiento consta de 2.97B de documentos, 1.12B de diálogos y 13.39B de enunciados de diálogos, con un total de 1.56T de palabras. Más del 90% del conjunto de datos de preentrenamiento está en inglés.	Chatbot	-	Bard, chatbot de google	Online	Accesibles por Google	Derechos de autor propios de google ai	Total	Ninguna
14	**JURASSIC 1**	AI21	2021	Se baso en el módulo decodificador de la arquitectura Transformer con modificaciones	Los tokens de entrada se convierten primero en representación vectorial con una matriz de incrustación y luego se introducen en la red del Transformer	La arquitectura se compone de L Transformer de dimensión oculta H, cada una equipada con un módulo de autoatención atención de tamaño A y un módulo feed-forward. J1-Large (L= 32, H= 4095, A= 128) y J1-Jumbo (L= 76, H= 13824, A= 96).	Para J1-Jumbo diseñamos nuestra arquitectura con la idea de encontrar para un determinado presupuesto de parámetros una profundidad óptima. En concreto, para un presupuesto de parámetros de 175B (sin incluir la matriz de incrustación), la profundidad óptima debería rondar las 80 capas. Nuestro modelo se entrenó con el objetivo de entrenamiento autorregresivo autosupervisado convencional sobre 300B tokens extraídos de recursos disponibles públicamente. En cuanto al procedimiento de optimización se utilizó una tasa de aprendizaje base de 1,2 × 10-4 y 0,6 × 10-4, y un tamaño de lote de 2M y 3,2M de tokens, para J1-Large y J1-Jumbo, respectivamente. También utilizamos un calentamiento lineal durante aproximadamente los primeros 375 millones de tokens y aumentamos gradualmente el tamaño del lote desde 32.000 tokens hasta su valor objetivo para los primeros miles de millones de tokens.	Las 4 tareas principales son: Tareas de clasificación de parejas de frases, tareas de clasificación de frases individuales, tareas de responder preguntas y tareas de etiquetado de frases.	80%	APIs propias de AI21 que realizan las tareas para las que ha sido entrenado el modelo	En la plataforma de AI21	Accesibles por AI21	Derechos de autor propios de AI21	Total	Ninguna
15	**GOPHER**	Google AI	2022	La arquitectura autorregresiva Transformer con dos modificaciones:	Utilizamos RMSNorm en lugar de LayerNorm, y utilizamos el esquema de codificación posicional relativa del Transformer XL en lugar de codificaciones posicionales absolutas. Las codificaciones relativas permiten evaluar en secuencias más largas de las que se entrenan, lo que mejora el modelado de artículos y libros. Tokenizan el texto utilizando SentencePiece con un vocabulario de 32.000 y utilizan un backoff a nivel de byte para apoyar el modelado de vocabulario abierto.	El modelo Gopher más grande tiene el siguiente tamaño: L=80, H = 128 y A= 128; 280B de parámetros.	Entrenamos la familia de modelos Gopher en MassiveText, una colección de grandes conjuntos de datos de texto en inglés de múltiples fuentes: páginas web, libros, artículos de noticias y código. En total, MassiveText contiene 2.350 millones de documentos, unos 10,5 TB de texto. Dado que entrenamos a Gopher con 300.000 millones de tokens (el 12,8% de los tokens del conjunto de datos), realizamos un submuestreo de MassiveText con proporciones de muestreo especificadas por subconjunto (libros, noticias, etc.). Ajustamos estas proporciones de muestreo para maximizar el rendimiento posterior.	Tareas como Traducción, responder preguntas o el avance de la seguridad y la justicia	75%	Modelo privado para la investigacion	Modelo privado para la investigacion	Modelo privado para la investigacion	Derechos de autor propios de google ai	Total	Ninguna
16	**Chinchilla**	Google AI	2022	Misma que Gopher	Es un modelo Gopher optimizado con un entrenamiento distinto	El tamaño óptimo del modelo para el presupuesto de cálculo de Gopher se sitúa entre 40.000 y 70.000 millones de parámetros. Para 70B: L = 80, H = 64 y A = 128	Chinchilla utiliza la misma arquitectura de modelo y configuración de entrenamiento que Gopher, con la excepción de las diferencias que se indican a continuación. - Entrenamos Chinchilla en MassiveText (el mismo conjunto de datos que Gopher) pero utilizamos una distribución de subconjuntos ligeramente diferente para tener en cuenta el mayor número de tokens de entrenamiento. - Utilizamos AdamW para Chinchilla en lugar de Adam, ya que esto mejora la pérdida de modelado del lenguaje y el rendimiento de la tarea descendente después del ajuste fino. - Entrenamos a Chinchilla con SentencePiece tokenizer que no aplica la normalización NFKC. El vocabulario es muy similar: el 94,15 % de los tokens son los mismos que los utilizados para entrenar a Gopher.	Tareas como Traducción, responder preguntas o el avance de la seguridad y la justicia	77%	Modelo privado para la investigacion	Modelo privado para la investigacion	Modelo privado para la investigacion	Derechos de autor propios de google ai	Total	Ninguna

5. ANÁLISIS DE TAREAS DE BACK OFFICE

Como ya hemos mencionado anteriormente, se pretende analizar la capacidad de los modelos anteriormente mencionados para automatizar tareas rutinarias. Para poder tener una visión de como de eficiente serian dichos procesos automatizados vamos a primero establecer una metodología y un criterio para determinar como de efectivas serias esas automatizaciones a la automatización robótica de procesos (RPA).

5.1 ADECUACIÓN DE LAS TAREAS PARA RPA

Para evaluar la eficacia de cualquier tarea a RPA, nos vamos a centrar en dos aspectos. El primero de ellos se centrará en valorar si la tarea es rutinaria o no rutinaria y el segundo de ellos si requiere el uso de funciones manuales o capacidades cognitivas. Las tareas altamente cognitivas requieren pensamiento creativo, al igual que las tareas no rutinarias con pocos o ningún patrón que tienen una gran variabilidad. Este tipo de tareas no son adecuadas para la automatización. Una regla práctica para determinar si una tarea es apta para la automatización es determinar si se pueden escribir con precisión todos los pasos del proceso, teniendo en cuenta todos los posibles acontecimientos y resultados a lo largo del camino. Aunque los avances en Inteligencia Artificial han permitido automatizar algunas tareas no rutinarias, el principio general que utilizaremos para determinar la eficacia seguirá siendo el mismo. Resumimos estos dos aspectos en la siguiente gráfica:

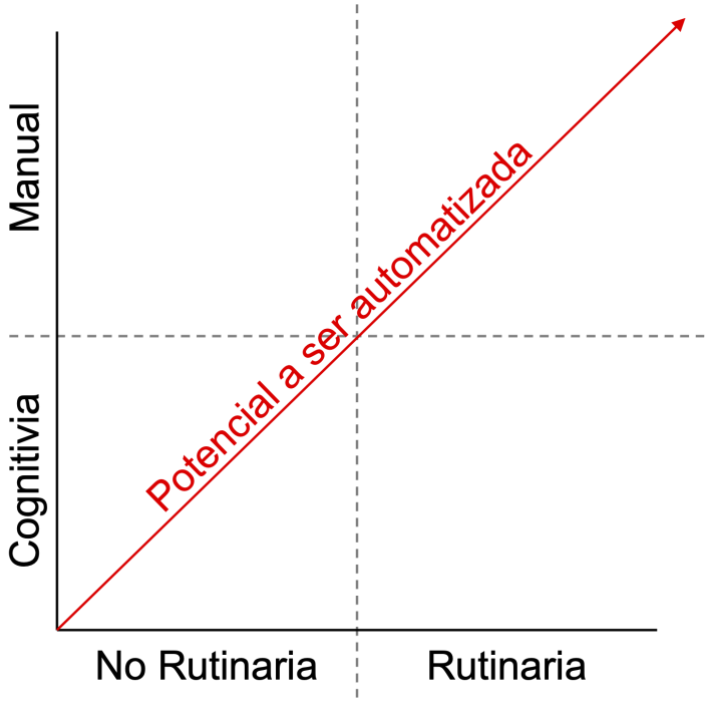

Figura 3. Guía para la automatización [50]

Para determinar si una tarea es adecuada para RPA, hay que tener en cuenta más factores. Además de la necesidad de que una tarea sea manual y rutinaria, una empresa dispuesta a utilizar la RPA debe considerar si es viable reemplazar a los humanos por robots para determinadas tareas y cuáles serían las implicaciones a largo plazo de tal decisión. Estos criterios también influyen en las decisiones estratégicas de los proveedores de RPA en relación con la comercialización y el marketing de la tecnología. La Tabla 1 presenta los criterios genéricos para decidir si una tarea es adecuada para RPA, utilizaremos estos criterios para asignar una puntación a las tareas que hemos seleccionado para determinar así su eficacia a la automatización. [50]

Tabla 3. Criterios para RPA [50]

Criterio	Descripción
Elevado número de transacciones	Tareas consideradas para RPA son tareas que se realizan de manera frecuente o incluyen un gran volumen de sub-tareas.
Necesidad de acceso a múltiples sistemas	El que haya que acceder a múltiples sistemas es un factor relevante. Por ejemplo, copiar datos de un Excel a una base de datos.
Entorno estable	Tareas que se ejecutan con un sistema predeterminado que no cambia durante la ejecución.
Bajos requisitos cognitivos	La tarea no requiere creatividad, juicios subjetivos o complejas interpretaciones.
Fácil de descomponer en pasos	La tarea es fácil de descomponer en sencillos pasos entre los cuales no hay espacio a las ambigüedades o malinterpretaciones.
Propensión al error humano	La tarea es propensa a que se cometan errores específicos a los humanos y no a los ordenadores. Por ejemplo, comparar valores de distintas columnas.
Necesidad limitada de tener que lidiar con excepciones	La tarea este estandarizada y por lo tanto no haya cavidad a excepciones mientras se completa la tarea.

Comprensión clara de los costes de la tarea	El comprender el desglose de costes permite estimar las diferencias al automatizar la tarea y determinar si se produce un retorno a la inversión.

Tras establecer unos criterios y unas reglas analizaremos las tareas y les daremos una puntuación del 1-5 en dos escalas: La primera siendo 1 - Tarea altamente Cognitiva y 5 - Altamente Manual. Y la segunda escala siendo 1 - Nada rutinaria y 5 – Altamente Rutinaria. Con estas dos escalas podremos tener una visión y comparas la eficacia a la automatización de las tareas seleccionadas para el análisis.

5.2 TAREAS DE BACK OFFICE

Como ya hemos mencionado anteriormente vamos a analizar las tareas que se han considerado relevantes en la industria para ser automatizadas. Son las siguientes:

1. Tareas de análisis de textos.
2. Extracción de datos, análisis de datos no estructurados y su organización.
3. Clasificación automática de documentos.
4. Generación automática de informes y resúmenes.
5. Traducción.
6. Generación automática de sugerencias de respuestas de mensajes y correos electrónicos.
7. Asistentes virtuales
8. Controles y verificaciones de *compliance*.
9. Sistemas de análisis y prevención de riesgos

Tareas de análisis de textos

El análisis de textos es un tipo de técnica que identifica los contenidos clave de grandes cantidades de documentos y los transforma en información procesable. Esta tarea de automatización extrae patrones e información de grandes fuentes de texto no estructurado y los transforma en datos estructurados.

Por ejemplo, esta tecnología permite identificar conceptos (como nombres de empresas, direcciones, fechas, etc.) en texto sin formato, o resumir información clave contenida en documentos extensos. También puede permitir identificar información incoherente en documentos distintos (por ejemplo, entre el folleto y otros documentos.

Este tipo de tarea requiere de pensamiento cognitivos ya que requiere de la interpretación de los textos para poder extraer la información, pero por el contrario es una tarea rutinaria ya que requiere constantemente de la búsqueda de información en textos de manera frecuente ya que en el día a día es bastante común la necesidad de buscar información en páginas web o documentos. Por lo tanto, puntuamos dicha tarea como: 2 – Cognitiva y 4 – Rutinaria.

Extracción de datos, análisis de datos no estructurados y su organización

Similar a la tarea anterior, pero centrándonos en el análisis de datos, tablas y bases de datos más que en texto. Es uno de los objetivos más perseguidos por la comunidad de expertos en gestión de datos. El desarrollo de sistemas automatizados que puedan procesar documentos semiestructurados y generar tablas que puedan consultarse sin esfuerzo humano ni personalización específica. Dada la enorme variedad de documentos que se podrían intentar estructurar, los sistemas más avanzados hacen suposiciones para simplificar y se basan en aprender información específica del sector para facilitar las tareas.

Los sistemas diseñados para abordar este problema deben equilibrar un triple balance entre coste (los lagos de datos pueden contener millones de documentos), calidad (las tablas de salida deben ser capaces de soportar con precisión las consultas de un analista) y generalidad (los diferentes lagos de datos tienen diferentes tipos de documentos y estructuras).

Este tipo de tareas son a la vez muy manuales y de realización rutinaria. Para un ser humano requiere mucho tiempo el analizar y estructurar grandes cantidades de datos. Un modelo de lenguaje puede realizar estas tareas rápidamente y si los resultados no son los esperados se puede fácilmente modificar una serie de parámetros partiendo del primer análisis realizado por el modelo. Punteamos esta tarea como 4 - Manual y 5 - Muy rutinaria.

Clasificación automática de documentos

A diferencia de las tareas anteriores no pedimos que se analicen el contenido de los documentos ni que se extraigan resultados de ellos. El objetivo principal es interpretar su contenido para determinar a donde pertenecen para poder tener unas bases de documentos estructuradas.

Los LLM aprovechan sus amplios conocimientos y capacidades de procesamiento del lenguaje para analizar datos no estructurados y extraer información significativa. Al entrenarse en amplios conjuntos de datos, los LLM pueden aprender patrones, semántica y claves contextuales dentro de los documentos, lo que les permite clasificar con precisión diversos tipos de contenido. Esta automatización elimina la necesidad de clasificación manual, ahorrando tiempo y recursos valiosos en diversos sectores.

Se pueden utilizar principalmente para estructurar documentos dentro de una empresa o donde más rendimiento se le puede dar a un clasificador de este tipo es para clasificar artículos, documentos, literatura facilitando la búsqueda de información a la hora de investigar ya que tendremos guardado en un mismo sitio toda la información relativa a el tema. La versatilidad en la clasificación automática de documentos los convierte en una poderosa herramienta para la automatización de back-office, mejorando la productividad y permitiendo a las empresas gestionar de forma eficiente sus procesos intensivos en documentos.

Dado que esta tarea sí que requiere una alta capacidad cognitiva para interpretar los textos la consideramos como un 1- Altamente cognitiva. Por otro lado, es una tarea que se realiza de manera muy constante por lo que la consideramos un 4 - Rutinaria.

Generación automática de informes y resúmenes

La tarea de crear informes a partir de datos de entrada es una importante tarea. Aprovechando las tecnologías avanzadas y las capacidades de procesamiento del lenguaje natural, las organizaciones pueden agilizar el proceso de generación de informes. Los datos de entrada, que pueden incluir resultados de investigaciones, análisis u otra información relevante, se analizan y transforman en informes completos mediante sistemas automatizados.

La generación automatizada de informes elimina la necesidad de compilación manual y reduce el riesgo de errores o incoherencias. Los informes generados mediante este proceso automatizado no sólo son eficientes, sino también precisos y coherentes en diferentes tareas y proyectos.

Además, la automatización del proceso de creación de informes permite a las organizaciones ahorrar tiempo y recursos, al tiempo que garantiza un plazo de entrega más rápido. Esta tarea es de gran utilizad y requiere una capacidad cognitiva elevada, pero a su vez se realiza de manera continua. Por lo tanto, le damos unas puntaciones de: 1- Altamente Cognitiva y 5 – Altamente Rutinaria.

Traducción

La traducción es una tarea administrativa que puede automatizarse considerablemente con la ayuda de la inteligencia artificial. Los sistemas de traducción automática han revolucionado el proceso de traducción al ofrecer soluciones de traducción rápidas y eficaces.

La automatización de la traducción no sólo acelera el proceso, sino que también mejora la coherencia y la precisión. A diferencia de la traducción manual tradicional, que puede llevar mucho tiempo y ser propensa a errores humanos. Además, los sistemas de traducción automática pueden personalizarse y ajustarse a ámbitos o sectores específicos, lo que mejora aún más la precisión y pertinencia de las traducciones.

Aunque la traducción automática es una herramienta valiosa, es importante señalar que la revisión humana sigue siendo crucial para garantizar el máximo nivel de precisión lingüística. Es por ella que es una tarea que requiere mucha capacidad cognitiva, 1 - Altamente Cognitiva. Por otro lado, es una tarea que se realiza a menudo, pero no tanto como otras de las que tenemos en la lista, es por ello por lo que se le da una. Puntuación de 3 – Medianamente Rutinaria.

Generación automática de sugerencias de respuestas de mensajes y correos electrónicos

Las organizaciones y particulares pueden agilizar sus comunicaciones y ahorrar tiempo garantizando una respuesta coherente a los mensajes que se reciben que se asimile al estilo de escritura de cada uno. Cabe destacar que una comunicación personal, actualmente, es algo que requiere de verificación del dueño del contenido. Pero es un sistema que es de gran utilidad para agilizar las respuestas y la jornada laboral.

Cuando se recibe un correo electrónico, los LLM pueden analizar el contenido, extraer información clave y generar sugerencias de respuesta adaptadas al contexto específico. Se pueden comprender los matices del contenido del correo electrónico, incluidos el tono, el sentimiento y la intención, lo que permite generar sugerencias apropiadas y contextualmente relevantes.

Además, se pueden utilizar datos históricos de correos electrónicos anteriores para ofrecer sugerencias de respuesta más personalizadas.

No solo se ahorra tiempo, sino que también se reduce la carga cognitiva de los empleados. En lugar de elaborar manualmente las respuestas desde cero, los empleados pueden revisar y modificar las respuestas sugeridas, lo que hace que el proceso sea más eficiente y sin errores. También el modelo puede aprender de los comentarios y ajustes realizados por los usuarios, mejorando continuamente sus capacidades de generación de respuestas con el tiempo.

Los emails y cualquier tipo de comunicación entre personas es una tarea de gran importancia que nunca se va a dejar de realizar de manera diaria. Si que aun así es una tarea que requiere de gran personalización y depende mucho del mensaje de entrada por ello no es una tarea tan monótona. Por ello puntuamos esta tarea con un 3 - Medianamente cognitiva y 4 - Rutinaria.

Asistentes virtuales y *Chatbots*

Los asistentes virtuales o *chatbots* han revolucionado las industrias automatizando una amplia gama de tareas de consulta de información.

Al aprovechar la potencia de los LLM, los asistentes virtuales ofrecen a las organizaciones soluciones escalables y eficientes para las interacciones con los clientes. Pueden proporcionar respuestas instantáneas a preguntas frecuentes, resolver problemas comunes e incluso simular conversaciones similares a las humanas. Los asistentes virtuales no sólo ahorran tiempo a los clientes, sino que también mejoran su satisfacción ofreciéndoles una asistencia precisa y personalizada. Estos asistentes pueden acceder y recuperar información de bases de datos, realizar búsquedas complejas y presentar resultados relevantes. Al automatizar estas tareas, los asistentes virtuales agilizan las operaciones administrativas, reducen el esfuerzo manual y permiten a las organizaciones ofrecer asistencia las 24 horas del día.

Además, los asistentes virtuales basados en modelos LLM aprenden continuamente y mejoran su rendimiento con el tiempo. A medida que interactúan con los usuarios, recopilan datos y conocimientos que pueden utilizarse para mejorar su comprensión del lenguaje y sus capacidades de generación de respuestas. Este proceso de aprendizaje iterativo permite a los asistentes virtuales ser más eficientes, precisos y capaces de gestionar tareas complejas. Las organizaciones pueden aprovechar este aprendizaje continuo para formar a los asistentes virtuales en dominios específicos, asegurándose de que poseen conocimientos especializados y pueden responder a consultas específicas del sector.

Cabe destacar que también es importante que el ámbito de aplicación del chatbot siga siendo limitado y que los humanos tomen el relevo cuando se discutan temas más críticos. La responsabilidad por la información proporcionada y las acciones realizadas por los *chatbot*s, incluso cuando son de naturaleza puramente informativa, siguen siendo de la institución.

Dado que nunca van a sustituir las consultas con personas sí que pueden facilitar tareas más sencillas de búsqueda de información y procedimientos sencillos para los usuarios. Es por ello por lo que no son de uso bastante rutinario y no requieren de gran capacidad cognitiva. 3 – Medianamente Manuales y 3 – Rutinario.

Controles y verificaciones de *compliance*

Los controles de conformidad son fundamentales para garantizar el cumplimiento de los requisitos normativos y las políticas internas. Actualmente se realizan este tipo de verificaciones a través de auditorías, tanto internas como externas que pueden llegar a tener repercusiones serias si no se están cumpliendo determinadas normas.

Las organizaciones pueden automatizar la verificación y validación de documentos, bases de datos e información con respecto a diversas normativas. Se pueden automatizar tareas de identificación de los principales requisitos a cumplir por las empresas, extraer la información esencial de textos jurídicos complejos y estudio de las políticas internas. Esta automatización reduce significativamente la necesidad de revisión manual, ahorra tiempo y mejora la eficacia general del cumplimiento, asegurando la detección en caso de no cumplimiento para poder remediar la situación.

Un enfoque proactivo ayuda a las organizaciones a identificar posibles problemas de cumplimiento en fases temprana o de aparición de nuevas normativas, lo que permite tomar medidas correctivas a tiempo. Esta automatización garantiza la coherencia, precisión y puntualidad de los informes de cumplimiento, reduciendo la carga de trabajo de los equipos de cumplimiento.

Este tipo de tareas no se realizan de carácter rutinario, pero con un sistema automático de detección una empresa se podría asegurar el correcto cumplimiento en todo momento. A su vez no es una tarea manual ya que requiere mucha experiencia sobre las legislaciones vigentes. Por lo tanto, se considera que, aunque es una tarea que tendría un gran impacto su automatización no es de las más sencillas y eficaces de automatizar. 2- Coherente y 1 – No rutinaria.

Sistemas de análisis y prevención de riesgos

Los sistemas de análisis y prevención de riesgos son herramientas esenciales para las operaciones, ya que permiten a las organizaciones identificar y mitigar proactivamente los riesgos potenciales en diversos ámbitos. Los LLM, con su capacidad para analizar grandes cantidades de datos y detectar patrones, ayudan a las organizaciones a tomar decisiones informadas y proteger sus intereses.

Al entrenar ser capaces de entrenar a un modelo con datos históricos e incorporar información en tiempo real, los sistemas de análisis de riesgos pueden evaluar eficazmente los riesgos y proporcionar señales de alerta temprana. Los sistemas de mantenimiento predictivo se basan en la identificación de anomalías, valores atípicos y tendencias que pueden indicar riesgos o amenazas potenciales. Automatizar la supervisión y detección de riesgos proporcionan a las organizaciones una visión completa de su panorama de riesgos, lo que les permite tomar medidas y aplicar estrategias de mitigación de riesgos antes de que sucedan.

Al introducir continuamente nuevos datos en los modelos, las organizaciones pueden mantener actualizados sus sistemas de mantenimiento predictivo y asegurarse de que están equipados para hacer frente a los riesgos emergentes. Son tareas que sin ningún sistema ni modelo son complicadas de realizar, son manuales ya que se basan en el análisis de los mismos datos y estudiar las mismas tendencias, por ello son de gran utilizad su automatización, 4 - Medianamente Manual. Por otro lado, el verificar que los sistemas y maquinas funcionan de manera adecuada es una tarea que se realiza a menudo, pero no diariamente por lo tanto se puntúa también como un 3 - Medianamente Rutinaria.

6. COMPARATIVA Y ANÁLISIS

Tras analizar los distintos tipos de tareas podemos analizar estos resultados. Primero podemos ver a que cuadrante de nuestra grafica de la guía para la automatización pertenece cada tarea:

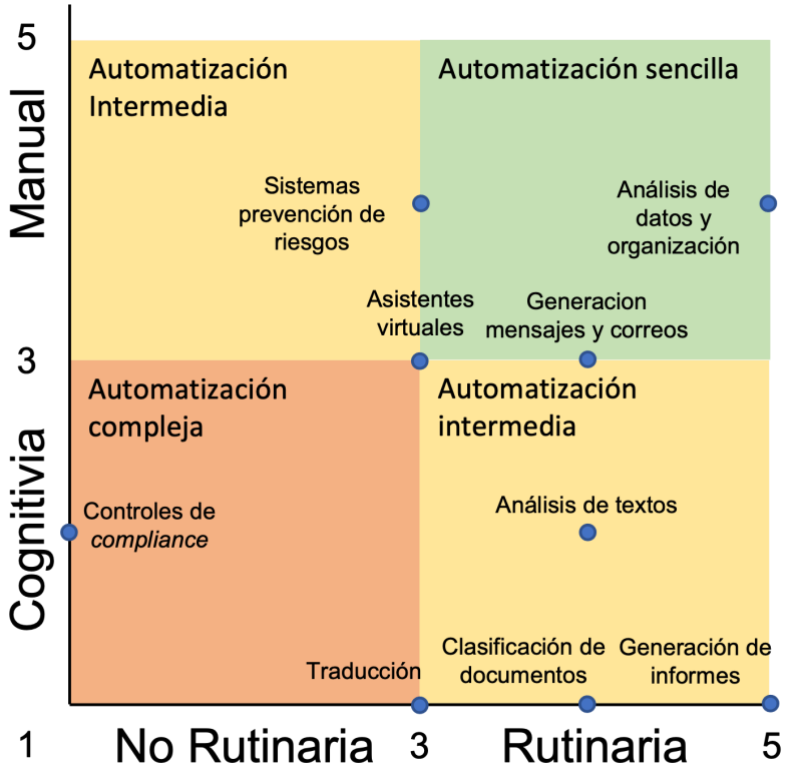

Figura 4. Gráfica de automatización con las tareas

En la tabla anterior podemos ver que se han delimitado 4 cuadrantes, en función de cómo de factible y eficaz se considera la automatización de ese proceso. Podemos ver la gran mayoría se encuentran actualmente en los cuadrantes que requerirían tiempo automatizar las tareas, lo que no implica que no sea posible. Esto demuestra que los avances de estas tecnologías nos llevan a que cada vez será más fácil este proceso.

6.1 COMPARATIVA MODELOS Y TAREAS

En el análisis realizado sobre los modelos seleccionados hemos podido ver que son diseñados con ciertas tareas en mente, por lo tanto, se van a comparar y analizar que modelos se podrían utilizar para automatizar cada tarea.

De los modelos que se han analizado, se pueden clasificar en las siguientes categorías:

Tabla 4. Tabla resumen tareas y modelos.

Tareas que pueden procesar	Nuestros Modelos	Nuestras Tareas
Clasificación de textos	BERT, ELECTRA, RoBERTa, LLAMA, ELMO, XLNET, DistilBERT y JURASSIC 1	Clasificación automática de documentos.
Responder a preguntas	BERT, T5, ELECTRA, GPT-4, RoBERTa, LLAMA, ELMO, XLNET, DistilBERT, JURASSIC 1, GOPHER y Chinchilla	
Resumen de documentos	T5, GPT- 4 y XLNET	Generación automática de informes y resúmenes.
Clasificar sentimientos	T5, GPT- 4, LLAMA y ELMO	
Generación de textos	Reformer, GPT- 4, UniLM y MT-NLG 530B	Generación automática de mensajes y correos

Generación música, imágenes y videos	Reformer	
Generación de código	LLAMA	
Razonamiento de sentido común	GPT- 4, LLAMA, ELMO y MT-NLG 530B	
Compresión lectora	UniLM y MT-NLG 530B	Tareas de análisis de textos.
Chatbot	GPT- 4 y LaMDA	Asistentes virtuales
Traducción	GOPHER y Chinchilla	Traducción
Razonamiento de Seguridad y Justicia	GOPHER y Chinchilla	Controles y verificaciones de compliance.
		Extracción de datos, análisis de datos no estructurados y su organización.
		Sistemas de análisis y prevención de riesgos

Como podemos comprobar al analizar la tabla anterior vemos que la gran mayoría de modelos pueden servir para realizar el mismo tipo de tareas. Modelos como LLAMA y GPT-4 son los más versátiles.

Por el contrario, solamente dos de las tareas que se han analizado no tienen ningún modelo específicamente diseñado para realizarlas. La extracción, análisis de datos no estructurados y su organización; y los sistemas de análisis y prevención de riesgos; se clasifican como unos tipos de tarea distintos a los anteriores. A diferencia de las demás, estas no se centran tanto en el lenguaje natural y más en los números y los datos. Es por ello por lo que estos modelos específicamente diseñados para trabajar con el lenguaje tienen más dificultades con la realización de este tipo de tareas.

6.1.1 SELECCIÓN TAREA Y MODELO PARA LA REALIZACIÓN DE UNA DEMO

Dado que la gran mayoría de los modelos analizados son *open source*, se podrían utilizar para realizar un caso práctico de uso. Valdría con descargárselo y generar una aplicación propia con el modelo.

Otra opción es conectándose directamente a la API de los modelos directamente en internet, de esta manera nos ahorraríamos el tener que descargar y tener el modelo en nuestro propio ordenador. Se escogió, por lo tanto, concertarse directamente a la API de *Openai* para acceder a sus modelos, para simplificar al máximo la demo y a su vez utilizar uno de los modelos más relevantes de la actualidad. Se utilizará por lo tanto *GPT 3.5 Turbo*.

En cuanto a la tarea se quiere automatizar, los modelos GPT se especializan en la generación de texto y respuestas a preguntas por lo tanto se seleccionará una tarea en estos ámbitos para la demo. A la hora de la selección se ha tenido en cuanta intentar automatizar una tarea que resulte útil y de realización diaria para todo el mundo. Es por ello por lo que se selecciona la generación automática de correos electrónicos, ya que en el día a día se reciben una gran cantidad de ellos y el poder partir de una posible respuesta facilitará la vida a quienes utilicen la aplicación.

7. SIMULACIÓN/DEMO

Como hemos podido comprobar en el apartado anterior la eficacia de los modelos analizados para poder automatizar las tareas presentadas se estima que es muy elevada debido a lo avanzadas que se encuentran. Para demostrar cómo se podría realizar dichas automatizaciones y su eficacia hemos decidido centrarnos en una tarea que actualmente nos ha parecido que sería de gran utilizad para un elevado porcentaje de la población: La Generación automática de sugerencias de respuestas a mensajes y correos electrónicos.

Se ha desarrollado una sencilla aplicación haciendo uso de *Python.* En esta pequeña demo nos centraremos en generar una sugerencia a un correo electrónico entrante. Los parámetros de entrada serán conversaciones de correo electrónico pasadas para que el modelo pueda analizar el estilo de escritura del usuario para utilizarlo en su sugerencia de respuesta.

Como hemos mencionado anteriormente, el modelo seleccionado para esta demo ha sido el modelo de *Open AI GPT 3.5 Turbo*, que, aunque no es uno de los modelos analizados es el modelo en el que se basaron para GPT 4, al cual todavía no se tiene acceso total para ser utilizado. Se ha elegido este modelo porque se ha visto que los modelos de Open AI son los más versátiles y los que mejor interaccionan con el usuario, en tareas de generación de texto y respuesta a preguntas.

7.1 PARÁMETROS DE ENTRADA

Dado que se pretendía solamente realizar una pequeña demo de cómo se podría automatizar este proceso y no realizar una aplicación, el código desarrollado no se ha conectado directamente a una aplicación de correo electrónico, como podría ser Outlook. Esta conexión se hubiera utilizado para extraer tanto los correos ya contestados para ser utilizados como repositorio para que el modelo aprenda, y también para saber a qué correo debe el modelo generar una respuesta.

Para simplificar dicha extracción de datos se introducen al modelo un archivo Excel. Se titula repositorio de mails y tiene la siguiente estructura en su única hoja:

Tabla 5. Estructura archivo Repositorio de mails

Emisor	Asunto	Contenido	Respuesta del usuario
Nombre Emisor 1	Asunto mail 1	Contenido mail1	Respuesta del usuario a dicho correo electrónico.
Emisor 2	Asunto 2	Contenido 2	Respuesta 2
…	…	…	…

El tamaño de este archivo dependerá de cuantos correos queremos que el modelo analice para después sugerir la respuesta.

Una vez introducido esto el modelo analiza el estilo de escritura del usuario para poder después sugerir una respuesta a un correo entrante. Dicho correo entrante se introducirá al modelo de forma manual directamente en la interfaz de usuario que se explica a continuación.

7.2 INTERFAZ USUARIO

Se ha desarrollado una sencilla interfaz usuario para que sea fácil de manejar el programa.

Nada más inicializar el código nos aparece la siguiente pantalla de inicio:

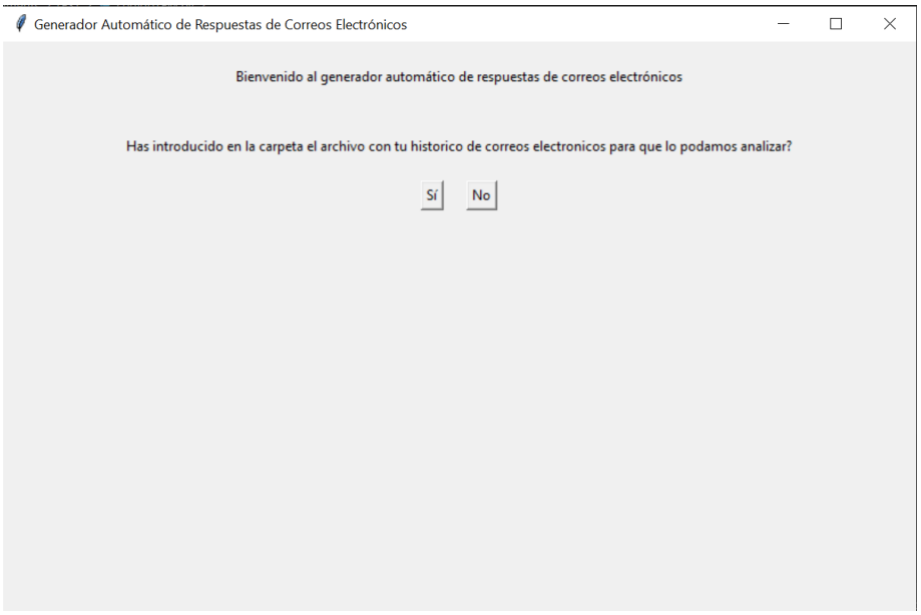

Figura 5. Ventana de Inicio

Una sencilla ventana que nos da la bienvenida y nos pregunta si hemos insertado el archivo Excel de correos electrónicos en la carpeta correspondiente.

Si se pulsa que no, salta un aviso indicando que es necesario tener el Excel.

Figura 6. Ventana emergente con aviso

Tras confirmar la presencia del archivo, la aplicación pide al usuario que introduzca el correo al que se desea contestar:

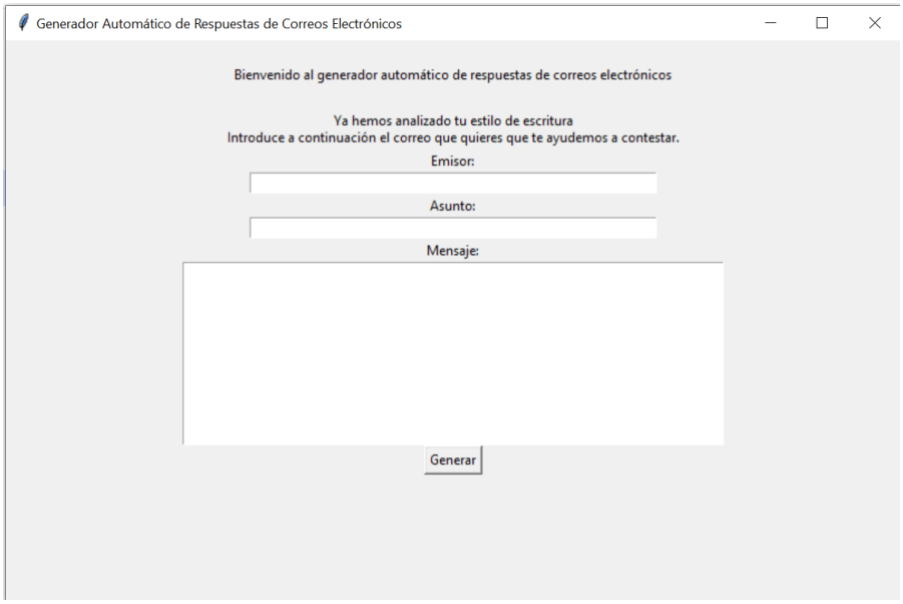

Figura 7. Venta de introducción de correo entrante

Una vez introducido se pulsará generar y el programa automáticamente genera una posible respuesta a dicho correo. La cual, puede ser editada por el usuario antes de enviarla.

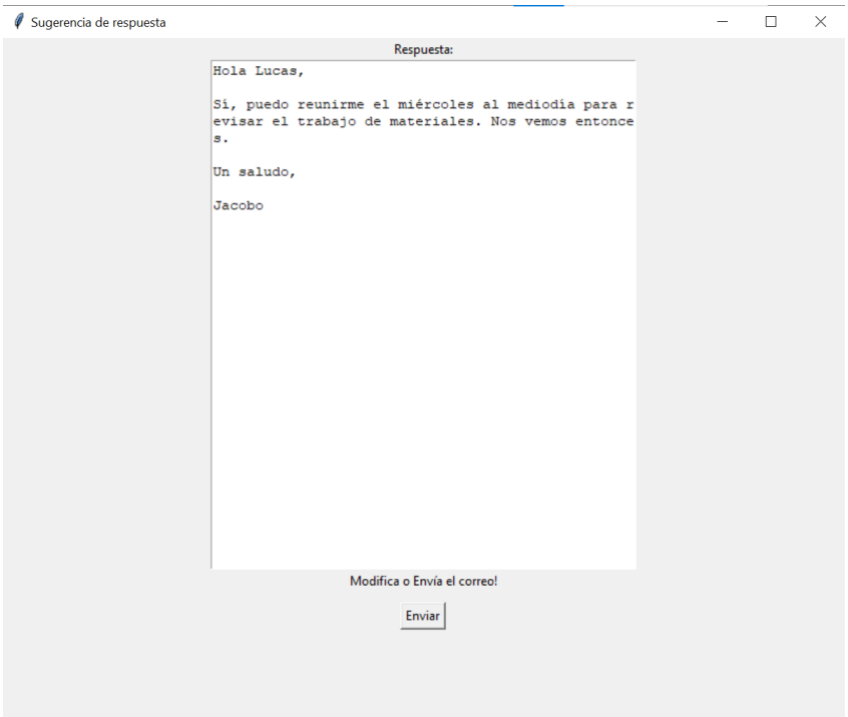

Figura 8. Correo Automáticamente Generado

Dicho correo una vez se pulse enviar, será guardado dentro del archivo de correos históricos para que la próxima vez que se vuelva a utilizar el programa lo tenga en cuenta para analizar el estilo de escritura del usuario.

7.3 ANÁLISIS DE LOS ASPECTOS MÁS RELEVANTES DEL CÓDIGO

Vamos a continuación a analizar algunas partes del código para entender cuál ha sido el razonamiento utilizado para esta automatización.

La parte más importante del código es la función que nos permite, una vez accedido con la API *Key* a *Openai*, mandar preguntas al modelo para que nos las conteste. Estos bloques de texto, *prompts*, contienen la información relevante para que el modelo nos dé una respuesta o si se lo pedimos nos genere un texto directamente.

```
def get_completion(prompt, model="gpt-3.5-turbo"):
    messages = [{"role": "user", "content": prompt}]
    response = openai.ChatCompletion.create(
        model=model,
        messages=messages,
        temperature=0,
    )
    return response.choices[0].message["content"]
```

Figura 9. Función con la conexión a través de la API a Openai

Es una función donde introducimos el *promt* y nos genera directamente la respuesta del modelo. El único otro parámetro de entrada que se puede variar es la temperatura. La temperatura es un oarametro de los LLMs que controla la aleatoriedad en la generación de texto, una temperatura alta produce resultados más creativos y una temperatura baja genera respuestas más deterministas y coherentes.

La segunda función de gran relevancia de nuestro programa es la cuenta los tokens de las frases o promts. Los tokens son palabras o partes más pequeñas de las palabras que el modelo utiliza para entender y procesar textos. Se define por lo tanto un proceso para realizar esta división en tokens. Este proceso se llama tokenización.
La división en tokens es importante para representar y procesar el texto de manera adecuada

en modelos de lenguaje. También mejora la eficiencia, el manejo de palabras desconocidas y la comprensión del lenguaje.

La división en tokens al usar modelos de lenguaje tiene muchas razones importantes, y para nosotros, tres de ellas destacan. La primera razón crucial es evitar problemas de memoria y ajustarse a los límites del modelo, ya que estos tienen una capacidad limitada para procesar tokens en una secuencia. Además, contar los tokens nos permite optimizar la eficiencia computacional, asignando recursos adecuados y estimando el rendimiento necesario para procesar la entrada de manera eficaz. Por otra parte, el costo en sistemas de pago por token es relevante, ya que algunas API de modelos de lenguaje, como la que se está utilizando en este caso, cobran por el número de tokens utilizados. Contar los tokens ayuda a estimar los costos y ajustar el uso para mantenerse dentro de los límites presupuestarios.

```python
def num_tokens_from_string(string: str, encoding_name: str) -> int:
    """Returns the number of tokens in a text string."""
    encoding = tiktoken.get_encoding(encoding_name)
    encoded_string = encoding.encode(string)
    num_tokens = len(encoding.encode(string))
    print("Encoded string:", encoded_string)
    return num_tokens
```

Figura 10. Función contadora de tokens

Es por ello por lo que nos queremos ceñir a 3000 tokens de entrada cada vez que llamemos al modelo. Por ello antes de realizar esta llamada se contarán los tokens del *promt*. La metodología seguida es la siguiente:

Primero se intentan meter todos los correos como información historia dentro del *promt* para poder tener la mayor precisión posible. Si hay más de 3000 tokens analizamos uno a uno el estilo empleado en cada uno de los correos y vamos guardando en una cadena de caracteres, el análisis realizado por el modelo. De esta manera reducimos el número de tokens de entrada de cada *promt*, pero se mantiene como informacion relevante el estilo de escritura del usuario para que el modelo se intente asemejar a él lo más posible.

El siguiente trozo de código es el utilizado para guardar el nuevo correo al que se le ha generado una respuesta dentro del archivo histórico de correos de tal manera que la siguiente vez que se vuelva a utilizar el programa se tiene toda la información disponible.

```
# Define los nuevos valores
nuevo_emisor = emisor
nuevo_asunto = asunto
nuevo_contenido = message
nueva_respuesta = respuesta_final

# Agrega los nuevos valores en la última fila del DataFrame
df.loc[ultima_fila] = [nuevo_emisor, nuevo_asunto, nuevo_contenido, nueva_respuesta]

# Guarda el DataFrame actualizado en el archivo de Excel
df.to_excel('C:\\Users\\jacobozf\\Documents\\TEST\\repositorio_de_mails.xlsx', sheet_name='Sheet1',
index=False)
```

Figura 11. Código que guarda el nuevo correo

Como se puede ver se guardan los nuevos valores en la siguiente fila disponible del archivo.

8. CONCLUSIONES

Este trabajo se ha enfocado en llevar a cabo un exhaustivo análisis y comparativa de los modelos de lenguaje más importantes, con el fin de identificar sus características y capacidades para abordar diversas tareas de back office. Este proceso permitió evaluar la eficacia y versatilidad de los LLMs en el ámbito de la automatización inteligente de procesos.

Durante el desarrollo de este estudio, se destacó la importancia de contar con LLMs eficientes y precisos para el procesamiento del lenguaje natural. La diversidad de modelos disponibles, cada uno con sus propias características y tamaños, ofreció un panorama amplio para la selección de aquellos más adecuados para tareas específicas de back office.

La segunda fase del trabajo consistió en el análisis de posibles tareas de automatización en el ámbito del back office. Se exploraron diferentes áreas, como el análisis de textos, la extracción y organización de datos no estructurados, la clasificación automática de documentos, la generación de informes y resúmenes, la traducción, entre otras. Cada tarea se evaluó cuidadosamente en términos de su idoneidad para la automatización.

La posterior comparativa detallada entre los modelos de lenguaje y las tareas seleccionadas reveló importantes perspectivas sobre cómo los LLMs pueden desempeñar un papel fundamental en la automatización de tareas de back office, mejorando la eficiencia y reduciendo la carga de trabajo manual.

Es importante destacar que el desarrollo y avance de los Modelos de Lenguaje de Aprendizaje Profundo es un campo en constante evolución. A medida que se avance en la investigación y la mejora de estos modelos, se espera un crecimiento continuo en sus capacidades y aplicaciones. La combinación de una selección adecuada de modelos con tareas optimizadas para la automatización inteligente promete un futuro prometedor en el ámbito de la transformación digital y la eficiencia empresarial.

Con una simple demo final se ha conseguido demostrar la facilidad con la que se pueden automatizar determinadas tareas. Escogiendo los modelos y las aplicaciones adecuadas rápidamente se pueden empezar a automatizar tareas de oficina de realización diaria.

Este trabajo, por lo tanto, ha sentado las bases para futuras investigaciones y aplicaciones en la utilización de LLMs en el ámbito del back office, resaltando el potencial de estas tecnologías para revolucionar y mejorar el mundo laboral a través de la automatización inteligente y la optimización de procesos.

9. BIBLIOGRAFÍA

[1] S. R. Joseph, H. Hlomani, K. Letsholo, F. Kaniwa, and K. Sedimo, "Natural Language Processing: A Review," *International Journal of Research in Engineering and Applied Sciences*, vol. 6, no. 3, 2016.

[2] M. Rahman Minar and J. Naher, "Recent Advances in Deep Learning: An Overview," 2018.

[3] J. Yang, H. Jin, and X. Han, "Harnessing the Power of LLMs in Practice: A Survey on ChatGPT and Beyond; Harnessing the Power of LLMs in Practice: A Survey on ChatGPT and Beyond," 2023.

[4] E. Alpaydin, *Introduction to Machine Learning*, Fourth Edition. MIT Press, 2020.

[5] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. 2016.

[6] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language Models are Unsupervised Multitask Learners".

[7] D. Jurafsky and J. H. Martin, "Speech and Language Processing An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition Third Edition draft Summary of Contents," 2023.

[8] R. Syed *et al.*, "Robotic Process Automation: Contemporary themes and challenges," *Comput Ind*, vol. 115, p. 103162, Feb. 2020.

[9] "UiPath Inc. (n.d.). Navigate RPA Journey & Steps - Automation Process | UIPATH. https://www.uipath.com/rpa/journey".

[10] Commission de Surveillance du Secteur Financier, "OPPORTUNITIES, RISKS AND RECOMMENDATIONS FOR THE FINANCIAL SECTOR," 2018.

[11] C. Benedikt Frey *et al.*, "THE FUTURE OF EMPLOYMENT: HOW SUSCEPTIBLE ARE JOBS TO COMPUTERISATION?," 2013.

[12] A. Vaswani *et al.*, "Attention Is All You Need".

[13] J. Devlin, M.-W. Chang, K. Lee, K. T. Google, and A. I. Language, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding".

[14] A. R. Openai, K. N. Openai, T. S. Openai, and I. S. Openai, "Improving Language Understanding by Generative Pre-Training".

[15] T. B. Brown *et al.*, "Language Models are Few-Shot Learners," 2020.

[16] C. Raffel *et al.*, "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer," *Journal of Machine Learning Research*, vol. 21, pp. 1–67, 2020, Accessed: May 31, 2023.

[17] S. Smith *et al.*, "Using DeepSpeed and Megatron to Train Megatron-Turing NLG 530B, A Large-Scale Generative Language Model," 2022.

[18] M. E. Peters, M. Neumann, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," 2018.

[19] K. Aitken, "Understanding How Encoder-Decoder Architectures Attend," 2021.

[20] J. Lin, X. Mao, Y. Chen, L. Xu, Y. He, and H. Xue, "D 2 ETR: Decoder-Only DETR with Computationally Efficient Cross-Scale Attention," 2022.

[21] Y. Liu *et al.*, "RoBERTa: A Robustly Optimized BERT Pretraining Approach," 2019, Accessed: Jul. 02, 2023.

[22] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter," 2020.

[23] J. Lee *et al.*, "Data and text mining BioBERT: a pre-trained biomedical language representation model for biomedical text mining," 2019.

[24] G. Lample and A. Conneau, "Cross-lingual Language Model Pretraining," 2019, Accessed: Aug. 01, 2023.

[25] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V. Le, "XLNet: Generalized Autoregressive Pretraining for Language Understanding," *Adv Neural Inf Process Syst*, vol. 32, Jun. 2019.

[26] Z. Lan *et al.*, "ALBERT: A LITE BERT FOR SELF-SUPERVISED LEARNING OF LANGUAGE REPRESENTATIONS," 2020.

[27] K. Clark, M.-T. Luong, G. Brain, Q. V Le Google Brain, and C. D. Manning, "ELECTRA: PRE-TRAINING TEXT ENCODERS AS DISCRIMINATORS RATHER THAN GENERATORS," 2020.

[28] A. Zeng *et al.*, "GLM-130B: AN OPEN BILINGUAL PRE-TRAINED MODEL," 2022, Accessed: Aug. 01, 2023.

[29] Z. Chi *et al.*, "XLM-E: Cross-lingual Language Model Pre-training via ELECTRA," 2022, Accessed: Aug. 01, 2023.

[30] S. Soltan *et al.*, "AlexaTM 20B: Few-Shot Learning Using a Large-Scale Multilingual Seq2seq Model," 2022, Accessed: Aug. 01, 2023.

[31] S. Zhang *et al.*, "OPT: Open Pre-trained Transformer Language Models," 2022, Accessed: Aug. 01, 2023.

[32] A. Chowdhery *et al.*, "PaLM: Scaling Language Modeling with Pathways," 2022.

[33] B. Workshop *et al.*, "BLOOM: A 176B-Parameter Open-Access Multilingual Language Model Major Contributors Prompt Engineering Architecture and Objective Engineering Evaluation and Interpretability Broader Impacts," 2023.

[34] N. Du *et al.*, "GLaM: Efficient Scaling of Language Models with Mixture-of-Experts," 2022.

[35] J. W. Rae *et al.*, "Scaling Language Models: Methods, Analysis & Insights from Training Gopher," 2022.

[36] J. Hoffmann *et al.*, "Training Compute-Optimal Large Language Models," 2022.

[37] R. Thoppilan *et al.*, "LaMDA: Language Models for Dialog Applications," 2022.

[38] H. Touvron *et al.*, "LLaMA: Open and Efficient Foundation Language Models," 2023, Accessed: Aug. 01, 2023.

[39] OpenAI, "GPT-4 Technical Report," 2023.

[40] S. Wu *et al.*, "BloombergGPT: A Large Language Model for Finance," 2023.

[41] A. Karpathy, J. Johnson, and L. Fei-Fei, "VISUALIZING AND UNDERSTANDING RECURRENT NETWORKS," 2015.

[42] A. Karpathy, "The Unreasonable Effectiveness of Recurrent Neural Networks," 2015.

[43] R. Jozefowicz, O. Vinyals, M. Schuster, N. Shazeer, Y. Wu, and G. Brain, "Exploring the Limits of Language Modeling," 2016.

[44] "EU AI Act: first regulation on artificial intelligence | News | European Parliament." https://www.europarl.europa.eu/news/en/headlines/society/20230601STO93804/eu-ai-act-first-regulation-on-artificial-intelligence

[45] N. Kitaev, Ł. Kaiser, A. Levskaya, and G. Research, "REFORMER: THE EFFICIENT TRANSFORMER," 2020.

[46] H. Touvron *et al.*, "LLaMA: Open and Efficient Foundation Language Models," 2023, Accessed: Jul. 02, 2023.

[47] L. Dong *et al.*, "Unified Language Model Pre-training for Natural Language Understanding and Generation," 2019, Accessed: Jul. 02, 2023.

[48] O. Lieber, O. Sharir, B. Lenz, and Y. Shoham, "JURASSIC-1: TECHNICAL DETAILS AND EVALUATION".

[49] J. Hoffmann *et al.*, "Training Compute-Optimal Large Language Models," 2023.

[50] A. Asatiani and E. Penttinen, "Turning robotic process automation into commercial success - Case OpusCapita," *Journal of Information Technology Teaching Cases*, vol. 6, no. 2, pp. 67–74, Nov. 2016.

ANEXO I. CÓDIGO PROGRAMA GENERADOR

AUTOMÁTICO DE CORREOS ELECTRÓNICOS

```
# Set-up
# Importaciones + conexión con API de OpenAI + funciones auxiliares
import tiktoken
import tkinter as tk
from tkinter import filedialog, messagebox
import pandas as pd
import numpy as np
import openai
import os
import json
import win32com.client as win32
from IPython.display import display, Markdown, Latex, HTML, JSON


from dotenv import load_dotenv, find_dotenv
_ = load_dotenv(find_dotenv()) # read local .env file
api_key_test = os.environ['API_KEY']
openai.api_key = api_key_test
#print(api_key_test)
#print(openai.api_key)

def get_completion(prompt, model="gpt-3.5-turbo"):
    messages = [{"role": "user", "content": prompt}]
    response = openai.ChatCompletion.create(
        model=model,
        messages=messages,
        temperature=0,
    )
```

```
    return response.choices[0].message["content"]


encoding = tiktoken.get_encoding("cl100k_base")

def num_tokens_from_string(string: str, encoding_name: str) -> int:
    """Returns the number of tokens in a text string."""
    encoding = tiktoken.get_encoding(encoding_name)
    encoded_string = encoding.encode(string)
    num_tokens = len(encoding.encode(string))
    #print("Encoded string:", encoded_string)
    return num_tokens


contador_tokens = 0

promt_inicial = f"""
Soy un estudiante de segundo de master en ingenieria industrial e industria conectada.\
Las actividades que realizo a diario son estudiar, ir a clase y realizar trabajos en grupo.

A continuación yo te voy a dar una lista de ejemplos de emails recibidos y respuestas.\
Tu tarea es analizar el estilo con el que respondo para luego generar sugerencias de respuesta de forma automática.\
Que tu respuesta sea solo la respuesta al email.\
Lista de emails:
"""
type(promt_inicial)

contador_tokens += num_tokens_from_string(promt_inicial, "cl100k_base")
#print(contador_tokens)


# Crear una matriz vacía para almacenar los datos de los correos
matriz_emails = []
memoria_promts = ""
memoria_promts += promt_inicial
```

```
# Lee el archivo de Excel
# con doble barra el camino
df = pd.read_excel('C:\\Users\\jacobozf\\Documents\\TEST\\repositorio_de_mails.xlsx', sheet_name='Sheet1')

# Almacena la información en una matriz
matriz_emails = df.values

ultima_fila = df.shape[0]

# Imprimir los datos de los correos almacenados
for i, datos in enumerate(matriz_emails):
    #print(f"Correo {i + 1}:")
    #print("Emisor:", datos[0])
    #print("Asunto:", datos[1])
    #print("Cuerpo:", datos[2])
    #print("Respuesta:", datos[3])
    #print("--------------------")

    memoria_promts += (f"Correo {i + 1}:")
    memoria_promts += (f"Emisor: {datos[0]}")
    memoria_promts += (f"Asunto: {datos[1]}")
    memoria_promts += (f"Cuerpo: {datos[2]}")
    memoria_promts += (f"Respuesta: {datos[3]}")



# Lee el archivo de Excel
# con doble barra el camino
#df2 = pd.read_excel('C:\\Users\\jacobozf\\Documents\\TEST\\mail_entrante.xlsx', sheet_name='Sheet1')
#nuevo_mail = df2.values


def on_yes():
    show_input_screen()

def on_no():
```

```
    messagebox.showinfo("Info", "Por favor, introduce el archivo en la carpeta antes de continuar.")
    # Implement logic to check if the file is in the folder and update the message accordingly.

def show_input_screen():
    welcome_label.pack_forget()
    button_frame.pack_forget()

    analyzed_label.pack()
    emisor_label.pack()
    emisor_entry.pack()
    asunto_label.pack()
    asunto_entry.pack()
    message_label.pack()
    message_entry.pack()
    generate_button.pack()

def generate_email():
    global emisor
    global asunto
    global message
    emisor = emisor_entry.get()
    asunto = asunto_entry.get()
    message = message_entry.get("1.0", "end-1c")

    app.destroy()  # Close the application window after generating the email.

app = tk.Tk()
app.title("Generador Automático de Respuestas de Correos Electrónicos")
app.geometry("800x500")

welcome_label = tk.Label(app, text="Bienvenido al generador automático de respuestas de correos
electrónicos")
welcome_label.pack(pady=20)
welcome_label = tk.Label(app, text="Has introducido en la carpeta el archivo con tu historico de correos
electronicos para que lo podamos analizar?")
```

```
welcome_label.pack(pady=20)

button_frame = tk.Frame(app)
button_frame.pack()
yes_button = tk.Button(button_frame, text="Sí", command=on_yes)
no_button = tk.Button(button_frame, text="No", command=on_no)
yes_button.pack(side=tk.LEFT, padx=10)
no_button.pack(side=tk.LEFT, padx=10)

analyzed_label = tk.Label(app, text="Ya hemos analizado tu estilo de escritura\nIntroduce a continuación el
correo que quieres que te ayudemos a contestar.")

emisor_label = tk.Label(app, text="Emisor:")
emisor_entry = tk.Entry(app)
emisor_entry.config(width=60)
asunto_label = tk.Label(app, text="Asunto:")
asunto_entry = tk.Entry(app)
asunto_entry.config(width=60)
message_label = tk.Label(app, text="Mensaje:")
message_entry = tk.Text(app, width=60, height=10)


generate_button = tk.Button(app, text="Generar", command=generate_email)

app.mainloop()

memoria_promts += (f"Responde a este correo como lo haria yo:")
memoria_promts += (f"Emisor: {emisor}")
memoria_promts += (f"Asunto: {asunto}")
memoria_promts += (f"Cuerpo: {message}")

numero_tokens = num_tokens_from_string(memoria_promts, "cl100k_base")
#print(numero_tokens)
#print(memoria_promts)

if numero_tokens < 3000:
```

```
    respuesta_sugerida = get_completion(memoria_promts)
    print(respuesta_sugerida)
else:
    promt_para_resumir = f"""Por favor, analiza el siguientes ejemplo de correo electronico \
    escritos por mi y proporciona información sobre sus comportamientos de escritura. Ten en cuenta mi estilo, estructura de oraciones, uso de vocabulario, nivel de formalidad, patrones gramaticales \
    y cualquier otro aspecto relevante que puedas identificar. A partir del ejemplo proporcionado, describe cómo crees que esta persona se expresa por escrito y qué características distintivas destacan en su estilo de escritura. \
    Proporciona tambien un resumen del email:
    email:
    emisor: <{matriz_emails[0][0]}>
    asunto: <{matriz_emails[0][1]}>
    contenido: <{matriz_emails[0][2]}>
    emisor: <{matriz_emails[0][3]}>
    '''
    iteracion = get_completion(promt_para_resumir)
    #print(iteracion)

    for i, j in enumerate(matriz_emails):
        promt_para_resumir_2 = f'''Aqui esta contenida informacion de como escribo correos electronicos: <{iteracion}>
        analiza el siguiente mail y actualiza la informacion de como escribo yo correos:
        emisor: <{matriz_emails[i][0]}>
        asunto: <{matriz_emails[i][1]}>
        contenido: <{matriz_emails[i][2]}>
        mi respuesta: <{matriz_emails[i][3]}>
        '''
        iteracion = get_completion(promt_para_resumir_2)
        #print(iteracion)

    promt_inicial_modificada = f'''
    Soy un estudiante de segundo de master en ingenieria industrial e industria conectada.\
    Las actividades que realizo a diario son estudiar, ir a clase y realizar trabajos en grupo.
```

```
    A continuación te voy a dar lista de cosas que describe como escribo.\
    Tu tarea es  generar una sugerencia de respuesta de forma automática. Al email que tambien esta debajo\
    Que tu respuesta sea solo la respuesta al email.\
    Como escribo: <{iteracion}>
    Mail que quiero que respondas:
    <"Emisor: {emisor}")
    Asunto: {asunto}")
    Cuerpo: {message}")>
    Solo quiero que me des de respuesta la respuesta que darias al mail.
    '''
    respuesta_sugerida = get_completion(promt_inicial_modificada)
    #print(respuesta_sugerida)



def guardar_respuesta():
    global respuesta_final
    respuesta_final = entry_respuesta.get("1.0", tk.END)
    root.destroy()

# Crear la ventana principal
root = tk.Tk()

# Configurar las dimensiones y posición de la ventana
root.geometry("800x800")
root.title(f"Sugerencia de respuesta")

# Definir la respuesta del correo electrónico
respuesta = respuesta_sugerida

# Crear la etiqueta de la respuesta
label_respuesta = tk.Label(root, text="Respuesta:")
label_respuesta.pack()

# Crear el campo de texto para la respuesta
```

```
entry_respuesta = tk.Text(root, height=30, width=50)  # Aumenta el valor de height para hacer la caja de texto más grande
entry_respuesta.insert(tk.END, respuesta)
entry_respuesta.pack()

# Crear el botón "Enviar"
label_respuesta = tk.Label(root, text="Modifica o Envía el correo!")
label_respuesta.pack()
boton_enviar = tk.Button(root, text="Enviar", command=guardar_respuesta)
boton_enviar.pack(pady=10)


# Mostrar la ventana
root.mainloop()

# Imprimir la respuesta final seleccionada o modificada
#print("Respuesta final:", respuesta_final)


# Define los nuevos valores
nuevo_emisor = emisor
nuevo_asunto = asunto
nuevo_contenido = message
nueva_respuesta = respuesta_final

# Agrega los nuevos valores en la última fila del DataFrame
df.loc[ultima_fila] = [nuevo_emisor, nuevo_asunto, nuevo_contenido, nueva_respuesta]

# Guarda el DataFrame actualizado en el archivo de Excel
df.to_excel('C:\\Users\\jacobozf\\Documents\\TEST\\repositorio_de_mails.xlsx', sheet_name='Sheet1', index=False)
```

ANEXO II. ALINEAMIENTO CON LOS OBJETIVOS DE DESARROLLO SOSTENIBLE

Examinamos a continuación cada uno de los objetivos de desarrollo sostenible que hemos considerado que este trabajo ayuda a impulsar. Destacaremos su relevancia en el contexto actual y como la inteligencia artificial y la innovación desempeñan un papel crucial en la consecución de estos objetivos. La relación que tienen con el desarrollo tecnológico nos permite visualizar a su vez nuevas estrategias y oportunidad hacia un futuro mas sostenible. Se espera por lo tanto tener una compresión solida de como la automatización de procesos basados en LLMs puede ser un importante camino hacia la consecución de estos objetivos.

Los tres ODS que se relacionan con los objetivos del proyecto son el ODS 8, el ODS 9 y el ODS 12.

- El ODS 8 - Trabajo decente y crecimiento económico.

  Se busca promover el crecimiento económico para todas las personas. Los avances tecnológicos que supone la automatización de tareas en el ámbito industrial pueden contribuir a una mayor eficiencia y productividad, lo que puede resultar en un aumento del empleo y de la calidad del trabajo en el sector. Además, la utilización de LLMs puede permitir la toma de decisiones más precisas y rápidas, lo que puede mejorar la eficiencia y reducir los errores en el lugar de trabajo.
- El ODS 9 - Industria, Innovación e Infraestructura.

  Tiene como objetivo fomentar la innovación y el desarrollo de tecnologías sostenibles para mejorar la calidad de vida de las personas. El uso de LLMs en la industria puede permitir el desarrollo de tecnologías más avanzadas y eficientes, lo que puede conducir a una mayor innovación y avance en la industria. Además, la automatización de tareas en el ámbito industrial puede mejorar la eficiencia y reducir los costes, lo que puede conducir a una mayor productividad y rentabilidad.

- El ODS 12 - Producción y consumo-responsables.

  Busca garantizar un consumo y producción sostenibles y responsables. La utilización de LLMs puede permitir una mejor toma de decisiones y análisis de datos, lo que puede llevar a una mejor comprensión de las necesidades y preferencias de los consumidores y una producción más responsable y sostenible. La automatización de tareas en industriales puede contribuir a una mayor eficiencia y productividad, lo que puede reducir los desperdicios y la producción innecesaria de productos.