Síntesis de texto usando modelos del lenguaje

Text synthesis using language models

Alejo Torres
atorres686@ucasal.edu.ar

Ingeniería en Informática, Facultad de Ingeniería, UCASAL

Resumen

Este artículo presenta un proyecto orientado a explotar las capacidades inherentes de los modelos de inteligencia artificial enfocados en el procesamiento del lenguaje natural con el objetivo final de sintetizar información contenida en grandes volúmenes de texto. El enfoque de este puesto es desarrollar un flujo de inferencia que permita al usuario generar resúmenes a partir de documentos de texto, al exponerlo como un servicio web para luego ser consumido por una interfaz intuitiva que se adapte a los dispositivos más populares y de uso común.

Palabras clave: *machine learning*, síntesis de texto, *large language models, transformer*.

Abstract

This article presents a project aimed at leveraging the inherent capabilities of artificial intelligence models focused on natural language processing, striving to perfect a method to synthesize information contained in large volumes of text. It is focused on developing an inference pipeline that allows the user to generate summaries from a text corpus, exposing it as a web service for consumption through an intuitive interface that adapts to the most popular and commonly used devices.

Key words: machine learning, text synthesis, large language models, transformer

Introducción

Los modelos de inteligencia artificial enfocados a la generación de contenido, también llamados inteligencias artificiales generativas, sin duda alguna han revolucionado el paradigma de trabajo de sectores completos. La tecnología es un hecho y existen numerosos productos y servicios que integran su potencial. Los *large language models* (LLM, por su sigla en inglés) o grandes modelos de lenguaje, que generan contenido del tipo *text-to-text*, son el motor que impulsa numerosas aplicaciones de tendencia actual. *ChatBots*, asistentes virtuales y motores de búsqueda personalizados son algunos ejemplos de estos. Sin embargo, los LLM comprenden una potente herramienta para el tratamiento de la información. Al ser capaces de procesar lenguaje natural, es posible utilizar estos modelos generalistas en tareas de extracción y síntesis de contenido.

Hemos visto el surgimiento de *startups* tecnológicas basadas por completo en esta esfera de conocimiento, generando aún más especulación y entusiasmo por el alcance que pueden lograr los productos y servicios que integran de manera exitosa las virtudes de la inteligencia artificial. [1]

Un área interesante para explotar es la del tratamiento de información de manera automática, al aprovechar las capacidades de procesamiento del lenguaje natural que brindan los grandes modelos de lenguaje. La capacidad de condensar la información relevante es fundamental para un correcto proceso de estudio sobre cualquier tema.

En una era en la cual la información abunda y contamos con acceso casi instantáneo a grandes volúmenes de datos, la capacidad de síntesis se vuelve aún más importante.

La iniciativa de abordar este tema surge gracias a la oportunidad que me fue brindada por la Universidad Católica de Salta en conjunto con la Universidad Politécnica de Madrid. Donde pasé un semestre cursando las asignaturas de Sistemas Inteligentes, Aprendizaje Automático II y Minería de Datos, las cuales las dictaban los profesores Alberto Díaz Álvarez y el Dr. Francisco Serradilla García. Sus excelentes cátedras me sirvieron a modo de introducción y pude adquirir las bases necesarias para continuar el aprendizaje de forma autónoma. Debido a esa experiencia decidí inclinarme por las IA generativas, en específico aquellos modelos basados en la arquitectura *transformer*.

Durante este proceso de investigación y de asimilación de conocimientos se despertó en mí la curiosidad por el desarrollo de aplicaciones que utilicen estas tecnologías a su favor. Así fue como pude tomar noción del alcance real que se puede lograr si se utilizan estas técnicas a fin de agilizar y modernizar los procesos de enseñanza y estudio. El objetivo final de esta herramienta es facilitar la asimilación de conocimiento, tanto para personas involucradas en tareas de investigación como estudiantes de cualquier nivel.

1. Desarrollo

1.1. Introducción

El trabajo implica, en primera instancia, una investigación y familiarización con las técnicas y tecnologías involucradas, como lo son la arquitectura *transformer*, la cual puede definirse como una red neuronal que aprende contexto y, por lo tanto, también significado, mediante el seguimiento de relaciones en datos secuenciales, como las palabras de esta oración [2]. Esta arquitectura de red neuronal funciona como piedra angular para los LLM.

Además, se aborda la revisión de librerías, proyectos y estudios relacionados con el procesamiento del lenguaje natural, cuyas siglas en inglés son NLP; técnicas de *clustering* y *machine learning,* al desglosar los conceptos hasta el punto que puedan comprenderse y utilizarse para la construcción del *software* de síntesis de texto.

Se trabaja también en el diseño de la arquitectura de los servicios de inferencia del sistema, donde se integrarán todas estas herramientas para lograr el objetivo del *software*. Sobre este diseño se evaluará su rendimiento a través de la generación de resúmenes de temas específicos que luego serán evaluados por profesionales en la materia. Gracias a esto se podrá perfeccionar las directivas y los parámetros de los modelos.

La metodología que se utilizará en el proyecto es la de desarrollo en cascada [3]. La elección de esta metodología se debe a que, en primer lugar, el desarrollo completo del proyecto lo llevará a cabo una sola persona, por lo que se opta por un modelo que simplifique el trabajo y no esté orientado a la intercomunicación de diferentes equipos. Además, es importante aclarar que para el desarrollo de aplicaciones y modelos de IA resulta útil una flexibilidad que puede no encajar en marcos metodológicos convencionales. Sin embargo, en este caso particular se optó por la tercerización de los servicios de inferencia, por lo que es posible ajustarse a un modelo más clásico de desarrollo de *software*.

El modelo en cascada es uno de los enfoques más antiguos y tradicionales para la gestión de proyectos de desarrollo de *software* y se caracteriza por su enfoque lineal y secuencial en el que

cada fase debe completarse antes de que pueda comenzar la siguiente.

1.2. Objetivo

En cuanto al desarrollo del sistema, el objetivo es analizar, diseñar y construir un prototipo de aplicación que permita resumir el texto de un archivo PDF con una extensión que no supere las 300 páginas, al utilizar modelos del lenguaje.

En la actualidad, existen herramientas disponibles para realizar resúmenes de texto que aplican técnicas que no necesariamente involucran en su totalidad a la inteligencia artificial, pero sí son consideradas técnicas de *machine learning* y NLP. Entre ellas podemos nombrar al método LSA (*latent semantic analysis*, en inglés), el cual utiliza técnicas de reducción de dimensionalidad a fin de encontrar relaciones semánticas entre palabras y frases en un texto. Luego, selecciona las frases más significativas para formar el resumen, los algoritmos de *ranking* de oraciones, los cuales se basan en asignar puntajes a cada oración en función de su importancia dentro del texto. Posteriormente, escoge las oraciones con los puntajes más altos para construir el resumen. Los algoritmos de agrupamiento de texto consisten en conformar agrupaciones con las frases o párrafos similares en temas y seleccionar representantes de cada grupo como parte del resumen. El uso de estructuras gramaticales donde se identifica la estructura gramatical del texto, como encabezados, subencabezados y listas numeradas, se utilizan con el objetivo de crear un resumen estructurado.

Las herramientas que incorporan las ventajas que el *deep learning* junto con los LLM son producto de la creciente popularidad de esta tecnología y, por lo tanto, son relativamente nuevas. Gracias a la naturaleza de estos modelos y su capacidad de simular el lenguaje humano, los resultados tienden a ser de mejor calidad, al mostrar una mayor atención en los detalles y en el contexto del cual se extrae el texto a resumir. Esto establece a los sintetizadores de texto basados en modelos del lenguaje como una fuerte alternativa, que además aporta una fácil incorporación en aplicaciones web ya existentes debido a la flexibilidad que brindan los marcos de trabajo que giran en torno a esta temática.

Teniendo en cuenta esto, se identifican los requisitos clave que debe cumplir el prototipo del sistema.

El sistema debe permitir al usuario seleccionar un archivo solo en formato PDF, desde el sistema de archivos de su computadora.

- El sistema debe poder procesar y extraer todo el texto del archivo.
- El sistema debe permitir al usuario modificar los hiperparámetros involucrados en el proceso de inferencia y síntesis del texto.
- El sistema debe presentar el resumen al usuario como un archivo en formato PDF.
- El sistema debe informar de manera clara y en un lenguaje comprensible las excepciones en el flujo de inferencia causadas por terceros o mala configuración de los hiperparámetros.
- El sistema debe ser accesible tanto en computadoras como en dispositivos móviles.

1.3. Tecnologías involucradas

Gracias a la comunidad de desarrollo de código abierto y al gran interés que gira en torno a este campo de estudio, existen marcos de trabajo y librerías capaces de facilitar las tareas de desarrollo brindando abstracciones robustas y extensibles que resuelven en gran medida la mayoría de los aspectos engorrosos de construir aplicaciones impulsadas por la inteligencia artificial. Entre ellas podemos destacar:

- *Large Language Model*: La pieza fundamental del sistema, que se encarga de realizar la síntesis del texto y todas las tareas de inferencia necesarias a fin de lograr un resumen total y adecuado, tanto en contexto como en semántica. El tamaño de estos modelos hace que su ejecución requiera de mucho poder computacional y recursos de *hardware*, por lo que la solución más factible para alcanzar la mejor calidad posible en los resúimes a realizar es optar por un servicio de inferencia para tercerizar el poder de procesamiento necesario. [4]
- Langchain: Es una poderosa librería con implementaciones en Python y Javascript, que cuenta con abstracciones a fin de facilitar la implementación de aplicaciones impulsadas por IA, introduce el concepto de cadenas o *chains* que permiten agrupar los elementos involucrados sobre la inferencia del modelo, personalizar parámetros, directivas e incluso el propio motor de inferencia, generando así una interfaz común que brinda soporte para los clientes específicos de cada uno de los servicios de inferencia más populares como lo son OpenAI, StreamLit, HuggingFace etc. [5]
- SkLearn: scikit-learn, también conocido como sklearn, es una de las bibliotecas de aprendizaje automático más populares y ampliamente utilizadas en el lenguaje de programación Python. Esta biblioteca proporciona herramientas eficientes y fáciles de usar para

llevar a cabo tareas relacionadas con el aprendizaje automático, como la clasificación, la regresión, el agrupamiento, la reducción de dimensionalidad y demás algoritmos de *machine learning*. [6]

» FastAPI: Es un marco de desarrollo web de alto rendimiento y fácil de usar que se utiliza para crear aplicaciones web y API. Fue diseñado para ayudar a los desarrolladores a construir aplicaciones rápidas y escalables con una sintaxis clara y una documentación automática. Gracias al tipado estático aprovecha el poder de librerías de modelado de datos como Pydantic y posee una serie de *middlewares* que facilitan la construcción de funcionalidades web.[7]

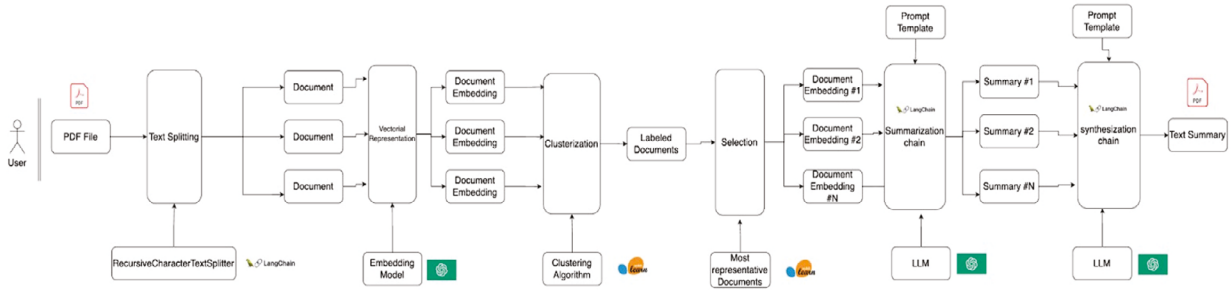

Figura 1: Diagrama de flujo del prototipo para la aplicación de síntesis de texto

» ReactJS: Es una biblioteca de JavaScript de código abierto, que se utiliza con el objetivo de construir interfaces de usuario interactivas y dinámicas en aplicaciones web. Desarrollada y mantenida por Facebook, React se ha convertido en una de las herramientas más populares en el desarrollo web moderno debido a su enfoque en la construcción eficiente y modular de componentes reutilizables. En lugar de abordar la creación de una aplicación web como un conjunto de páginas separadas, React se centra en la creación de componentes autónomos y encapsulados de manera individual. Estos componentes pueden ser pequeñas piezas de la interfaz de usuario, como botones, formularios o elementos más grandes, tales como encabezados y paneles de navegación. Cada componente puede tener su propia lógica y estado interno, lo que facilita la gestión y la reutilización en diferentes partes de la aplicación.

1.4. Análisis y diseño del prototipo

A continuación, se muestra un diagrama del flujo de trabajo del prototipo para la aplicación de síntesis de texto junto con las diferentes tecnologías y herramientas involucradas.

El flujo puede explicarse de la siguiente manera: Partiendo de un archivo provisionado por el usuario, se procederá a leer y extraer el texto plano del mismo. El proceso de validar la petición, la manipulación y la extracción del contenido del archivo PDF establece el *data pipeline,* que resulta en un corpus de texto manejable por las demás partes del sistema. Este luego se divide en fragmentos manejables denominados “Documentos”, al utilizar la implementación de Langchain llamada *RecursiveCharacterTextSplitter*, la cual permite dividir el texto con cierto solapamiento entre las piezas a fin de conservar parte del contexto y la continuidad del contenido. El siguiente paso es obtener una representación en un espacio vectorial denso al utilizar un modelo de *embeddings* [8], en este caso se utilizó servicios de terceros a través de la API de OpenAI [9]. Al implementar esta representación se aplicó un proceso de clusterización [10], al definir un algoritmo de agrupamiento gracias a la librería de SKLearn. Una vez realizadas las agrupaciones, se seleccionaron aquellos Documentos, una abstracción implementada por Langchain con el objetivo de manejar fragmentos de texto más representativos de cada agrupación. Al utilizar estos documentos generamos un resumen individual de cada uno de los documentos seleccionados, con la ayuda de un modelo de lenguaje y a través de la abstracción *chains* provista por el *framework* de Langchain. Una vez obtenidos estos resúmenes, es posible combinarlos para generar el resumen final al utilizar la cadena de síntesis.

1.5. Desarrollo del prototipo

El desarrollo del prototipo explicado en el apartado anterior se realizó al crear un entorno virtual del lenguaje de programación Python en su versión 3.11. Gracias a la herramienta Jupyter Notebooks, la cual nos permite prototipar y ex-

plorar soluciones de manera dinámica, ya que permite ejecutar código Python en formato de celdas, compartiendo la misma sesión del kernel y acompañándolo con celdas especiales a modo de documentación en formato Markdown. Se comenzó por poner en práctica diferentes técnicas de síntesis de texto en diferentes problemas, variando el tamaño del corpus de texto y la complejidad requerida del resumen, hasta llegar a la solución final. A continuación, se muestran imágenes de cada uno de los bloques de código, correspondientes a la solución final.

Figura 2: Fragmento de código I

Figura 3: Fragmento de código II

```python
# Loaders
from langchain.schema import Document

# Splitters
from langchain.text_splitter import RecursiveCharacterTextSplitter

# Model
from langchain.chat_models import ChatOpenAI

# Embedding Support
from langchain.vectorstores import FAISS
from langchain.embeddings import OpenAIEmbeddings

# Summarizer we'll use for Map Reduce
from langchain.chains.summarize import load_summarize_chain

# Data Science
import numpy as np
from sklearn.cluster import KMeans
```

```python
text_splitter = RecursiveCharacterTextSplitter(separators=["\n\n", "\n", "\t"],
chunk_size=10000, chunk_overlap=3000)

docs = text_splitter.create_documents([text])
```

Figura 4: Fragmento de código III

```python
text_splitter = RecursiveCharacterTextSplitter(separators=["\n\n", "\n", "\t"],
chunk_size=10000, chunk_overlap=3000)

docs = text_splitter.create_documents([text])
```

```python
num_documents = len(docs)

print (f"Now our book is split up into {num_documents} documents")
```

```
Now our book is split up into 78 documents
```

obtenemos las representaciones vectoriales de estos 78 documentos

```python
embeddings = OpenAIEmbeddings(openai_api_key=openai_api_key)

vectors = embeddings.embed_documents([x.page_content for x in docs])
```

Para agrupar los documentos usaremos el algoritmo de KMedias

```python
num_clusters = 11

# Perform K-means clustering
kmeans = KMeans(n_clusters=num_clusters, random_state=42).fit(vectors)
```

Figura 5: Fragmento de código VI

visualizamos las las agrupaciones formadas

```python
kmeans.labels_
```

```
array([ 2,  2,  2,  2,  2,  2,  6,  9,  9,  4,  4,  4,  4,  9,  9,  9,  9,
        4,  4,  4,  4,  9,  9,  5,  4,  0,  0,  0,  5,  5,  5,  9,  9,  3,
        3,  9,  3,  3,  3,  3,  3,  9,  9,  9,  3,  3,  3,  3,  7,  7,  7,
        7,  7,  6,  6,  6,  6,  6,  6,  9,  1,  1,  3,  4,  4,  1,  1,  1,
        1,  9, 10, 10, 10,  8,  8,  8,  8,  8], dtype=int32)
```

Figura 6: Fragmento de código V

Vamos a realizar una reducción de la dimensionalidad de estos vectores para poder visualizarlos en un plano 2D

```python
from sklearn.manifold import TSNE
import matplotlib.pyplot as plt

# Taking out the warnings
import warnings
from warnings import simplefilter

# Filter out FutureWarnings
simplefilter(action='ignore', category=FutureWarning)

# Perform t-SNE and reduce to 2 dimensions
tsne = TSNE(n_components=2, random_state=42)
reduced_data_tsne = tsne.fit_transform(vectors)

# Plot the reduced data
plt.scatter(reduced_data_tsne[:, 0], reduced_data_tsne[:, 1], c=kmeans.labels_)
plt.xlabel('Dimension 1')
plt.ylabel('Dimension 2')
plt.title('Book Embeddings Clustered')
plt.show()
```

Figura 7: Fragmento de código VI

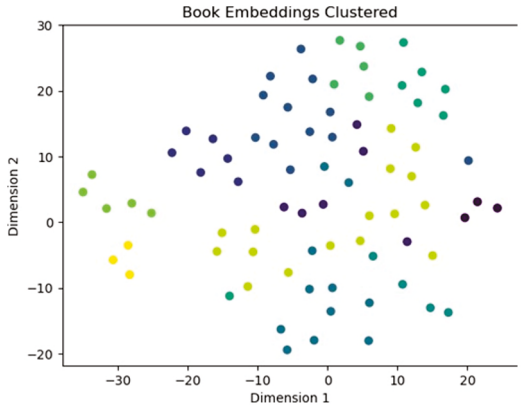

Figura 8: Visualización del clúster de embeddings

Podemos observar algunas agrupaciones claras

El siguiente paso es obtener aquellos documentos mas representativos de cada cluster

```python
closest_indices = []

# Loop through the number of clusters you have
for i in range(num_clusters):

    # Get the list of distances from that particular cluster center
    distances = np.linalg.norm(vectors - kmeans.cluster_centers_[i], axis=1)

    # Find the list position of the closest one (using argmin to find the
    smallest distance)
    closest_index = np.argmin(distances)

    # Append that position to your closest indices list
    closest_indices.append(closest_index)
```

MagicPython

Figura 9: Fragmento de código VII

ahora debemos ordenarlos para evitar perder el orden del texto original

```
selected_indices = sorted(closest_indices)
selected_indices
```

```
[0, 12, 26, 29, 39, 41, 51, 54, 65, 71, 75]
```

Lo que haremos a continuación sera resumir cada uno de los documentos seleccionados, para luego combinarlos y obtener el resumen final

```
llm3 = ChatOpenAI(temperature=0,
                  openai_api_key=openai_api_key,
                  max_tokens=1000,
                  model='gpt-3.5-turbo'
                 )
```

Figura 10: Fragmento de código VIII

```
map_prompt = """
You will be given a single passage of a book. This section will be enclosed in
triple backticks (```)
Your goal is to give a summary of this section so that a reader will have a
full understanding of what happened.
Your response should be at least three paragraphs and fully encompass what was
said in the passage.

```{text}```
FULL SUMMARY:
"""
map_prompt_template = PromptTemplate(template=map_prompt, input_variables=
["text"])
```

```
map_chain = load_summarize_chain(llm=llm3,
                                 chain_type="stuff",
                                 prompt=map_prompt_template)
```

obtenemos los documentos en base a los indices seleccionados

```
selected_docs = [docs[doc] for doc in selected_indices]
```

Figura 11: Fragmento de código IX

iteramos sobre ellos y almacenamos los resumenes individuales en una lista

```
summary_list = []

# Loop through a range of the lenght of your selected docs
for i, doc in enumerate(selected_docs):

    # Go get a summary of the chunk
    chunk_summary = map_chain.run([doc])

    # Append that summary to your list
    summary_list.append(chunk_summary)

    print (f"Summary #{i} (chunk #{selected_indices[i]}) - Preview:
    {chunk_summary[:250]} \n")
```

Figura 12: Fragmento de código X
```

concatenamos los resúmenes en orden

```python
summaries = "\n".join(summary_list)

# Convert it back to a document
summaries = Document(page_content=summaries)

print (f"Your total summary has {llm.get_num_tokens(summaries.page_content)} tokens")
```

[51]

MagicPython

```
Your total summary has 4002 tokens
```

Figura 13: Fragmento de código XI

```python
llm4 = ChatOpenAI(temperature=0,
                  openai_api_key=openai_api_key,
                  max_tokens=3000,
                  model='gpt-4',
                  request_timeout=120
                 )
```

+ Code + Markdown

```python
combine_prompt = """
You will be given a series of summaries from a book. The summaries will be enclosed in triple backticks (```)
Your goal is to give a verbose summary of what happened in the story.
The reader should be able to grasp what happened in the book.

```{text}```
VERBOSE SUMMARY:
"""
combine_prompt_template = PromptTemplate(template=combine_prompt, input_variables=["text"])
```

```python
reduce_chain = load_summarize_chain(llm=llm4,
                                    chain_type="stuff",
                                    prompt=combine_prompt_template,
#                                     verbose=True # Set this to true if you want to see the inner workings
                                    )
```

Figura 14: Fragmento de código XII

1.6. Desarrollo de la interfaz de usuario

A continuación, se muestran capturas del prototipo de la interfaz web de usuario, construida utilizando la librería de desarrollo web *ReactJs*, la cual también cuenta con un modo *responsive* para adaptarse a cualquier tamaño de pantalla del dispositivo.
```

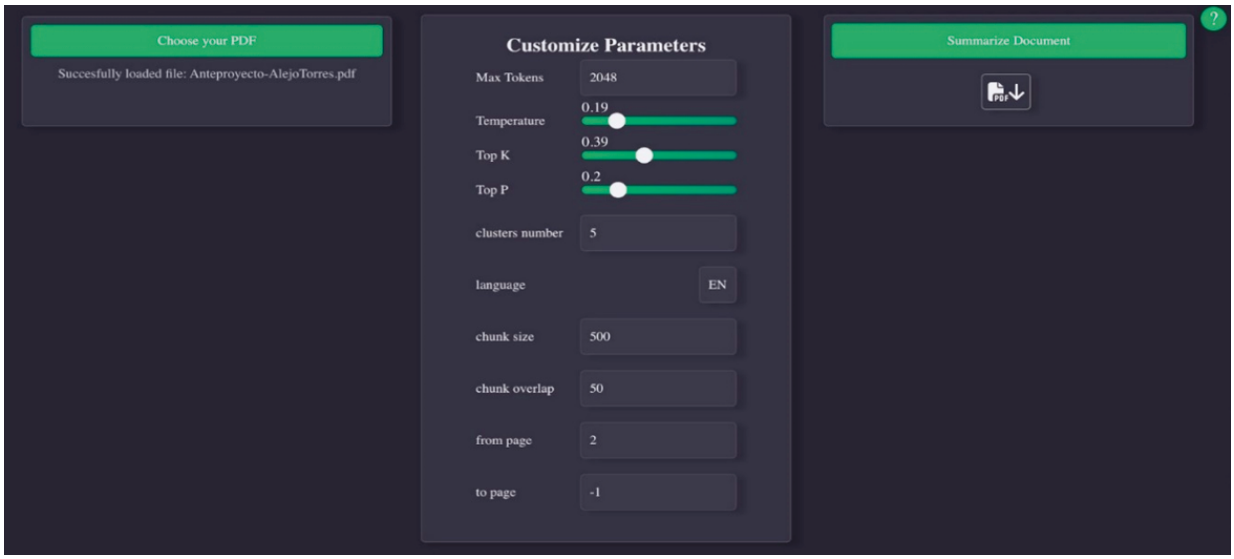

Figura 15: Interfaz de usuario I

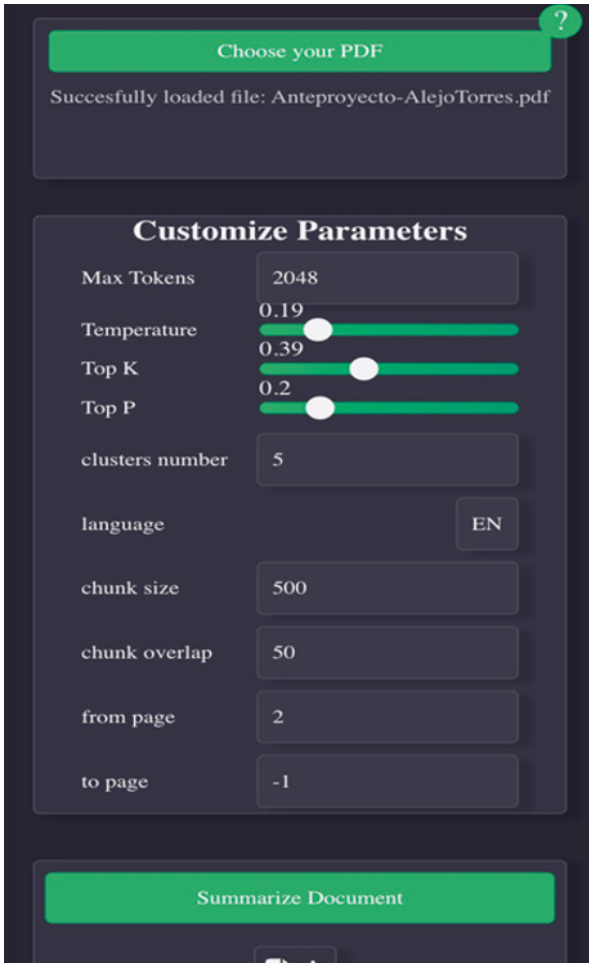

Figura 16: Interfaz de usuario II

2. Resultados

Se detalla una prueba realizada sobre el sistema en un ambiente preproductivo, el mismo puede explicarse mediante el siguiente diagrama, en el cual se tomaron los servicios desarrollados en el prototipo a fin de construir una API web que expone el flujo de inferencia a través del protocolo de comunicación HTTP.

La prueba se realizó utilizando el libro “Into the Air” [13] de alrededor de 350 páginas:

“In this story, the author recounts their experience as part of a New Zealand-based team attempting to summit Mount Everest on May 10, 1996. Despite months of anticipation, the author finds themselves unable to fully appreciate the achievement due to extreme exhaustion, lack of sleep, and the effects of high altitude. After a brief moment at the summit, the author begins their descent, encountering a traffic jam of climbers from multiple expeditions. The author’s oxygen tank runs out, forcing them to navigate dangerous terrain without supplemental oxygen. Unbeknownst to the climbers, a disaster is about to unfold, resulting in the deaths of eight climbers, including two of the author’s teammates.

The story also highlights the challenges and dangers of high- altitude mountaineering, as well as the importance of teamwork and respect for local cultures and customs. The author and their team face various obstacles, including altitude sickness, treacherous terrain, and the reliance on Sherpas, who work incredibly hard for little pay. The author emphasizes the need to show gratitude and respect to the Sherpas, who are essential to the success of the climbers.

Throughout the story, the author describes various events and challenges faced by the

climbers, including medical emergencies, the importance of proper acclimatization, and the difficulties of providing medical care in a remote, high-altitude environment. The author also discusses the cultural differences between Western climbers and Sherpas, who are often reluctant to acknowledge physical infirmities due to an element of machismo in their culture.

As the climbers continue their ascent, they face physical and mental challenges, as well as the individualistic nature of the climb. The author emphasizes the importance of timing and weather conditions in determining the success of a summit assault. The story also touches on the controversy surrounding the author's book, 'Into Thin Air,'and its portrayal of Anatoli Boukreev, a professional climbing guide involved in the 1996 Everest disaster.

In the aftermath of the tragedy, the author reflects on the emotional impact, including survivor's guilt and the grief of the families and loved ones of the deceased. The author also considers the role of hubris in the disaster and questions their own actions during the storm. The story highlights the complex emotions and difficult questions that arise in the wake of such a devastating event, as well as the contentious nature of reporting on a tragedy and the importance of accuracy and thorough research in journalism."

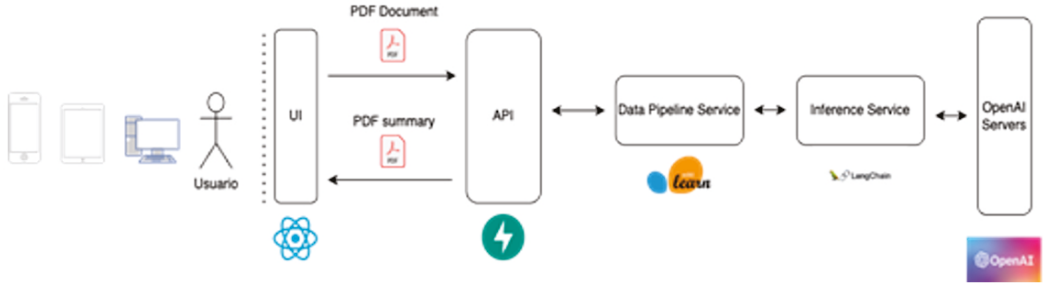

Figura 17: Diagrama del sistema

3. Conclusiones

"Inteligencia artificial", "modelos generativos" y "ChatGPT" son palabras tendencia que están en la boca de todo el mundo hoy en día. Tanto en usuarios con formación técnica, como aquellos quienes aprovechan las ventajas de esta tecnología sin preocuparse realmente por su trasfondo. A través de este trabajo es posible darse cuenta que la tecnología ya es un hecho y son numerosas las aristas sobre las cuales puede aplicarse. En el campo del procesamiento del lenguaje natural, los modelos del lenguaje y su arquitectura inherente demuestran ser uno de los avances del *deep learning* con más potencial para ser explotados, al ser su uso en el tratamiento automático de la información uno de los tantos campos para explorar que brinda resultados más que satisfactorios.

4. Antecedentes relacionados

El problema de la síntesis de texto es un desafío importante en el campo del procesamiento del lenguaje natural y el aprendizaje profundo. Antes de la aparición de las técnicas de *deep learning*, la generación de texto, ya que el problema en sí se basa en generar un resumen, estaba impulsada principalmente por enfoques basados en reglas y modelos de lenguaje estadísticos. Sin embargo, la llegada del *deep learning* revolucionó la forma en que se aborda este problema, permitiendo generar un texto más coherente y contextualmente relevante.

Hoy en día la tendencia está en el uso de LLM. Existe una gran variedad de modelos del tipo *open source* con *fine- tunnings* que realiza la comunidad con esta tarea en mente, los mismos pueden encontrarse en el portal de Hugging Face y pueden utilizarse a través de la librería *transformer*. Sin embargo la usabilidad de estos modelos se ve limitada por el *hardware* en el que se ejecutan. Por lo tanto, es casi imposible alcanzar las velocidades y rendimiento de servicios *freemium* de inferencia al utilizar *hardware* de grado de consumidor final. Si bien es cierto que técnicas como la cuantización de los modelos o la ejecución de estos a través del protocolo de Petals se acerca un poco más , todavía resulta una tarea ardua para este tipo de *hardware*.

Referencias

[1] M. Law. "Top 10: Promising AI Startups in 2022". 2022. [En línea] Disponible en https://aimagazine.com/articles/top-10- promising-ai-startups-in-2022

[2] R. Merrit. "¿Qué es un modelo modelo *transformer*?". Blog de NVIDIA. *Blog oficial de NVIDIA Latino América*. 2022. [En línea] Disponible en https://la.blogs.nvidia.com/2022/04/19/que-es-un-modelo- transformer/

[3] C. I. Rivas, V. P. Corona, J. F. Gutiérrez, L. Hernández. "Metodologías actuales de desarrollo de *software*". *Revista de Tecnología e Innovación*. 2015.

[4] A. Álvarez Días. "Procesamiento de lenguaje natural (NLP). Departamento de Sistemas Informáticos". *Universidad Politécnica de Madrid*. 2023.

[5] LangChain, v.0.0.1, LangChain Inc., San Francisco, CA, USA, 2023. [En línea] Disponible en: https://www.langchain.com/. [Accedido: 24 de octubre de 2023].

[6] "scikit-learn: Machine Learning in Python — scikit-learn 1.3.2 Documentation," scikit-learn, 2023. [En línea] Disponible en: Disponible en: https://scikit-learn.org/stable. [Accedido: 24 de octubre de 2023].

[7] "FastAPI," [En línea]. Disponible en: https://fastapi.tiangolo.com/. [Accedido: 24 de octubre de 2023].