Universidad Zaragoza
1542

Trabajo Fin de Grado

Representación mediante clusters de similaridad semántica, generada con modelos de lenguaje

Clustering based on semantic similarity, generated by language models

Autor

Alberto Pérez Blasco

Directora

María Piedad Garrido Picazo

Escuela Universitaria Politécnica de Teruel
2022

Tabla de Contenidos

Agradecimientos

Este trabajo ha sido realizado dentro del área de Innovation & Strategic Investments (I&SI) en el grupo de A2DC de la empresa NTT Data en Aragón. Como se indica en la memoria, se han utilizado aceleradores y componentes generados en el entorno KNIME por parte de miembros del área/grupo. La propiedad de estos flujos de trabajo (Workflows) y componentes es única y exclusiva de la empresa. Por todo esto, me gustaría agradecer a los compañeros de la empresa, especialmente a Diego Romero San Martín, Alejandro Ruberte Sanz y Miguel Salmerón Marco por toda la ayuda brindada, así como por todos los conocimientos aprendidos durante el periodo de prácticas, ya que sin su apoyo y consejos, no hubiese sido posible.

Por otro lado, quiero agradecer a mi tutora del TFG María Piedad Garrido Picazo, por toda la ayuda que me ha ofrecido, ya no solo a nivel académico, y finalmente a mi familia y amigos por estar siempre presentes y ayudándome.

Resumen y Palabras Clave

Este Trabajo de Fin de Grado (TFG), trata de obtener documentos de una fuente de internet, para posteriormente clasificarlos en clústeres dependiendo de su similaridad semántica y representarlos, de tal modo que se pueda analizar su parecido. Para ello, se utilizarán modelos de lenguaje que tratarán de entender el contexto del documento, generando los embeddings asociados.
Como documentos de entrada, se utilizarán papers de caracter científico-técnicos y para la salida, se deberán visualizar los distintos clústeres, ya que es la mejor forma de hacer referencia al conjunto de documentos de entrada, debido a que de otra forma, es difícil visualizar la relación entre los documentos descargados (aprox. 800). Finalmente, se obtendrá un "topic" que sirva para describir cada clúster.
Al ser un TFG orientado a la investigación, el desarrollo del mismo tuvo varias fases, siendo éstas la de formación en conocimientos básicos relacionados con las tecnologías, la de investigación de las opciones que existían en cada apartado y finalmente la de implementación, fase en la que se desarrolló el proyecto con tecnologías como KNIME, Jupyter Notebook, Python, Conda...

This TFG, addresses the extraction of documents from an information source and, later, its classification in clusters depending on his semantic similarity and associating it to its representation so that the similarity between them can be analysed. To reach this purpose, language models will be used to understand the document context by generating the appropiate embeddings.
As input documents, scientific-technical papers will be used and, for the output, the way of visualization will be the different clusters because is the best option to refer to all input documents (it is difficult to visualize the similarity between all input documents), using a topic to describe them.
Finally, the TFG, was an investigation-oriented project, so the development was aimed in three stages being the first one the formation in basic subjects related to the technologies, the research phase, wich dealt to search the possibilities for every part, and the last one, the implementation, phase in wich the project was developed with technologies like KNIME, Jupyter Notebook, Python, Conda...

Palabras clave (spa)

- **KNIME.** Programa "Open Source" destinado a la rama de "Data Science", para desarrollar flujos de trabajo (Workflows) mediante programación visual, arrastrando los componentes e interconectándolos.
- **Token.** Unidad mínima de división de textos para poder ser interpretados por un modelo de lenguaje.
- **GPT-3.** Modelo de lenguaje autorregresivo, que trata de generar textos que simulan la redacción humana.
- **Clúster.** Agrupación de un conjunto de datos con características similares.
- **Modelo de lenguaje.** Modelo de aprendizaje automático que trata de predecir la siguiente palabra dentro de una oración, teniendo en cuenta el contexto de las palabras anteriores.

Keywords (eng)

- **KNIME.** Open Source program for Data Science, that let you build Workflows using Visual Programming with drag and drop style.
- **Token.** Minimum unit of a separated text (word, characters, subwords...), that can be interpreted by a language model.
- **GPT-3.** Autoregressive language model that tries to generate human-like text.
- **Cluster.** Set of a dataset with similar features.
- **Language Model.** Machine learning model that tries to predict the next word of a phrase, considering the previous words.

1. Contextualización

La idea de este Trabajo de Fin de Grado (TFG), surge de realizar las prácticas en la empresa NTT Data, en el departamento dedicado a Inteligencia Artificial "A2DC".

El objetivo principal del proyecto, es el de generar herramientas internas que permitan un análisis y organización (mediante clústeres) de la información de la forma más rápida posible, incluyendo técnicas semánticas para el análisis de los documentos científico-técnicos. Otro objetivo a destacar es el de generar estas herramientas en plataformas de tipo "low-code" (KNIME), para la implementación final de la solución debido a que de esta forma, gente sin experiencia programando podrá operar los Workflows con distintas fuentes de información, al trabajar únicamente con archivos csv (ya que toda la lógica está programada en los distintos bloques y en pocos casos habría que adaptarlos), por lo que agilizaría los análisis posteriores.

Como primer paso, se realizó un proceso de formación general en Inteligencia Artificial (IA) y en programación con Python. Este proceso duró unas semanas hasta que posteriormente, se realizó uno más específico, orientado a la arquitectura de los transformers y clústeres, ya que eran apartados sobre los que se necesitan conocimientos previos.

Una vez realizado el proceso de formación, se procedió a la instalación y configuración de todas las herramientas que se utilizan en el departamento, para así facilitar el proceso de investigación e implementación de la solución final. Después de esto, se decidió dividir el proyecto en varias fases, para de esta forma dedicar tiempo a cada una de las mismas buscando la mejor opción y posteriormente, a su desarrollo. Al haber dependencia entre las mismas, no se podía avanzar sin tener totalmente finalizada cada fase.

Tras concluir la implementación de cada una de las fases, se realizó un proceso de readaptación de los distintos componentes para añadirles configuraciones extra, como por ejemplo para la selección de las columnas de entrada, el número de keywords a obtener por cada tema o las columnas que se quieren mostrar en la matriz. De esta forma, se generalizaron un poco más los Workflows (conjunto de componentes de KNIME que realizan una función) pudiendo adaptarse a otro tipo de soluciones. Finalmente, se va a proceder a comentar brevemente las secciones de las que consta la memoria, comenzando por un apartado de contextualización, en el que se explicarán brevemente los conocimientos previos necesarios para la realización del trabajo. Posteriormente, en el estado de la cuestión se comentarán algunos trabajos con características comunes al Trabajo de Fin de Grado. Los últimos apartados, serán los de análisis, diseño, implementación y resultados en los que se comentará todo el proceso y su resultado, finalizando con otro apartado de conclusiones.

A continuación, se van a explicar unos conceptos previos sobre los que fue necesario realizar un proceso de aprendizaje, para poder empezar a investigar sobre todo lo relacionado con el proyecto. Dichos conceptos son: token, embedding, transformer, modelo de lenguaje, clúster y learning rate y reducción de dimensiones.

Token

Un token, es la unidad mínima en la que se divide un texto, para poder ser interpretado por el transformer. Siendo los tokens, secuencias de caracteres encontradas en el texto. Para obtener los tokens, se utilizan herramientas de tokenización, que separan el texto en estas unidades. Un ejemplo sería la herramienta de OpenAI para GPT-3 (cada modelo de lenguaje suele tener su propia herramienta):

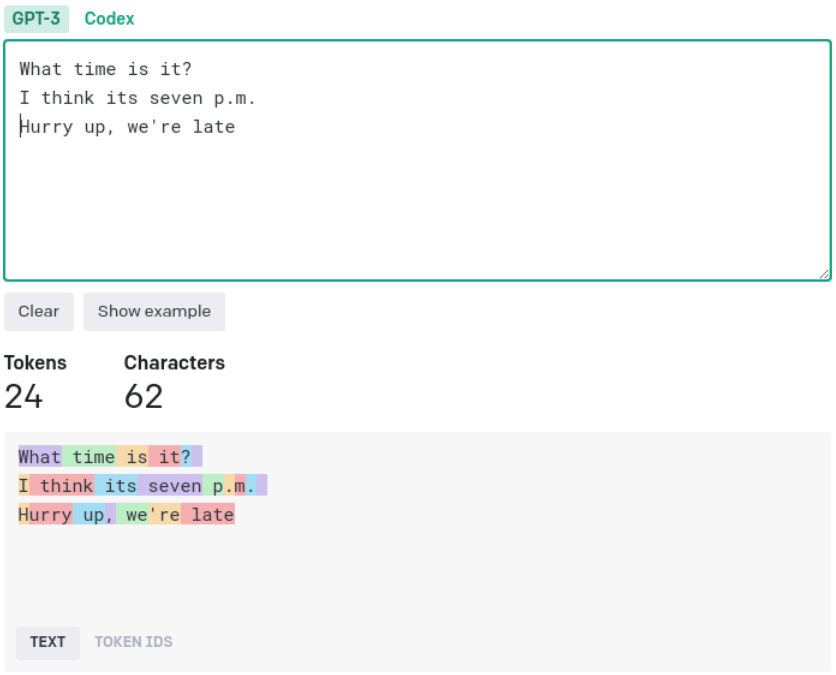

Figura 1: Ejemplo de tokenización para GPT-3

Finalmente, como se puede ver en la figura 1, se obtienen 24 tokens (cada uno siendo un color del texto) y se puede apreciar que la mayoría de palabras las entiende enteras y las convierte un token (al ser inglés, ya que todos estos modelos están preparados principalmente para el inglés), pero algunas palabras como “Hurry” o “p.m.” las interpreta como varios tokens.

Embedding

Un embedding, es la representación vectorial de cada token. Para obtener los embeddings, se utilizan algoritmos propios en cada modelo, debido a que están entrenados para ver las relaciones entre las palabras y que así de esta forma, cada grupo de palabras relacionadas entre sí, salgan en una zona del espacio vectorial cercanas. Un ejemplo de espacio vectorial para distintas palabras sería:

Figura 2: Ejemplo de un espacio vectorial

En la figura 2, se pueden ver distintas palabras representadas en el espacio vectorial, apreciándose como las palabras que tienen una relación se encuentran en una zona bastante cercana, por ejemplo los animales en el centro o las frutas en las zonas exteriores. [2]

Transformer

Los transformers [9] [16], son un tipo de arquitectura de red neuronal, que trata de gestionar la dependencia entre la entrada y salida de un texto basándose en la atención y en la recurrencia.

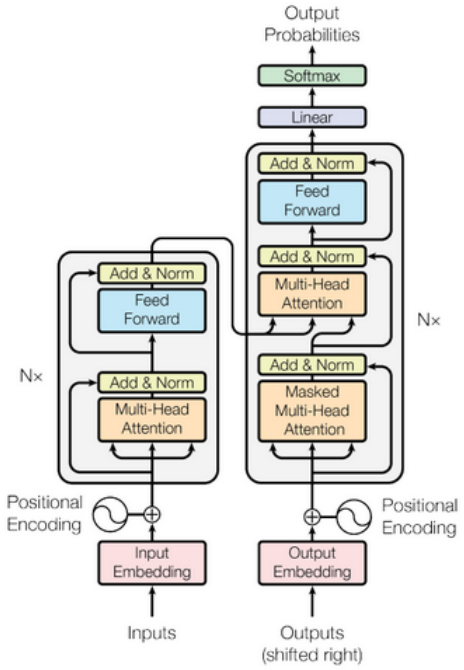

Figura 3: Arquitectura de un transformer

Esta estructura, se basa en un encoder y un decoder, utilizando el primero para analizar el contexto de la secuencia de entrada y el segundo para obtener la salida basada en el contexto anterior. En la figura 3, sería el encoder la parte de la izquierda y el decoder la de la derecha. Para la comunicación entre ambos, se utilizan los embeddings.

Otro aspecto a destacar de los transformers, es que para una entrada, se realizan los embeddings y una secuencia de codificaciones posicionales (representación vectorial de la palabra en la oración original), obteniendo de esta forma para una frase la posición de cada palabra y los embeddings para las mismas. Finalmente, para el mecanismo de atención, se toman 3 valores de entrada:

- Query (Q) que representa el valor de una entrada
- Keys (K) que son el resto de palabras de la secuencia
- V que es el valor vectorial de la palabra que se procesa

Para obtener la atención, se utilizará la siguiente fórmula:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

Figura 4: Fórmula para obtener la atención

En esta fórmula (Fig. 4), se realiza el producto escalar entre Q y K traspuesta, para estimar la alineación de los vectores (embeddings) devolviendo un peso para cada palabra del texto. Posteriormente, normalizando al dividir por la raíz del tamaño de k y se aplica softmax para conseguir el peso de la palabra entre un rango 0-1. Finalmente, se multiplicarán estos pesos por el valor para obtener las palabras importantes. Un ejemplo sería:

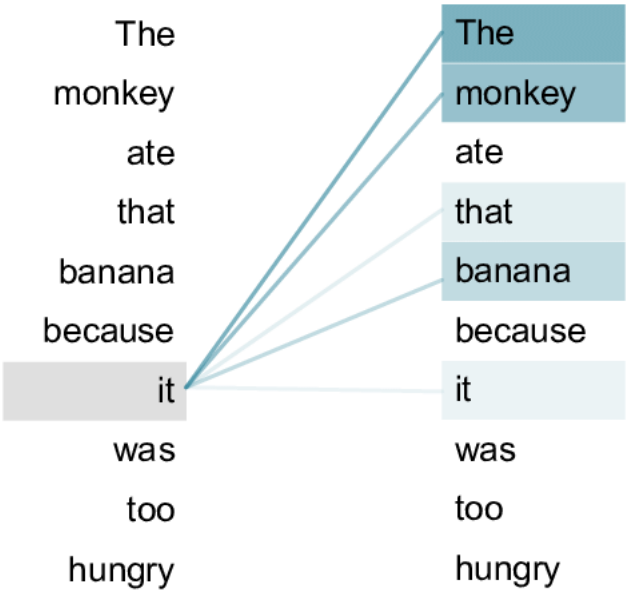

Figura 5: Ejemplo de obtención de la atención

Como se puede ver en la figura 5, se relaciona it con el mono y con la banana, debido a que puede hacer referencia a ambas palabras, resaltando con un color fuerte el caso que es más probable.

Modelo de lenguaje

Los modelos de lenguaje [7], son modelos de aprendizaje automático que tratan de predecir cual será la siguiente palabra dentro de una oración, teniendo en cuenta el contexto de las palabras anteriores. Un ejemplo de modelo de lenguaje, podría ser el autocompletado de los correos de Gmail:

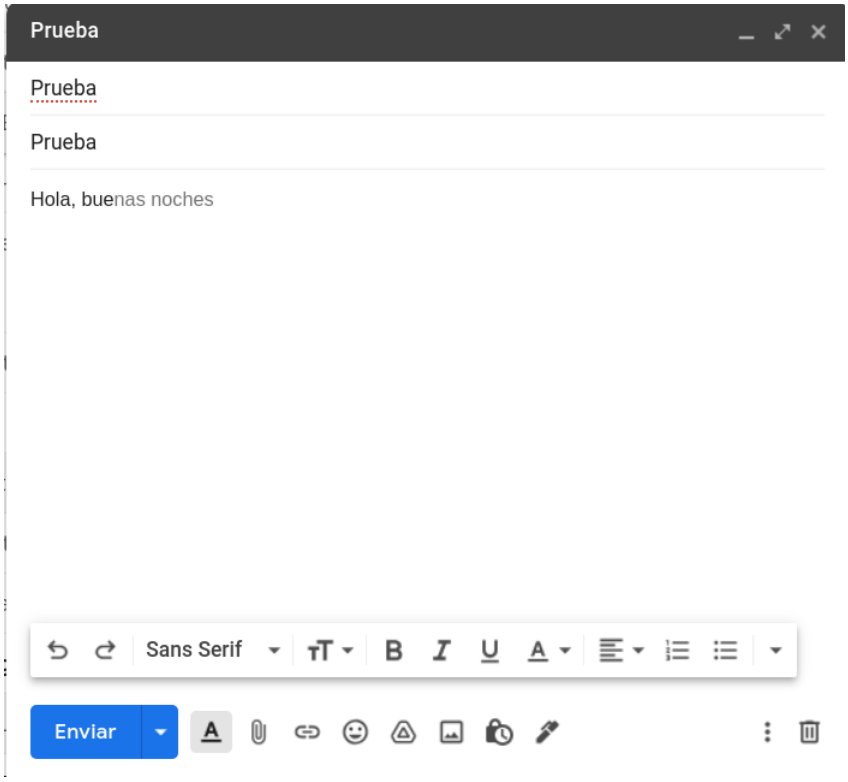

Figura 6: Ejemplo de un modelo de lenguaje

Estos modelos de lenguaje, se pueden utilizar para varias tareas, pudiendo ser éstas: traducción, predicción de palabras, respuesta a preguntas, resumen de textos... Para cada una de ellas, se puede realizar un entrenamiento personalizado a un modelo de lenguaje base o no (caso, en el que se podría adaptar la estructura/arquitectura del modelo de lenguaje). El entrenamiento personalizado denominado fine-tuning [15], trata de utilizar una arquitectura base de un modelo de lenguaje ya existente, a la cual se le aplica un entrenamiento con un conjunto de datos relacionados con la tarea definitiva que se quiere realizar, para que el modelo de lenguaje "fine-tuned", esté acostumbrado a trabajar con ese tipo de ejemplos y datos, siendo a su vez capaz de resolverlos de una mejor forma que la arquitectura/modelo de lenguaje inicial. Finalmente, los modelos de lenguaje que se van a utilizar en el proyecto, van a estar fundamentados en BERT o en GPT-3 (ambos basados en transformers). Las características principales de cada uno de ellos son:

- **BERT (Bidirectional Encoder Representations from Transformers).** Es una técnica basada en redes neuronales para el pre-entrenamiento del procesamiento del lenguaje natural desarrollada por Google. Este modelo, se creó utilizando dos corpus (conjunto amplio y estructurado de ejemplos reales de uso de la lengua) de lengua inglesa, siendo estos BookCorpus y Wikipedia. Al estar desarrollado por Google, su funcionalidad es la de interpretar consultas a la hora de realizar una búsqueda, utilizando la bidireccionalidad, que consiste en analizar una misma frase en ambas direcciones, llegando a entender de esta forma el contexto de la frase de una mejor forma. Este modelo al ser público, se utiliza en muchos otros modelos de lenguaje, por lo que sirve para realizar muchas funciones basadas en el modelo. [1] [13]

- **GPT-3.** En cuanto a la arquitectura de GPT-3, no se conoce mucho debido a que no es pública (la arquitectura de BERT la publicó Google). Lo que sí se sabe, es que es un modelo de lenguaje autorregresivo (depende de sus observaciones pasadas) y que está basado en la arquitectura de Transformers, habiendo invertido Microsoft más de 1.000 millones en OpenAI para su desarrollo. La funcionalidad principal de este modelo, es producir textos que simulan la redacción humana, por lo que se le tendrán que dar ejemplos previos para que de esta forma, el modelo trate de entender la tarea que tiene que realizar, denominándose el conjunto de ejemplos previos e instrucciones "prompt". [4] [10] Otro aspecto a destacar, es que se le podrán indicar varios parámetros, para ajustar la generación de texto a nuestras necesidades. Algunos de los parámetros más importantes son [6]:

  - **Temperatura.** Este parámetro, varía desde 0.00 hasta 1.00, indicando la aleatoriedad de las respuestas, por lo que se podrían obtener respuestas que tengan menos sentido en el lenguaje humano.
  - **Stop sequences.** En este campo, se introducen cadenas de texto que en el momento que se generen, se parará la ejecución y se obtendrá la respuesta.
  - **Longitud de la respuesta.** La longitud máxima de la respuesta, se indicará en tokens. El número total de tokens que se podrán pedir como respuesta, dependerá de la suma de los tokens de entrada y de este parámetro, por lo que si entre ambos, se da un número superior al que permite el engine, no se podrá realizar la ejecución.

  Finalmente, para realizar las llamadas se podrá utilizar la propia API que ofrece OpenAI o la página web.

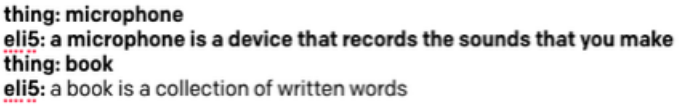

Figura 7: Ejemplo de GPT-3 describiendo objetos siguiendo la estructura dada en negrita

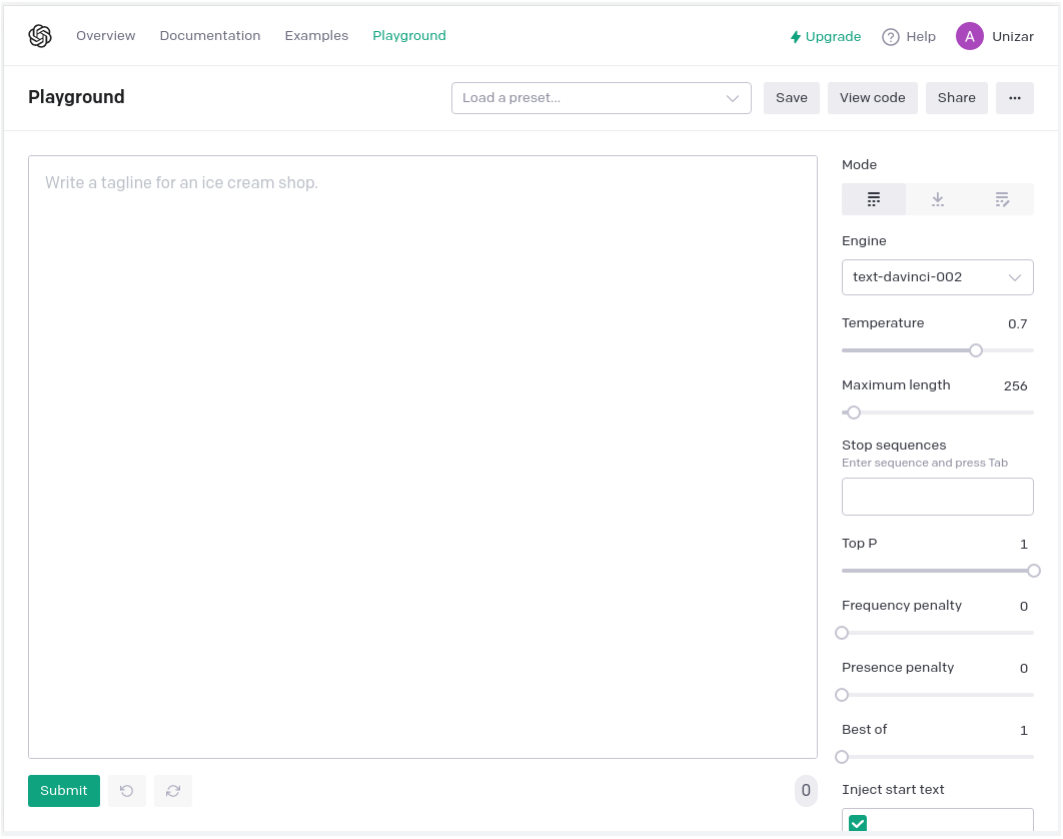

Figura 8: Playground de GPT-3 (página web para realizar pruebas y ejecuciones)

Clúster y learning-rate

El proceso de clusterización, trata de dividir varios datos en un número de grupos, en los cuales los datos que pertenecen al grupo, son similares y tienen una relación entre ellos, es decir, trata de agrupar datos en base a relaciones comunes entre los mismos. A cada grupo de datos, se le llamará clúster y al punto que tiene la misma distancia con todos los puntos se le denominará centroide.

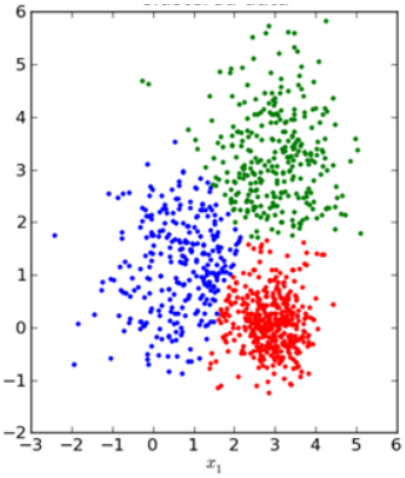

Figura 9: Ejemplo de clusterización

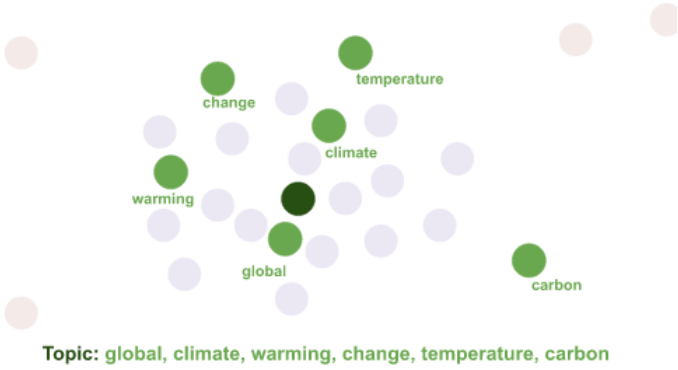

Figura 10: Ejemplo de centroide (punto verde más oscuro) [3]

Finalmente, la mayoría de estos algoritmos realizarán varias iteraciones para determinar la posición en la que se colocará cada clúster, por lo que sería similar al número de iteraciones que realizará el algoritmo de clusterización para designar los datos a un clúster.

Reducción de dimensiones

La reducción de dimensiones, es un proceso necesario en este proyecto, para poder visualizar la técnica de clusterización idónea. Esta reducción de dimensiones lo que hará es que de los embeddings generados por el modelo de lenguaje escogido, reducirá el espacio de dimensiones a 2 para poder visualizarlas como en la foto anterior (Fig. 9), debido a que imaginar una visualización de más de 3 dimensiones, es bastante complejo.

2. Estado de la cuestión

A raíz de surgir la idea del proyecto en un departamento orientado a la investigación, se decidió buscar en repositorios de documentos científicos de relevancia como Elsevier, Acm, IEEE o Google Scholar, para comprobar si ya existía algún trabajo científico publicado sobre la temática. Los resultados de la búsqueda fueron los siguientes:

Nº Doc	**Repositorio**	**Tecnologías**	**Dataset**	**Año**
1	Arxiv	Transformer + HDBSCAN + UMAP	20 news group y Yahoo respuestas	2020
2	Arxiv	MUSE + HDBSCAN + UMAP	Tweets de elecciones turcas	2022
3	IEEExplore	word2vec + varios tipos de clusterización	Papers de Web of Science	2019
4	Springer	doc2vec + DEC	Reportes de patentes KIPRIS1 y KISTA	2018

Cuadro 1: Documentos seleccionados

De estos documentos, se puede apreciar que el primero de ellos, con título "Top2Vec: Distributed Representations of Topics" sería el más parecido a la idea de proyecto definida, debido a que se sigue un procedimiento bastante similar, al mencionar la posibilidad de utilizar un modelo de lenguaje con la arquitectura de BERT. Aunque se puede apreciar que en este documento, se realiza primero la reducción de dimensiones y posteriormente la clusterización, contando con una menor cantidad de datos para este proceso. Finalmente, se obtiene el centroide y en base al mismo, se utilizan los "word vectors" (obtenidos con word2vec) de los puntos cercanos para convertirse en los topic del clúster. Finalmente, las fuentes de datos que se utilizan en el paper, serán un dataset de noticias y otro de Yahoo respuestas, aunque el código está disponible en Github para el uso del proyecto.

El segundo documento, titulado "Patent document clustering with deep embeddings", utiliza la extracción de Tweets como fuente de información, generando los embeddings también con una red neuronal, pero en este caso, no utilizará la arquitectura de Transformer, sino la red neuronal MUSE, tras aplicarle un pre-entrenamiento para representar tweets. En este documento, también destaca el uso de la reducción de dimensiones previa a la clusterización, utilizando UMAP para la reducción y HDBSCAN para la clusterización (mismos métodos que en el documento anterior). Finalmente, debido a que la funcionalidad del documento era encontrar la afinidad con una campaña política de Turquía, la representación de los clústeres, sirve para saber si están a favor o en contra de los distintos partidos.

Finalmente, en los dos últimos documentos titulados "Two-Stage Topic Extraction Model for Bibliometric Data Analysis Based on Word Embeddings and clustering" (2019) y "Embeddings-Based clustering for Target Specific Stances: The Case of a Polarized Turke" (2018) al ser más antiguos, no se utilizan técnicas tan modernas como en los anteriores, debido a que las arquitecturas mencionadas todavía no estaban tan popularizadas, por lo que para la generación de los embeddings, utilizan word2vec y doc2vec [8], los cuales crean un vector de dimensiones por cada palabra o documento respectivamente. En el apartado de la clusterización, en el tercer documento, utilizan varias técnicas para de esta forma comprobar cual es la que mejor se adapta a los embeddings obtenidos y en el último documento, realizan el mismo procedimiento con varios métodos, evaluando cual de ellos obtiene una mejor precisión, siendo éste DEC (técnica que utiliza "deep encoders").

Vistas las técnicas y los datos utilizados en los documentos anteriores, con este proyecto, se busca aportar otro enfoque, debido a que se va a analizar la similaridad de los clústeres, en lugar de obtenerlos únicamente, por lo que es posible que sea necesario emplear otro tipo de técnicas que se ajuste más a la finalidad del proyecto. Otro aspecto en el que se diferencia este proyecto, es que la clusterización se realiza antes que la reducción de dimensiones, consiguiendo de esta forma una menor pérdida de datos (no son lo mismo embeddings con 2 dimensiones que con 512, 768...)

3. Análisis, Diseño e Implementación

La metodología que se decidió aplicar a lo largo del desarrollo del proyecto, fue de tipo "agile" denominada SCRUM. Entre las características de esta metodología, destaca la adopción de los sprints, siendo estos un período breve de tiempo, en el que se tiene que completar una cantidad de trabajo concreta. En el caso de este proyecto, al ser de investigación y no tener una fecha determinada, no se llegaron a aplicar ya que no era necesario tener objetivos con plazos específicos, por lo que el proyecto se realizó con una variación que implementaba los scrums diaros, siendo estos reuniones de 15 minutos, cuyo objetivo era que todos los miembros del equipo se mantuviesen actualizados sobre los avances del resto de miembros del grupo, comentando posibles problemas que les surgían, cabiendo la posibilidad de que otros integrantes del grupo pudiesen ayudar a solventarlos, y finalmente, comentando las previsiones a realizar durante el día.

3.1. Análisis

El primer paso que se realizó, fue el de decidir el tipo de documentos que se iban a utilizar y a su vez, los temas sobre los que se van a extraer los documentos y el lugar de extracción de los mismos (`https://www.semanticscholar.org/`). El tipo de fuente de datos escogida, fueron documentos de investigación científica (papers), siendo su lugar de extracción la página web comentada, debido a que cuenta con una api (Semantic Scholar Academic Graph API) obtenida gracias al departamento, con la que la extracción de los documentos se puede realizar de forma muy sencilla.

Como se puede apreciar en la tabla 2, se ha escogido un tema que tiene menos relación entre sí que el resto (geografía), para así poder ver la relación entre los distintos campos de una forma más clara al tener algunos mucha relación entre ellos, otros menos y finalmente la geografía muy poca.

La elección de documentos científicos, se debe al tamaño de los documentos, ya que únicamente se utilizará el abstract de los mismos, al tener los modelos de lenguaje una limitación de tokens con los que se puede realizar la consulta (este número será dependiente del modelo).

Tema	Descripción	Aplicaciones/Ejemplos
Quantum Computing	Paradigma de computación distinto al de la informática clásica, al basarse en el uso de cubits: una combinación de unos y ceros distinta a la de la informática actual (1 o 0), en la que pueden estar en los dos estados simultáneamente, dando lugar a nuevas puertas lógicas y algoritmos.	Optimización, mejoría en cálculos de Machine Learning o cálculo más rápido de posibles escenarios
Digital Twin, Smart devices	Réplica digital de un producto, servicio o proceso. La idea es someter dicho producto o servicio a estrés, de manera que se vean testadas sus principales debilidades sin la necesidad de construir prototipos costosos	Comprobar eficiencia en una fábrica, analizar comportamientos frente a catástrofes meteorológicas...
AI assisted code generation	Es un modelo de lenguaje que se ha entrenado con código de varios lenguajes de programación, cuya función es generarlo de forma automatizada basado en un input	Codex, GitHub Copilot...
Cloud Computing, Edge Computing	Tecnología que permite acceso remoto a servicios por medio de Internet, siendo una alternativa a la ejecución en local. De esta forma, los recursos del ordenador no serán necesarios, ya que se utilizarán los recursos del lugar al que se accede mediante Internet	AWS, Azure, Google
Cybersecurity, AI Cybersecurity	Proceso por el que se tratan de defender los ordenadores, los servidores, los dispositivos móviles, los sistemas electrónicos, las redes y los datos de ataques maliciosos. El caso de la IA, se podría utilizar para reconocer patrones de ataque o ayudar a crear planes de defensa	VPN, patrones de comportamiento del tráfico de la red, identificación de usuarios...
Geography	Estudia la descripción o la representación gráfica de la Tierra. Es decir, es la ciencia que estudia la superficie terrestre, las sociedades que la habitan y los territorios, paisajes, lugares o regiones que la forman al relacionarse entre sí.	Mapas, análisis espacial, climatología...

Cuadro 2: Temas escogidos para analizar la similaridad semántica

Para los documentos que se excedan del número máximo de tokens, se podrán realizar varios métodos para generar los embeddings de forma completa:

- **Truncation.** Este método es el que se realiza de forma automática por la mayoría de modelos de lenguaje y en este caso, cuando se llega al número máximo de tokens, se truncan (los siguientes no se contemplan), generando de esta forma los embeddings de los tokens con la capacidad permitida.
- **Longformer.** [13] Este otro método, trata de utilizar otro modelo de lenguaje distinto con una mayor capacidad de tokens (longformer) para formar los embeddings. Unos ejemplos de longformers serían:
  - `https://huggingface.co/allenai/longformer-base-4096`
  - `https://huggingface.co/markussagen/xlm-roberta-longformer-base-4096`
- **Summarization.** El método de summarization, trata de utilizar un longformer (método anterior), pero con un entrenamiento, que le da la capacidad de realizar resúmenes de la entrada de texto, por lo que con este método, se podrían realizar resúmenes progresivos hasta conseguir el número de tokens máximo del modelo de lenguaje a utilizar para así generar los embeddings. Algunos ejemplos de estos modelos son:
  - `https://huggingface.co/sshleifer/distilbart-cnn-12-6`
  - `https://huggingface.co/facebook/bart-large-mnli`
  - `https://huggingface.co/google/pegasus-xsum`
  - `https://huggingface.co/csebuetnlp/mT5_multilingual_XLSum`
- **Chunking.** [5] Este método, separa el texto original en chunks (secciones más pequeñas del texto) con el número máximo de tokens permitidos por el modelo, realizándose el cálculo de los embeddings para cada chunk por separado. Para obtener los embeddings finales (de todo el texto), se podrán utilizar dos métodos, con la media de los embeddings o con el máximo de los embeddings.

  Para la media de los embeddings, se utilizará la fórmula siguiente:

$$embeddings\_final[0] = \frac{embeddings\_0[0] + [embeddings\_1[0] + ... + [embeddings\_n[0]}{nchunks}$$

Figura 11: Fórmula para la obtención de la media de embeddings

Este caso, se obtendría la media del embedding 0 de todos los chunk, por lo que habría que hacer este proceso para el número de embeddings que genere el modelo de lenguaje. El otro método, sería similar, pero habría que obtener el valor máximo para cada embedding de todos los chunks, es decir:

$$embeddings\_final[0] = max(embeddings\_0[0] + [embeddings\_1[0] + ... + [embeddings\_n[0])$$

Figura 12: Obtención del valor máximo de cada embedding

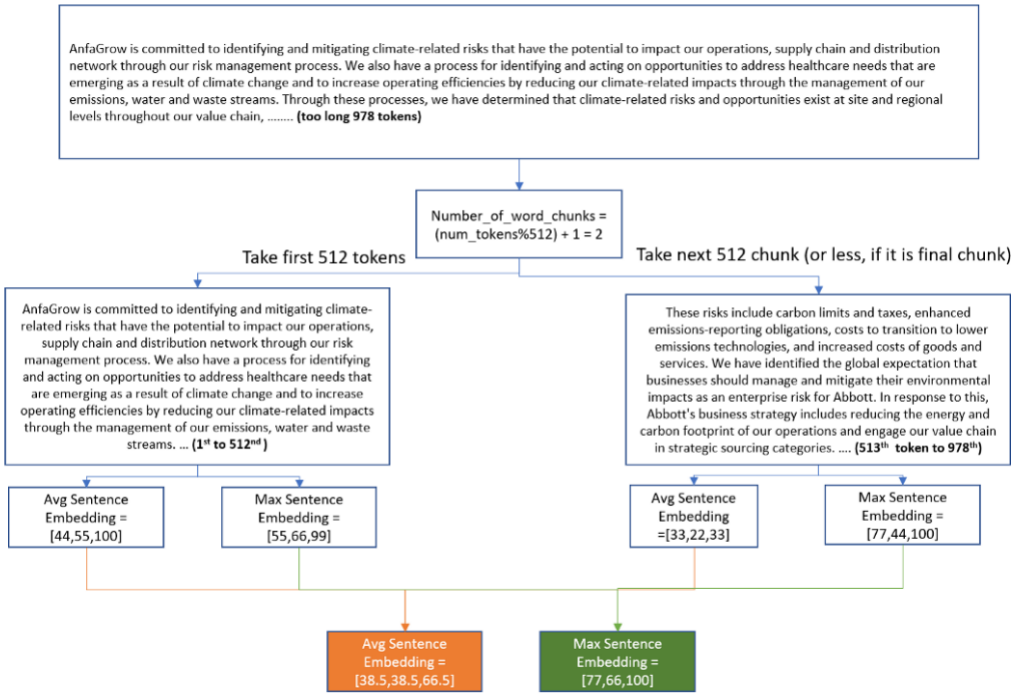

Figura 13: Ejemplo chunking

En la figura 13, se puede apreciar como el texto que excede los 512 tokens, se divide en otros dos textos más pequeños (en este caso de 512 y 465 tokens), para posteriormente obtener los embeddings medios y máximos de cada fragmento de texto, juntándolos finalmente en los embeddings del texto entero (cuadro naranja para los embeddings medios y verde para los máximos) tras aplicar las fórmulas anteriores.

Dentro de los modelos de lenguaje a elegir para realizar el proceso, se pueden diferenciar en dos tipos:

- **Generalistas.** Estos modelos de lenguaje, no están orientados a unos textos en específico, por lo que funcionan de forma parecida para todos los textos introducidos (GPT-3, Roberta...)
- **Especialistas.** Estos modelos, han recibido un entrenamiento o un fine-tuning específico para unos ejemplos concretos, por lo que son mejores trabajando con el tipo de textos que se han utilizado en el entrenamiento. (SciBert, Specter...)

Una vez identificados los tipos de modelos de lenguaje, destacan los siguientes[14]:

- **GPT's.** Estos modelos de lenguaje, pertenecen a OpenAI y tienen dos versiones, GPT-2 y GPT-3. Al ser GPT-3 una versión actualizada de la anterior, se decidió investigar la misma. Tras el proceso de investigación, GPT-3 es un modelo de lenguaje generalista el cual tiene un precio dependiendo de la versión, ya que dentro de GPT-3, existen 4 engines (versiones) con características distintas [10]. Los distintos engines del modelo de lenguaje son:

Nombre Modelo	$ por 1k tokens	Max tokens input	Embeddings output (dimensiones)
Ada	0.0008	2048	1024
Babbage	0.0012	2048	2048
Curie	0.0060	2048	4096
Davinci	0.0600	4000	12288

Cuadro 3: Comparación de los modelos de GPT-3

  En estos modelos, el resultado de las funcionalidades que pueden realizar va mejorando con respecto al precio, pero a su vez, también disminuye la velocidad con la que realizan las mismas. Otro aspecto a destacar con la mejoría de cada modelo, es que necesitan menos contexto para realizar la acción de forma correcta. Finalmente, los modelos anteriores también tienen precios para realizar un fine-tuning de los mismos.

- **Roberta Large.** Este modelo, fue propuesto en el siguiente paper ([17]), aunque para la tarea a realizar, se ha encontrado un modelo "fine-tuned" en Hugging Face, que está especializado en la clusterización. Este modelo, acepta un máximo de 128 tokens y la salida es de 1024 dimensiones. [18]

- **SciBert.** Este modelo, es una variación de BERT, pero fine-tuned con documentos científicos de Semantic Scholar. La salida es de 768 dimensiones y el input de token máximos son de 512.

- **Specter.** Este modelo de lenguaje, está basado en el anterior (SciBert), pero con otra capa de entrenamiento, por lo que está diseñado para trabajar con documentos científicos y con su estructura típica. El diseño, se realizó en el siguiente paper: [11] y el modelo, de Hugging Face [19], cuenta con una entrada de tokens máximos de 512 y una salida de 768 dimensiones .

A la hora de realizar la clusterización, se buscó información sobre varios métodos que podían realizarla de forma correcta en base a los embeddings generados. Los mejores métodos de clusterización para este proceso son:

- **Kmeans.** El algoritmo k-means, agrupa los objetos basándose en las características. Este agrupamiento, se realiza mediante la distancia cuadrática, realizando la asignación de objetos a los centroides y su actualización hasta que no se desplacen (o se muevan una distancia muy pequeña).

- **Gaussian Mixture Model.** Este algoritmo, es una generalización de K-means con la que en lugar de asignar cada observación a un único clúster, obtienen una distribución probabilística con la pertenencia que podría tener hacia cada clúster, por lo que para seleccionar el definitivo, habrá que calcular a cual podría llegar a pertenecer.

- **HDBSCAN.** HDBSCAN, es otro algoritmo de clusterización que extiende de DBSCAN, desarrollado en el siguiente paper [20].

- **BERTopic.** Es una técnica de "topic modeling", que utiliza la estructura de los transformers (modelos de lenguaje anteriores) para realizar la clusterización y c-TF-IDF para obtener las keywords de los distintos temas, pudiendo utilizar specter como modelo de lenguaje para detectar los clústeres.

Para obtener las keywords, primeramente habrá que realizar la clusterización con el método escogido. Posteriormente, con todo el texto en crudo del clúster (no embeddings), se podrán obtener las keywords para cada tema, debido a que ya se habrán agrupado los documentos. Para obtener las keywords en base a este texto generado se podrán aplicar estas técnicas:

- **KeyBERT.** Es una técnica de extracción de keywords y keyphrases, que utiliza embeddings generados por modelos de lenguaje BERT para extraer las keywords, por lo que tendrá las restricciones de tokens comentadas anteriormente.
- **TF-IDF.** Es una técnica de extracción de keywords, que se basa en buscar las palabras más repetidas en un texto, para así determinar su relevancia. Finalmente, se obtendrán esas palabras como keywords
- **BERTopic.** Como se ha comentado anteriormente, BERTopic utiliza c-TF-IDF (una variación de TF-IDF) para obtener las keywords, realizándose este proceso de forma automática al obtener los clústeres.

Para la obtención del tema del que habla un clúster, se utilizarán las keywords y GPT-3, debido a la gran capacidad de abstracción de tareas a realizar en base a unos ejemplos dados, por lo que de esta forma, se podrá generar una denominación de los clústeres a la hora de visualizarlos.

Finalmente, para representar las relaciones entre los distintos temas encontrados (clústeres), se va a utilizar la matriz de correlación, ya que en ella, se puede ver el grado de similaridad entre los clústeres. Un ejemplo de matriz de correlación sería el siguiente:

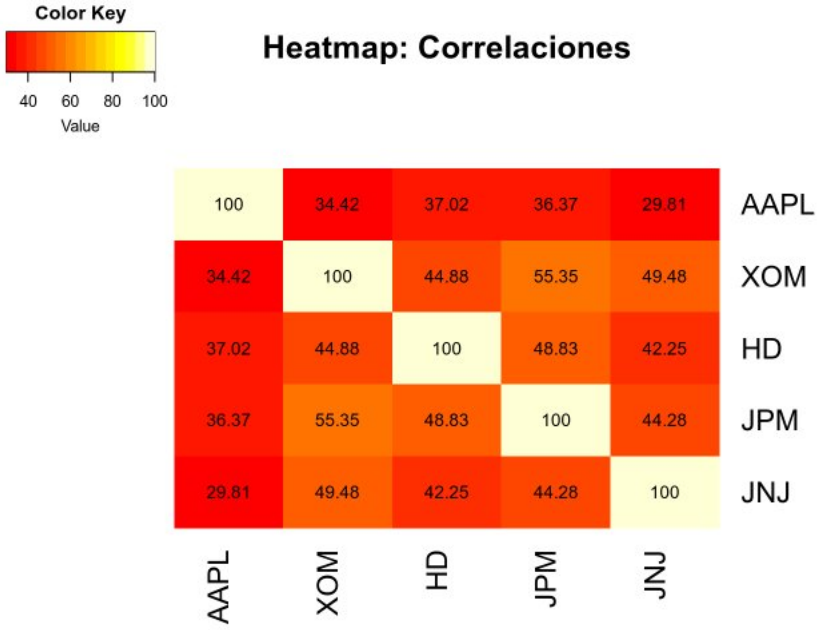

Figura 14: Ejemplo matriz de correlación

Una matriz de correlación, es una tabla en la que se muestra una lista multivariable de forma horizontal y la misma, de forma vertical. Para obtener los valores, se multiplica el valor de cada columna vertical con su respectiva horizontal, dando lugar de esta forma la similaridad asociada a ambos campos. En el caso de los embeddings al ser varias dimensiones, primero habrá que hacer el producto escalar de las mismas, para así de esta forma únicamente operar con un número concreto. Una vez obtenida la matriz, se podrán aplicar funciones de normalización (número más alto 1) o de realce (para visualizar las diferencias de forma más clara).

3.2. Diseño

Tras el proceso de búsqueda de todas las posibilidades, al ser un proceso de investigación, en algunas fases se tuvieron que realizar pruebas para comprobar que método funcionaba mejor y en otras, únicamente con ver las características de la propuesta, se pudo desestimar. Para cada una de las fases, se realizó lo siguiente:

- **Fuente de datos.** Mejor opción Semantic Scholar (API gratuita).
- **Modelo de lenguaje.** Al no conocer exactamente como se comportaría cada modelo de lenguaje a la hora de generar los embeddings, en la fase de diseño no se seleccionó ningún modelo de lenguaje, teniendo que realizar en la fase de implementación varias pruebas con los distintos modelos, con un dataset más pequeño, en el que se utilizaban varios servicios con un breve resumen de cada uno, por lo que se generaban los embeddings y se proyectaban en la matriz de correlación para ver con qué modelo se realizaban mejor (no es necesario realizar clusterización al ser un resumen concreto para cada tecnología).

  Por otro lado, el método escogido para tratar los abstract más largos de los tokens, ha sido el de truncation (por defecto en los propios modelos de lenguaje) debido a que los otros modelos tienen los siguientes problemas:

  - **Longformer.** Al tener que utilizar un longformer, hay pocos que tengan un entrenamiento con documentos científicos, por lo que será más difícil que obtengan buenos resultados
  - **Summarization.** Como se va a utilizar otro modelo de lenguaje que resuma, los resúmenes puede ser que omitan palabras claves o frases que sean fundamentales en el abstract, por lo que podría llegar a tener otro significado.
  - **Chunking.** Al dividir en chunks, uno puede tener menos relación que otro con los aspectos que trata el documento por lo que al hacer la media o el máximo, es posible que los embeddings obtenidos tengan poca similaridad con el documento tratado.

  Finalmente, al tener la mayoría de modelos de lenguaje 512 tokens, pueden ser una cantidad suficiente para entender el contexto del documento a la hora de generar los embeddings.
- **Clusterización.** Para este caso, se probaron los distintos métodos y se visualizaron (utilizando un método de reducción de dimensiones), para ver cuál la realizaba de la forma más equilibrada, comprobando a su vez si no había clústeres con muchos documentos.
- **Keywords.** A la hora de obtener las keywords, se realizaron pruebas con las distintas técnicas
- **Topic generation.** Para obtener el tema de los clústeres, como se ha comentado anteriormente se decidió realizar con GPT-3 pasándole una prompt con ejemplos, al ser uno de los modelos de lenguaje más avanzados a la hora de realizar tareas de este estilo y ofrecer una prueba gratuita como se ha comentado.
- **Representación.** Matriz de correlación como se ha comentado anteriormente.

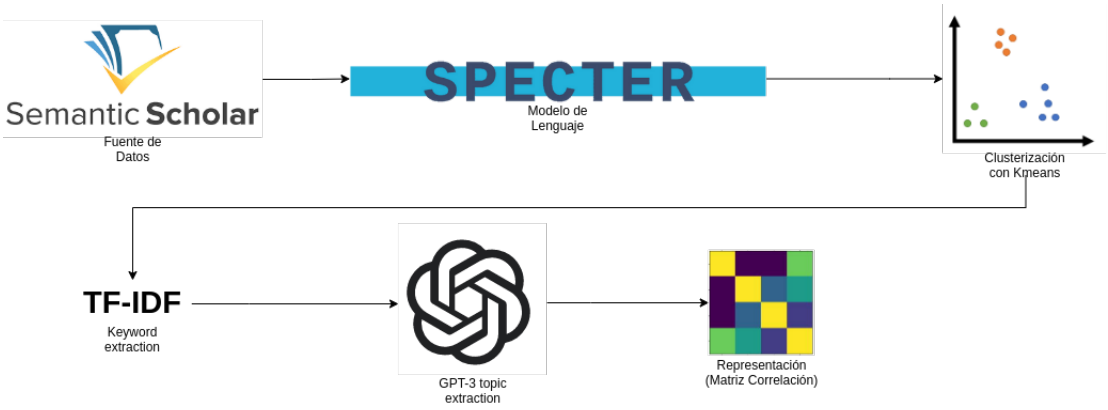

Figura 15: Diseño de las distintas fases que conforman el proyecto

3.3. Implementación

A lo largo de todo el proceso de implementación, se utilizaron las siguientes herramientas y tecnologías:

- **Python.** Lenguaje de programación con mayor cantidad de librerías y documentación a la hora de trabajar con aspectos relativos a la Inteligencia Artificial.
- **Conda.** Herramienta que permite crear distintos entornos en Python para poder trabajar con librerías específicas instaladas en cada uno, por lo que es más fácil trabajar con las librerías y en caso de error, se puede crear uno nuevo.
- **Jupyter Notebook.** Programa que sirve para ejecutar código Python, dividido en celdas independientes con kernel compartido entre las celdas. De esta forma, las variables se guardan en memoria, y procesos largos pueden ejecutarse una única vez, almacenándose su resultado para poder trabajar posteriormente con él y manipularlo.
- **KNIME.** Herramienta orientada a la minería de datos, que utiliza módulos con procesos habituales definidos, agilizando de esta forma la creación de pipelines y ejecución de procesos, es decir, una herramienta "low-code". Finalmente, esta herramienta trabaja con tablas (DataFrames de la librería de Python Pandas) en las salidas.

Extracción de los documentos de Semantic Scholar

Para este proceso, se utilizó la herramienta de KNIME, debido a que en el departamento se contaba con un componente, que extrae los documentos directamente de Semantic Scholar con palabras clave pasadas como entrada. El resultado del Workflow para uno de los temas concretos es el siguiente:

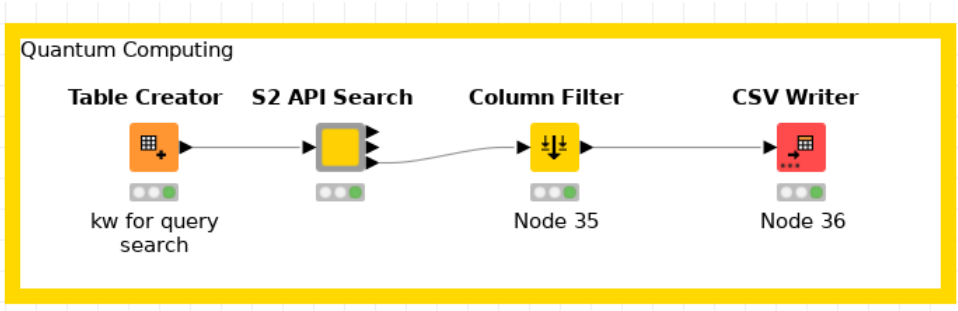

Figura 16: Proceso de extracción de documentos de Quantum

Lo que hace cada componente es:

- **Table creator.** Se introduce una keyword relacionada con Quantum Computing por cada fila de la tabla (las keywords dependerán del tema).
- **S2 API Search.** Con la entrada de la tabla anterior y configurando el componente con el máximo de documentos a buscar y la API Key, se buscan los documentos en Semantic Scholar.
- **Column Filter.** Se filtran las columnas de título y abstract al ser las necesarias para la generación de los embeddings
- **CSV Writer.** Se escribe el resultado de la tabla anterior (filtrada) en un csv, en el que se pondrá la opción de append para tener todos los documentos en un mismo csv al ir acumulándose (un proceso en KNIME para cada tema).

Pruebas con los modelos de lenguaje y selección del mismo

A la hora de elegir un modelo de lenguaje, se realizaron pruebas con un conjunto reducido de definiciones de servicios ya clasificados, por lo que únicamente había que realizar la matriz de correlación al generar los embeddings para comprobar que modelo realizaba la matriz con más sentido. Para esto, se creo un Jupyter Notebook realizando las pruebas con GPT-3 y sus distintos engines (obtención de 20$ prueba con una cuenta para realizarlo), con Roberta Large y con Specter. SciBert se descartó, debido a que en el paper de presentación de Specter se comprobaba que tenía un mejor funcionamiento en este tipo de documentos, por lo que directamente no se realizaron las pruebas propuestas [11]. Para la realización de estas pruebas, se siguieron estos pasos para llevar a cabo toda la implementación:

- Elección de los servicios y de su definición. Los servicios elegidos fueron:
  - Informática empresarial
  - Inteligencia de Negocio
  - Prestación de servicios de aplicación
  - Innovación
  - Computación en la nube
  - Automatización
  - Entorno Laboral Digital
  - Ciberseguridad
  - Outsourcing
  - Experiencia del cliente
  - Nuevas Tecnologías
- Generación de un fichero csv con los embeddings de cada definición de servicios para cada modelo de lenguaje, por lo que había que realizar la consulta con la API o la obtención del modelo de lenguaje para generar los embeddings. Para los GPT's, se implementó una llamada a la API que devuelve los embeddings. Por otro lado, para Roberta y Specter, se utilizó la librería de sentence_transformers, la cual importa el modelo desde Hugging Face y ya permite obtener los embeddings
- El último paso, fue el de comparar las siguientes matrices de correlación:

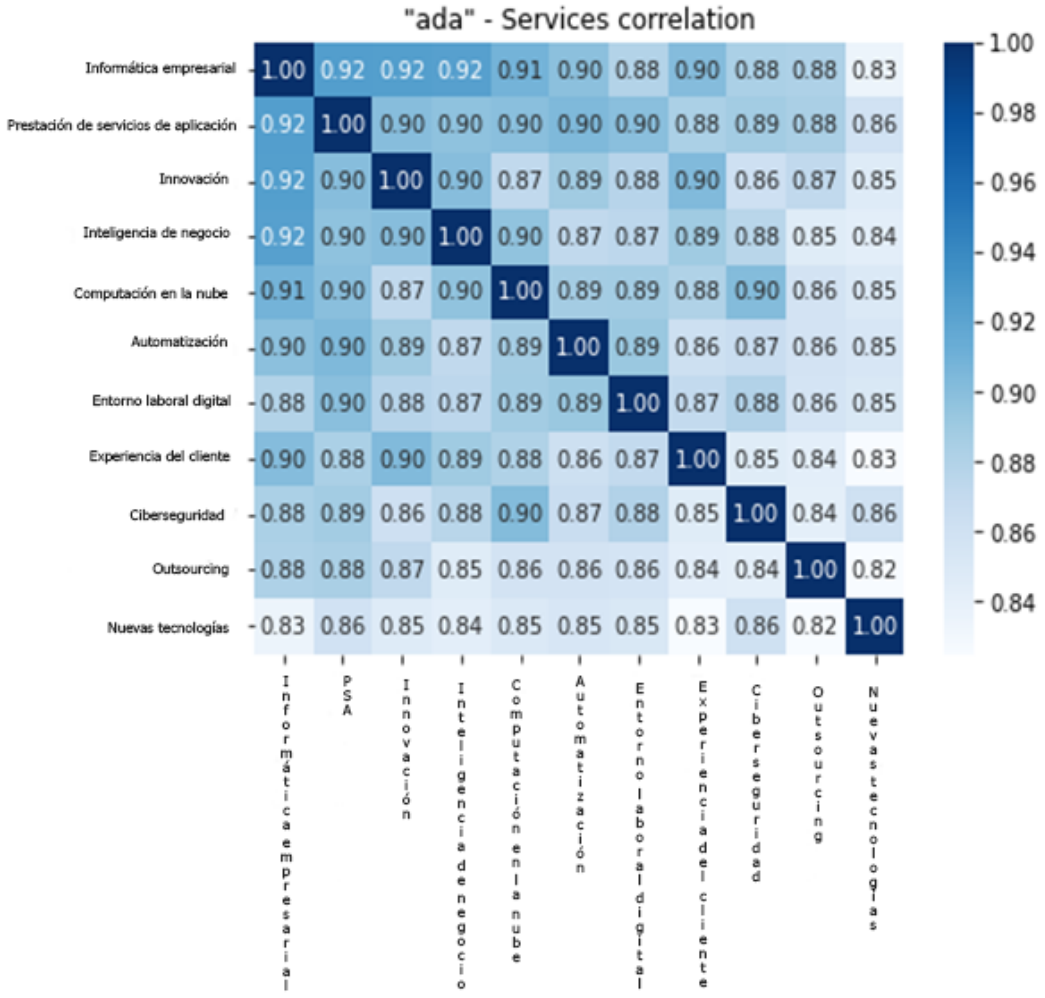

Figura 17: Matriz de correlación definida por Ada

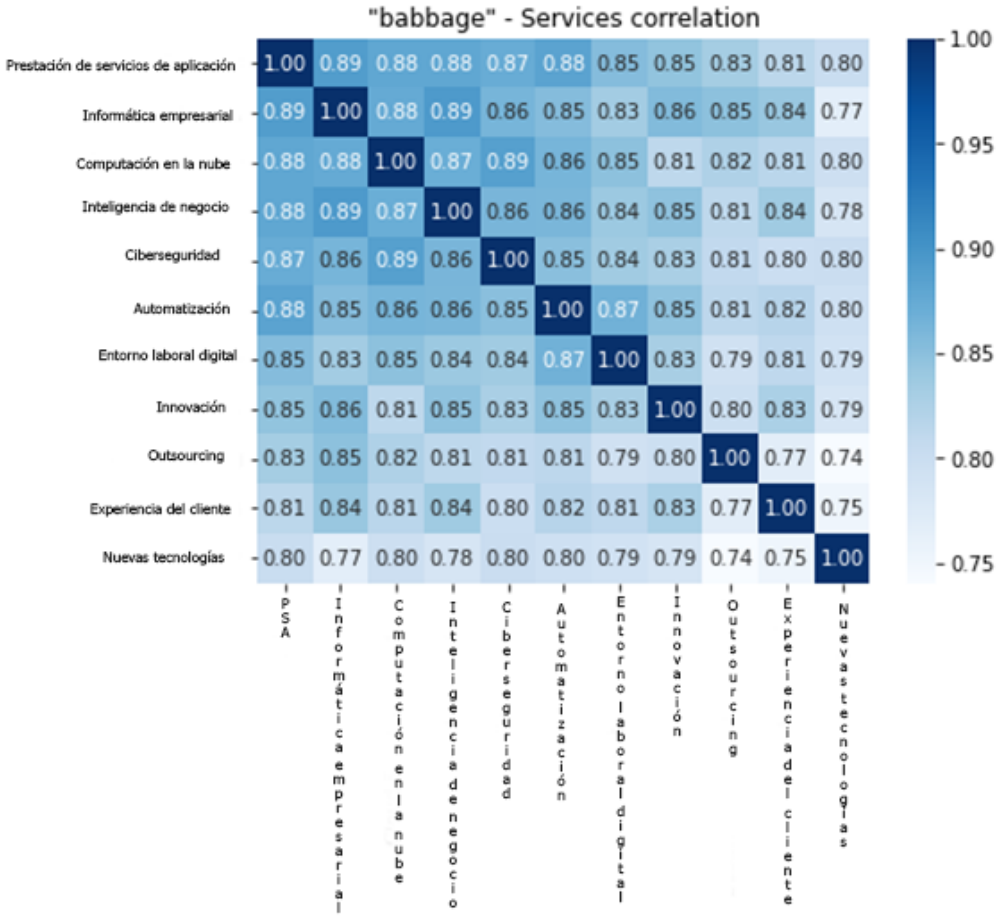

Figura 18: Matriz de correlación definida por Babbage

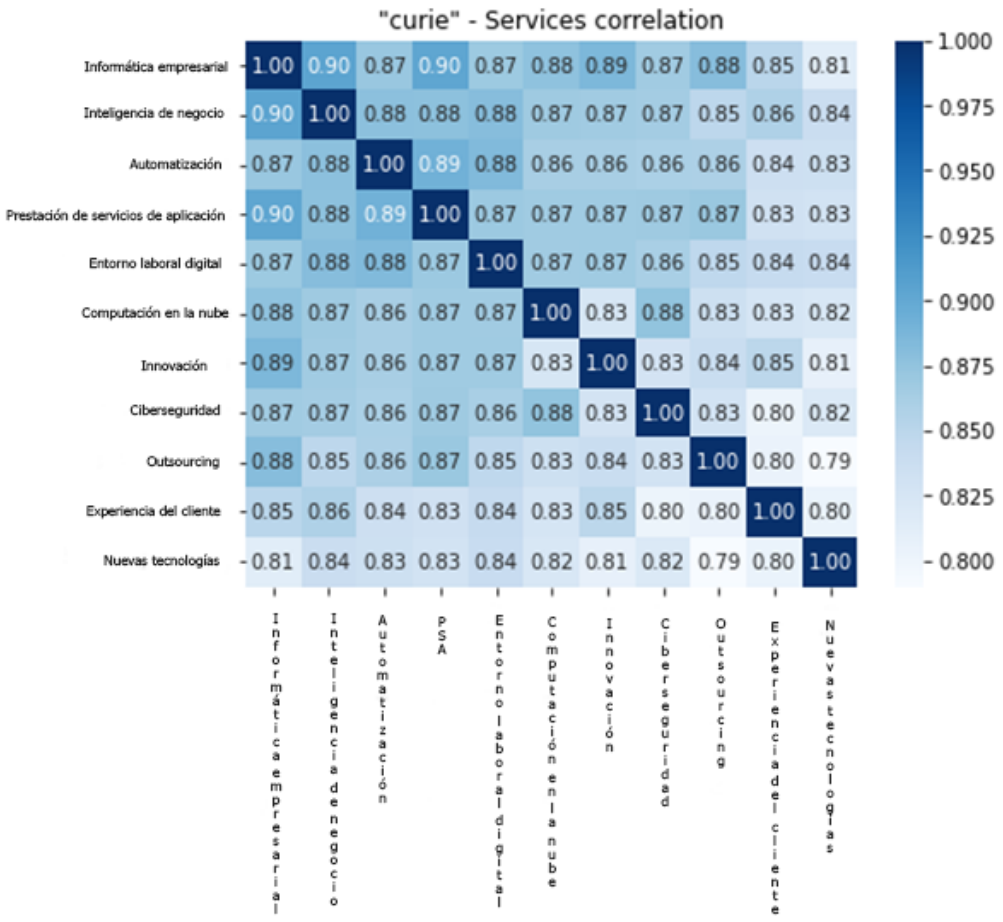

Figura 19: Matriz de correlación definida por Curie

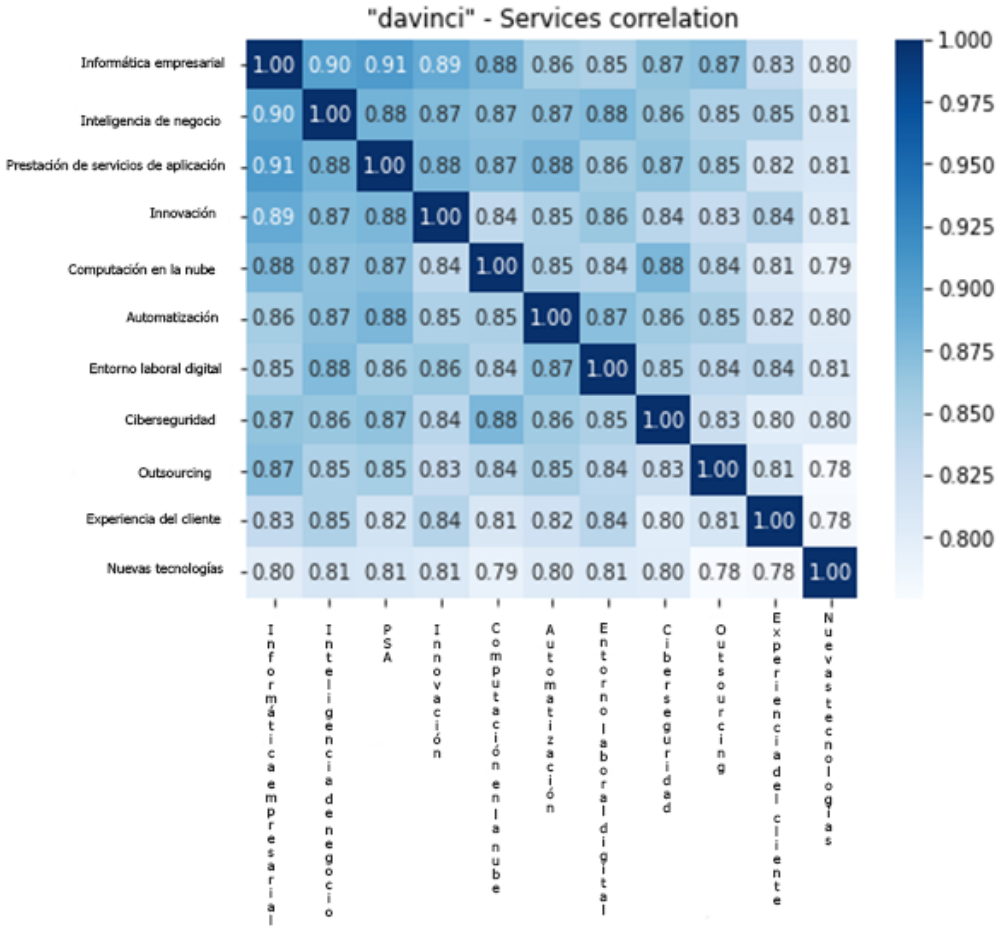

Figura 20: Matriz de correlación definida por Davinci

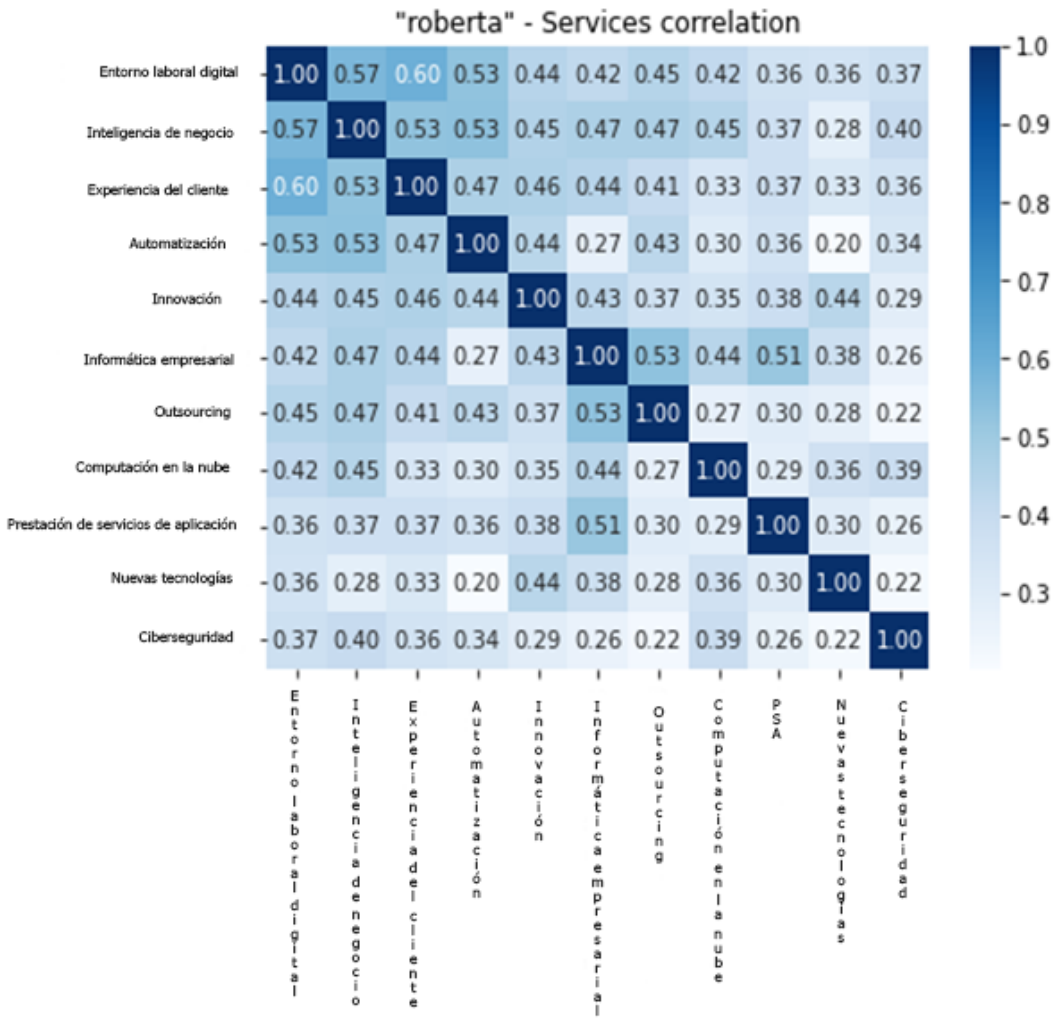

Figura 21: Matriz de correlación definida por Roberta

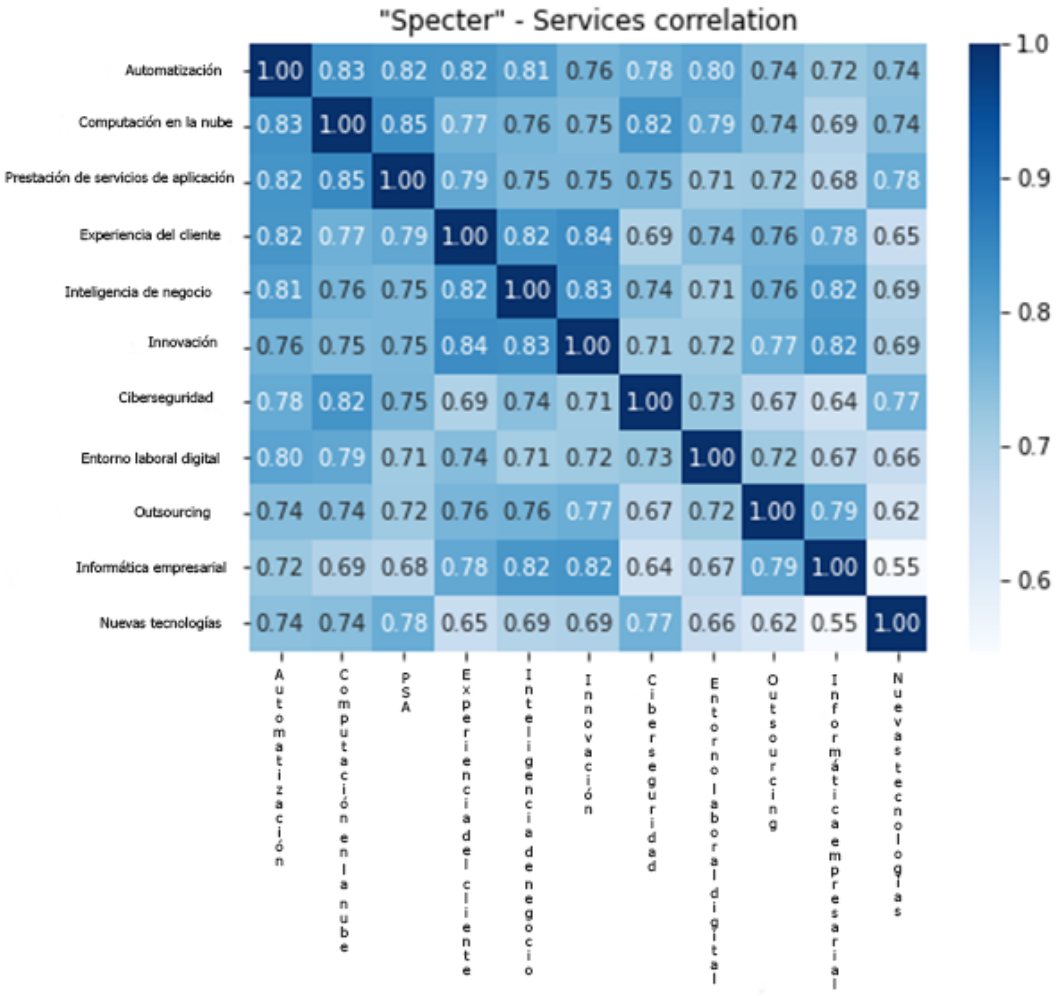

Figura 22: Matriz de correlación definida por Specter

Una vez vistas las matrices, se puede apreciar que para los GPT's con cada modelo van mejorando al tener más dimensiones y unas mejores características. Este cambio, se puede apreciar debido a que las transiciones cada vez son más suaves. Posteriormente, con Roberta se nota que no detecta muy bien las diferencias al tener un campo de colores bastante parecido en toda la matriz. Finalmente, se puede ver que Specter, detecta las relaciones en varias zonas de la matriz de correlación y se notan más claras las zonas en las que no hay una relación muy directa, por lo que sería la mejor opción, al estar acostumbrado a utilizar el lenguaje técnico, cosa que se ha visto reflejada en las matrices.

Para la generación de los embeddings con Specter de los documentos finales, se adaptará el Jupyter Notebook, ya implementado para las pruebas, en KNIME. Para ello, se utilizaron los componentes de Load Specter Model y Text Embeddings generados en el departamento, ya que realizaban la función en específico. La implementación en KNIME es la siguiente:

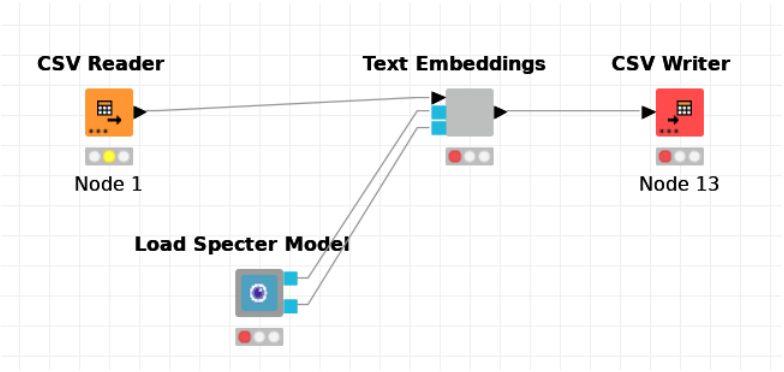

Figura 23: Visión general de la implementación

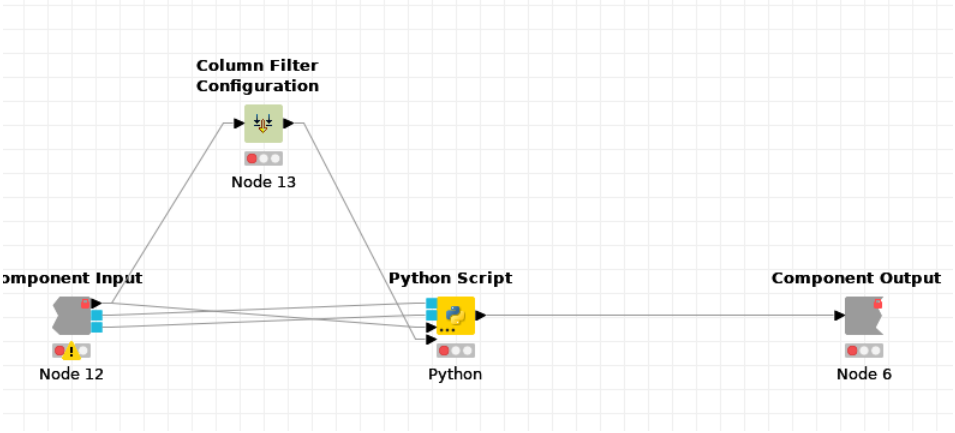

Figura 24: Visión específica del componente Text Embeddings

- **CSV reader (Fig. 23).** Se encargará de leer el fichero con los documentos obtenidos de Semantic Scholar
- **Load Specter Model (Fig. 23).** Este componente, se encarga de cargar el modelo de lenguaje de Specter (modelo escogido), para luego poder trabajar.
- **Text Embeddings (Figs. 23, 24).** Este componente, cuenta con un filtro en el que se escogerán las columnas, cuyo contenido será sobre el que se realicen embeddings y posteriormente, en el componente de Python Script se generarán correctamente
- **CSV Writer (Fig. 23).** Este componente, escribirá en un csv la salida obtenida, es decir, un csv con el título del documento y el abstract (ya obtenidos anteriormente) y los embeddings para cada documento.

Elección de la técnica de clusterización

Para la técnica de clusterización, se eligió K-means, debido a que se quería analizar la similaridad de unos documentos con otros. K-Means es el método que más se acerca, al estar basado en la clusterización mediante centroides (punto equidistante de todos los puntos del clúster). Otros motivos para la elección, fueron que a la hora de aplicar HDBSCAN se obtenía un clúster con mucho ruido, cosa que hacía que no estuviesen equilibrados, por lo que se prescindió de la técnica. BERTopic, se descartó directamente al depender de un modelo de lenguaje, por lo que tiene un número máximo de tokens para poder analizar los documentos, y las otras técnicas, no necesitan de ello. Otro motivo para no utilizar BERTopic, fue que realiza primero la reducción de dimensiones y posteriormente, la clusterización, por lo que de esta forma, se pierden datos al contar únicamente con dos dimensiones. Finalmente, se eligió K-means frente a Gaussian por lo comentado con respecto al centroide.

A la hora de poder visualizar los clústeres y de esta forma elegir un método u otro para realizar las pruebas, había que reducir las 768 dimensiones a 2, por lo que el método que se utilizó, fue TSNE (t-Distributed Stochastic Neighbor Embedding), que es un método de reducción de dimensiones basado en las distancias, basándose en la similaridad. Al elegir K-Means, se investigaron los clústeres óptimos, cuyos resultados se verán en el apartado de resultados (4). Para la implementación de la técnica de clusterización, se utilizó un componente del propio KNIME. Posteriormente, se realizó la reducción de dimensiones utilizando TSNE para poder representarlo (se reducen las 768 dimensiones a 2) y finalmente, se introduce en el csv el clúster al que pertenece cada documento para poder continuar con la extracción de keywords.

Para indicar el clúster al que pertenece cada documento, se volverá a adaptar el código desarrollado en el Jupyter Notebook como se ha realizado en el apartado anterior. El Workflow de KNIME es el siguiente:

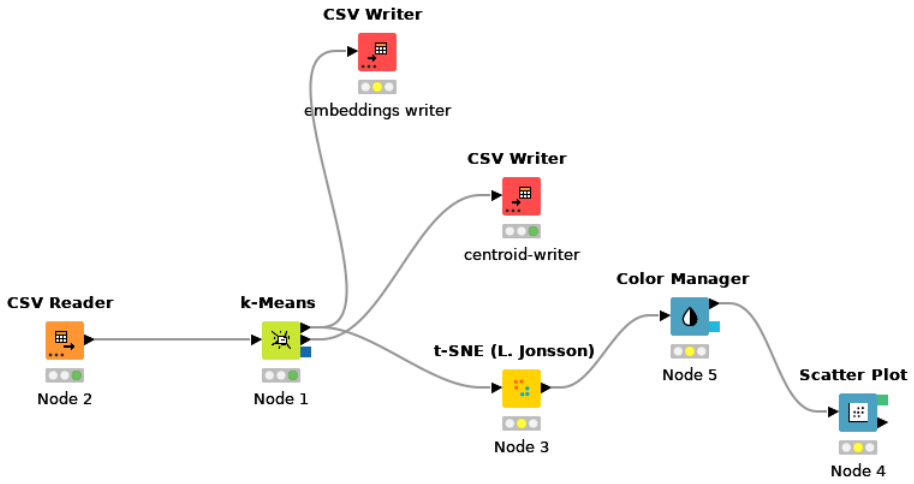

Figura 25: Workflow en KNIME para la clusterización

- **k-Means.** El componente de k-Means (incluido en KNIME), realiza la clusterización indicándole el número de clústeres previamente. Este componente, incluirá ya en el csv de entrada (generado en el apartado anterior) el clúster al que pertenece cada documento. Posteriormente, se puede ver una bifurcación, en la que por un lado se escribirán dos csv para el paso siguiente (centroid-writer y embeddings writer), y por otro se realizará la visualización.
- **Visualización.** A la hora de poder obtener la visualización de como quedan los documentos en una gráfica tras la clusterización, se utilizan los siguientes componentes:
  - **t-SNE.** Para poder obtener este componente, hay que instalar en KNIME la librería: KNIME Statistics Node Labs. En el componente de t-SNE, habrá que configurar el número de dimensiones que se quieran obtener de salida, en base a las columnas de entrada que hay que reducir (las 768 columnas relativas a las dimensiones).
  - **Color Manager.** Este componente, se encargará de aplicar un color a los componentes de cada clúster, para poder visualizarlos e identificarlos de una forma más clara.
  - **Scatter Plot.** Este componente, sirve para visualizar todos los documentos distribuidos en los clúster.

Selección del método de obtención de keywords

Una vez realizada la clusterización, para realizar TF-IDF, se obtienen todos los abstract de los documentos de un mismo clúster y se juntan en un fichero, ya que TF-IDF, trabaja obteniendo las veces que se repiten unas palabras en un documento, es decir, es estadístico. Posteriormente, se obtiene un conjunto de stopwords (palabras que son irrelevantes como conectores, artículos...) para que se centre en las palabras importantes. Finalmente, se obtienen las 10 keywords con una puntuación más alta, realizando una limpieza de las palabras, eliminando algunos verbos, palabras duplicadas, obteniendo las más largas (cuando hay unas palabras que contienen a otras) y eliminando más stopwords por si acaso no se han eliminado completamente.

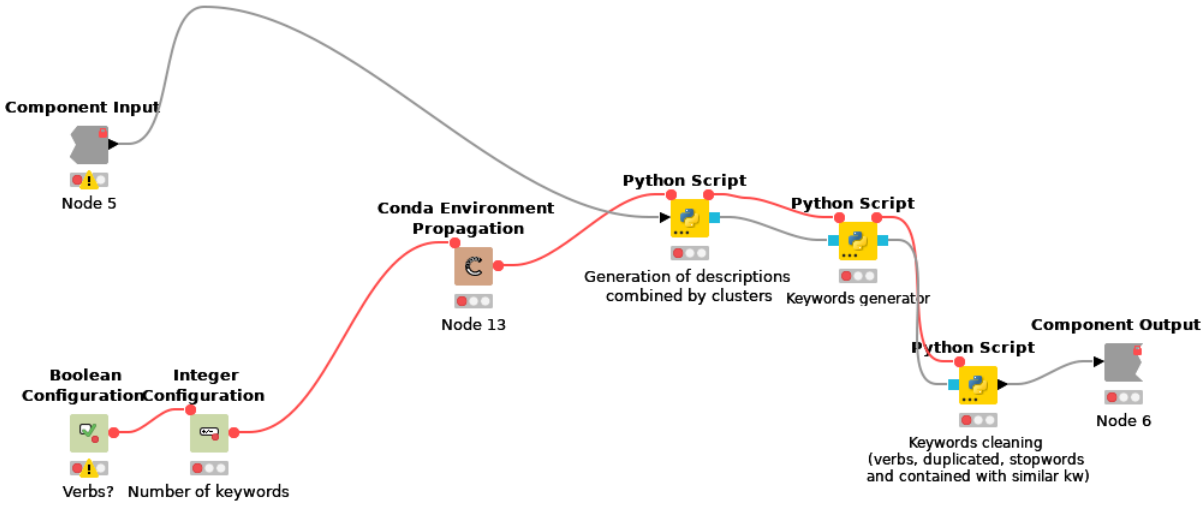

Figura 26: Implementación de TF-IDF en KNIME

En el componente de KNIME, la entrada es el fichero csv (embeddings writer Fig. 25) que contiene los embeddings, título, abstract y el clúster al que pertenecen. Los módulos que tienen configuration en su nombre, sirven para añadir variables a la selección de las keywords, como si se quieren eliminar los verbos o el número de keywords que se quieren obtener. El componente de Conda, sirve para instalar las librerías necesarias en Python. Finalmente, los componentes de Python, se separan en 3 realizando cada uno una acción de las descritas antes:

- El primero, genera los documentos con todos los abstracts separados por clústeres y se los pasa al siguiente.
- El segundo, obtiene las keywords aplicando TF-IDF
- El último realiza la limpieza de las keywords obtenidas en el componente de Python anterior.

Prompt y consultas con gpt-3

Para obtener un topic en base a las keywords anteriores de forma automática, se va a utilizar GPT-3 (Davinci al ser el mejor de todos) debido a que es bueno realizando tareas en base a ejemplos con una prompt de entrada, por lo que se realizaron los siguientes pasos para obtener la prompt de ejemplo y realizar las llamadas:

- Como primer paso, se escogieron varios temas y se extrajo toda la entrada de la Wikipedia del mismo, aplicando el Workflow anterior sobre el texto para obtener las keywords relacionadas con ese tema. Los temas escogidos con sus respectivas keywords para la prompt son:
  - **Fútbol:** football, team, ball, the game, the ball, women, players, being, of the game, kick, game, fifa, cups, penalty, league
  - **Biología:** cells, plants, proteins, dna, genes, molecules, eukaryotic, animals, species, hydrogen, transcription, organisms, bacteria, genome, tissue

- **Inteligencia Artificial:** intelligence, its, machines, artificial intelligence, symbols, learning, ai research, humans, artificial, neural networks, general intelligenca, problem, algorithmic, neural, agents
- **Electrónica:** circuits, transistor, digital circuits, being, of electronics, electronic devices, devices, semiconductor, electronics, digital, voltages, mosfet, analog circuits, radio, electrnoic systems
- **Desarrollo de Software:** software, software development, its, patents, the software, software is, user,the computer, software engineering, web, of software, engine, operating system, application software, instructions
- **Geología:** geology, rocks, earth, sedimentary, rock units, mineral, the earth, the rocks, deforming, fault, of rocks, igneous, of the earth, geologists, tectonics
- **Economía:** economics, price, economies, of economic, income, markets, monoey, neoclassical, demand, keynesian, macroeconomics, trade, economists, quantity, goods

La elección de los temas, se debe a que tienen que ser unos temas relacionados y no relacionados con el nuevo tema que se quiere inferir, para no introducir un bias (si todos los temas están muy relacionados con un mismo campo, en el nuevo ejemplo tenderá a producir respuestas que también estén relacionadas con ese campo aunque las keywords sean muy distintas).

- Una vez obtenidos los temas y las keywords, se formó la prompt en la que se utiliza la cadena "\n###\n" para separar cuando empieza una nueva lista de keywords. La prompt final queda así:

```
Infer 'Topic' from 'Keywords':\n###\n'Keywords': 'football' ,'team' ,'ball' ,'the game' ,'the
ball' ,'women' ,'players' ,'being' ,'of the game' ,'kick' ,'game' ,'fifa' ,'cups' ,'penalty'
,'league' \n'Topic':'Football'\n###\n'Keywords': 'cells' ,'plants' ,'proteins' ,'dna' ,'genes'
,'molecules' ,'eukaryotic' ,'animals' ,'species' ,'hydrogen' ,'transcription' ,'organisms'
,'bacteria' ,'genome' ,'tissue' \n'Topic':'Biology'\n###\n'Keywords': 'intelligence' ,'its'
,'machines' ,'artificial intelligence' ,'symbols' ,'learning' ,'ai research' ,'humans'
,'artificial' ,'neural networks' ,'general intelligence' ,'problem' ,'algorithmic' ,'neural'
,'agents' \n'Topic':'Artificial Intelligence'\n###\n'Keywords': 'circuits' ,'transistor' ,'digital
circuits' ,'being' ,'of electronics' ,'electronic devices' ,'devices' ,'semiconductor'
,'electronics' ,'digital' ,'voltages' ,'mosfet' ,'analog circuits' ,'radio' ,'electronic systems'
\n'Topic':'Electronics'\n###\n'Keywords': 'software' ,'software development' ,'its' ,'patents'
,'the software' ,'software is' ,'user' ,'the computer' ,'software engineering' ,'web' ,'of
software' ,'engine' ,'operating system' ,'application software' ,'instructions'
\n'Topic':'Software Development'\n###\n'Keywords': 'geology' ,'rocks' ,'earth' ,'sedimentary'
,'rock units' ,'mineral' ,'the earth' ,'the rocks' ,'deforming' ,'fault' ,'of rocks' ,'igneous'
,'of the earth' ,'geologists' ,'tectonics' \n'Topic':'Geology'\n###\n'Keywords': 'economics'
,'price' ,'economies' ,'of economic' ,'income' ,'markets' ,'money' ,'neoclassical' ,'demand'
,'keynesian' ,'macroeconomics' ,'trade' ,'economists' ,'quantity' ,'goods'
\n'Topic':'Economics'\n###\n
```

Figura 27: Prompt generada con los temas anteriores

- Finalmente, para obtener un nuevo tema, únicamente hay que añadir a la prompt la cadena

separador, las keywords y el texto “Topic” para que Davinci escriba el tema asociado a las keywords anteriores, parseando el resultado para obtener el tema asociado a las keywords

Finalmente, para este proceso se decidió utilizar KNIME. En el Workflow, queda de la siguiente forma:

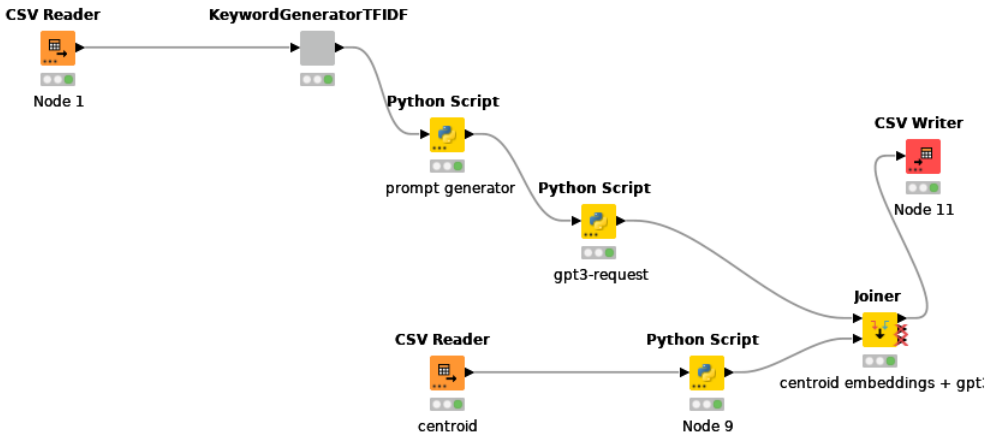

Figura 28: Extracción del tema y procesado del mismo para el paso siguiente

- El componente denominado KeywordGeneratorTFIDF, es el asociado al Workflow anterior, por lo que en este, se obtienen las keywords relacionadas con cada clúster.
- Los dos Python script siguientes, sirven para realizar la llamada a GPT-3 y obtener los topics
- Por el lado de abajo en el CSV reader, se obtienen los centroides con los embeddings y se unen a la tabla obtenida anteriormente, para que de esta forma, se pueda obtener la matriz de correlación.

Representación en la matriz de correlación

A la hora de representar los resultados en la matriz de correlación, se utilizó también un Workflow de KNIME, al que había que pasarle una tabla para su representación. El Workflow es el siguiente:

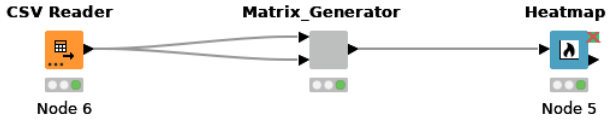

Figura 29: Visión general del Workflow

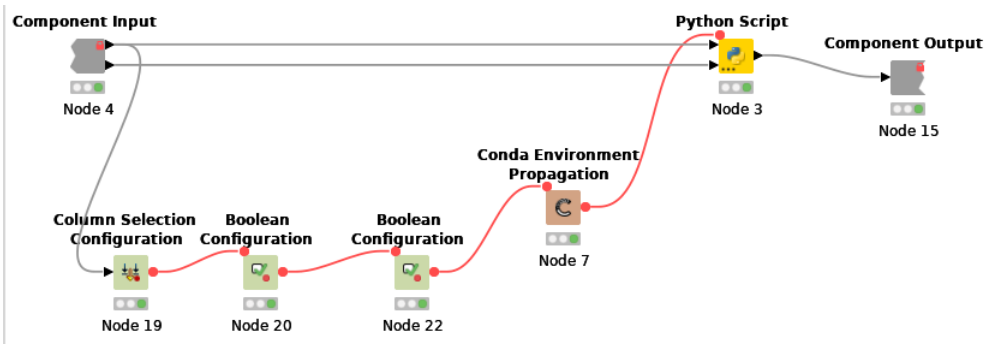

Figura 30: Visión específica del componente de generación de la matriz

- En la primera figura (29), se puede ver que el Workflow está compuesto por el lector del csv generado en el apartado anterior, el componente que genera la matriz y el componente que genera la visualización de la matriz.
- En la segunda figura (30), aparecen todas las configuraciones del componente (columna de la que se extrae el título, si se desea realizar normalización o también realce). Finalmente, se utiliza un componente de Conda para instalar las librerías necesarias, y el Python script generará la matriz de correlación, con los parámetros indicados por el usuario.

4. Resultados y Discusión

Los resultados obtenidos comienzan a partir de la clusterización. En los pasos anteriores, ya se ha explicado la toma de decisiones debido a que eran más claras (selección del modelo de lenguaje). A partir de la implementación de la clusterización con k-means, surge la duda del número de clústeres a escoger, siendo esta entre 6 y 7, ya que con números de clústeres mayores o menores, se reparten los documentos de una forma menos equilibrada 31 32.

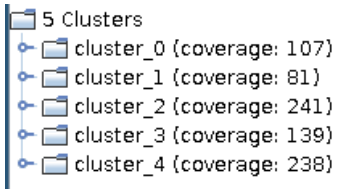

Figura 31: Reparto de los documentos en 5 clústeres

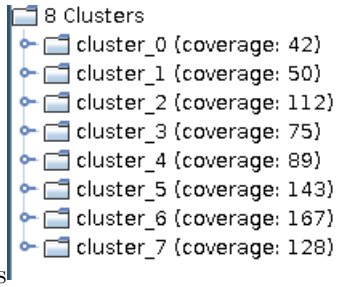

Figura 32: Reparto de los documentos en 8 clústeres

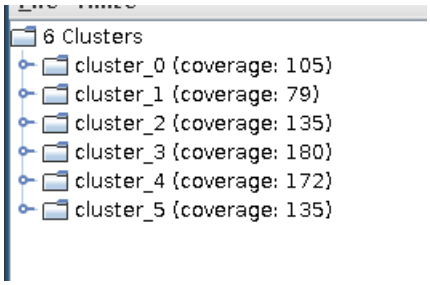

Figura 33: Reparto de los documentos en 6 clústeres

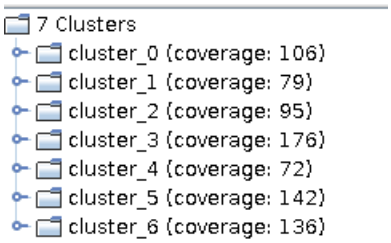

Figura 34: Reparto de los documentos en 7 clústeres

Como se puede observar en el reparto con 6 y 7 clústeres, hay muchos que son similares entre las dos figuras, pero se puede apreciar que el clúster_4 generado en la imagen 34 se genera en base al clúster_2 y al clúster_4 de la figura 33, debido a que en el resto de clústeres se mantienen los ficheros casi idénticos.

A la hora de la visualización de los clústeres en el mapa de dos dimensiones, aparecen representados de la siguiente forma:

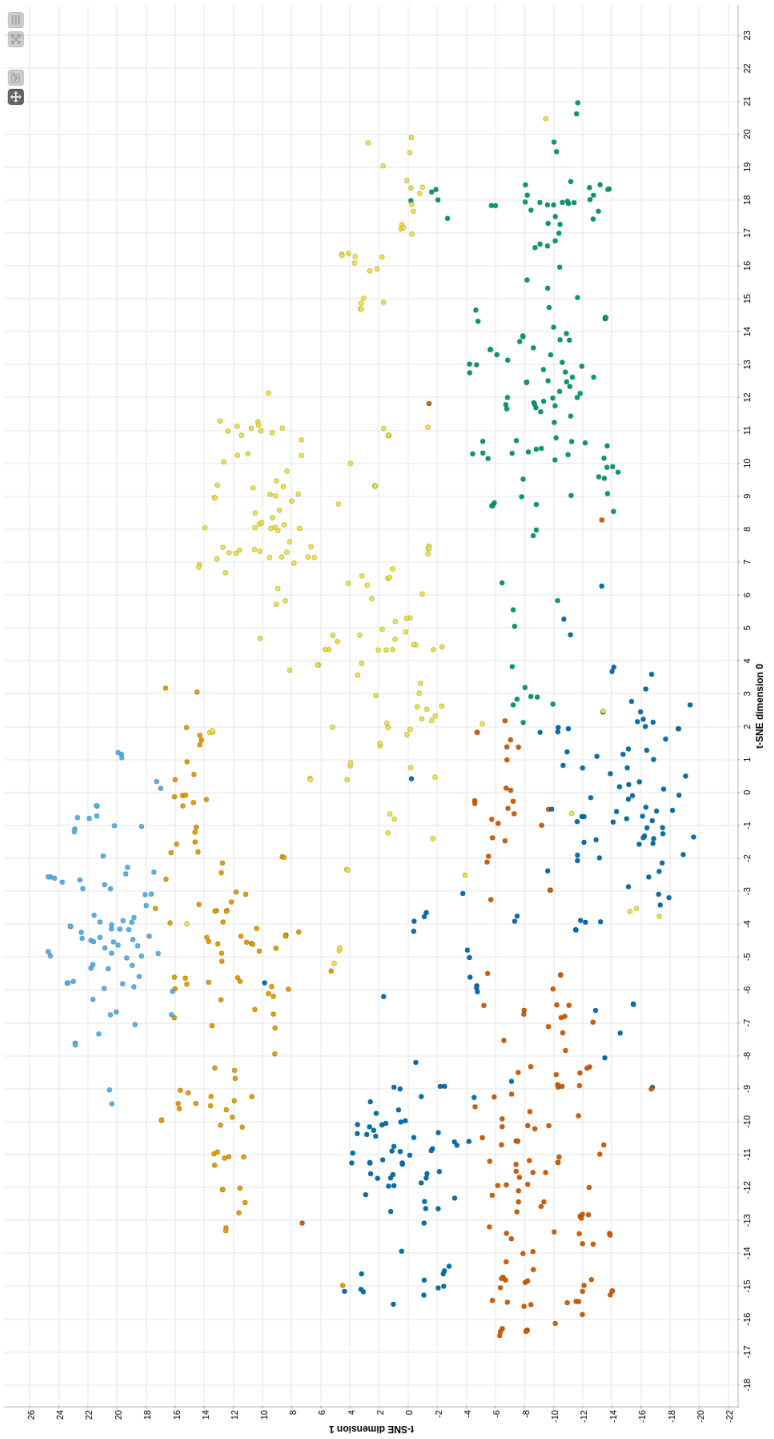

Figura 35: Visualización de los 6 clústeres

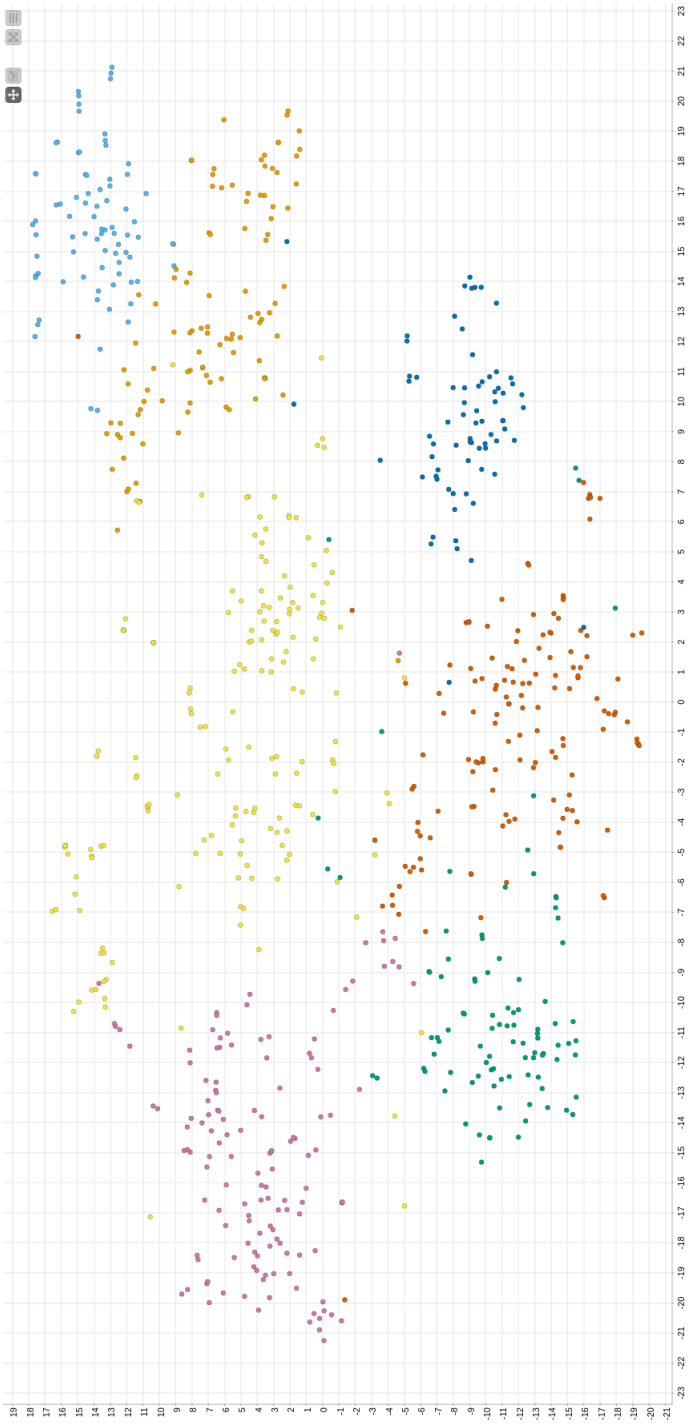

Figura 36: Visualización de los 7 clústeres

Al ser bastantes similares los resultados con ambas distribuciones de clústeres, se decidió seguir el proyecto utilizando las dos opciones, es decir, realizando todos los procesos restantes con 6 y con 7 clústeres.

Para el apartado de la obtención de las keywords, se obtuvieron estos resultados para las distintas configuraciones de clústeres:

RowID	Cluster	Keywords
Row0	cluster_0	[('quantum computing', 116.25910677629498), ('elliptic', 36.14709844115208), ('quantum cryptography', 31.782220954747 65), ('cryptographic', 29.872132116131 35), ('nist', 28.917678752921667), ('signature', 28.115025690738307), ('entanglement', 25.670240848065 41), ('cryptosystems', 23.495613986748854), ('public key', 21.688259064891252), ('entropies', 20.780671162719617), ('business', 19.902499913979 91), ('computing and blockchain', 19.880904142633646)]
Row1	cluster_3	[('patient', 129.57359566616635), ('code generation', 62.97368392049311 6), ('screening', 48.79858289555531), ('source code', 36.671772640093444), ('pain', 34.33974351909447 6), ('liver', 32.532388597036 88), ('generated code', 32.532388597036 88), ('ai ethics', 32.532388597036 88), ('automatically', 31.486841960246 558), ('therapy', 30.725033674979272), ('robots', 30.679487038 8956), ('therapeutic', 28.917678752921667), ('chat', 28.917678752921667), ('explanations', 28.115025690738307), ('copilot', 27.110323830864 06), ('covid-19', 25.835357505843333)]
Row2	cluster_1	[('chip', 76.69871759547239), ('quantum computation', 57.32219946608989), ('trapped', 56.028002583787 53), ('optical', 54.092779778594 7), ('quantum photonic', 45.1838730514401), ('qubit', 44.404520713168225), ('silicon photonic', 39.761808285267 29), ('waveguides', 37.954453363209 69), ('qds', 37.894165061429 89), ('gates', 26.892633269401 86), ('superconducting', 24.447848426728 96), ('quantum dots', 23.225456005392513)]
Row3	cluster_4	[('twin', 153.88030421296662), ('business model', 67.01045853078114), ('computing adoption', 41.569163207324 9), ('des', 39.761808285267 29), ('enterprises', 30.096463284554986), ('service providers', 28.115025690738307), ('digitalization', 27.669329148703778), ('dt', 25.670240848065 41)]
Row4	cluster_5	[('tls', 218.68994556897 01), ('mec', 135.68555876834574), ('mobile edge computing', 52.413292739670524), ('iot', 48.542682711702417), ('federated learning', 37.954453363209 69), ('sdn', 37.954453363209 69), ('vehicular', 36.14709844115208), ('network edge', 36.14709844115208), ('servers', 35.436158383427646), ('mobile device', 33.00459537 60841), ('task offloading', 32.532388597036 88)]
Row5	cluster_2	[('tourism', 112.05600516757146), ('spatial', 55.707489621974 68), ('land use', 50.605937817612 92), ('fungal', 41.569163207324 9), ('gis', 39.761808285267 29), ('parenting', 30.725033674979272), ('geographic information', 30.725033674979272), ('soils', 30.725033674979272), ('geography', 28.917678752921667), ('urban', 27.183902321533537), ('fungi', 25.302968908806446)]

Showing 1 to 6 of 6 entries

Previous 1 Next

Figura 37: Keywords con 6 clústeres

Show 10 entries

Search:

RowID	Cluster	Keywords
Row0	cluster_0	[('quantum computing', 145.0), ('elliptic', 40.0), ('cryptographic', 36.0), ('signature', 32.54586248341341), ('qubits', 32.54586248341341), ('nist', 32.0), ('entanglement', 29.715787484855717), ('cryptosystems', 26.0), ('quantum cryptography', 25.0), ('entropies', 24.055637487740340), ('public key', 24.0)]
Row1	cluster_1	[('chip', 95.0), ('qubit', 77.82706246603364), ('quantum computation', 71.0), ('optical', 67.0), ('trapped', 62.0), ('quantum photonic', 50.0), ('silicon photonic', 44.0), ('qds', 43.886162477644115), ('waveguides', 42.0), ('gates', 31.130824984134563), ('superconducting', 28.300749985557687 5), ('quantum dots', 26.885712486298093)]
Row2	cluster_3	[('patient', 105.0), ('code generation', 52.21153669367311), ('ethics', 51.0), ('treatment', 48.0), ('source code', 42.45112497836531 4), ('pain', 38.0), ('automatically', 37.0), ('screen', 36.790974981249 94), ('liver', 36.0), ('documented', 36.0), ('therapy', 34.0), ('explanations', 32.54586248341341), ('therapeutic', 32.0), ('chat', 32.0), ('copilot', 30.0)]
Row3	cluster_4	[('computing adoption', 46.0), ('business model', 38.65009859142035), ('cobf', 22.0), ('service providers', 21.0), ('organisational', 20.0), ('cloud adoption', 20.0), ('small businesses', 20.0)]
Row4	cluster_2	[('business', 32.787962443028654), ('robotics', 24.0), ('digitalization', 23.242099959615246), ('twins', 22.640599988461 5), ('dynamic knowledge', 20.0), ('university of cambridge', 20.0), ('dtj', 20.0), ('use case', 19.506762466105 6565)]
Row5	cluster_5	[('tls', 258.0), ('mec', 157.06916241995165), ('offloading', 148.57893742427 86), ('mobile edge computing', 58.0), ('edge servers', 53.771424972596 06), ('iot', 48.55938741562471), ('edge nodes', 44.0), ('latency', 42.71853002209618), ('federated learning', 42.0), ('sdn', 42.0), ('city', 40.25863743004784), ('vehicular', 40.0), ('caching', 40.0)]
Row6	cluster_6	[('tourism', 124.0), ('spatial', 69.0), ('land use', 56.0), ('fungal', 46.0), ('gis', 44.0), ('cities', 39.84359993076899 5), ('urban', 37.97202666863077 14), ('parenting', 34.0), ('geographic information', 34.0), ('soils', 34.0), ('geography', 32.0), ('fungi', 28.0)]

Showing 1 to 7 of 7 entries

Previous 1 Next

Figura 38: Keywords con 7 clústeres

Como se puede apreciar en las imágenes 37 38, los dos primeros clústeres están orientados hacia la computación cuántica, por lo que en este apartado se puede observar que la extracción de los documentos realizada por Semantic Scholar, no haya sido la mejor. Otra posibilidad, es que al no saber el contenido de los documentos en concreto o que esté en español, puede ser que aunque perteneciesen a un tema de los iniciales, en el abstract se hablase más de otros temas o no se entienda bien y por ello, aparezcan en otros clústeres.
Finalmente, puede ser que el método de extracción/limpieza de keywords no sea el correcto. Una vez pasadas las prompt generadas en GPT-3, se obtuvieron los resultados siguientes:

- **6 clústeres:**
  - **clúster_0.** Quantum Computing
  - **clúster_1.** Quantum Computing
  - **clúster_2.** Tourism
  - **clúster_3.** Healthcare
  - **clúster_4.** Digital Transformation
  - **clúster_5.** Mobile Edge Computing
- **7 clústeres:**
  - **clúster_0.** Quantum Computing
  - **clúster_1.** Quantum Computing
  - **clúster_2.** Business
  - **clúster_3.** Medical
  - **clúster_4.** Cloud Computing Adoption
  - **clúster_5.** Mobile Edge Computing
  - **clúster_6.** Tourism

Al ver la obtención de los topics por parte de GPT-3, se puede apreciar que tienen bastante relación con las keywords obtenidas en la fase anterior, por lo que para mejorar este aspecto (si fuese necesario, porque habría que comprobar exactamente el contenido de los documentos), habría que centrarse o en la parte de extracción de keywords o bien, en la parte de la obtención de los documentos.

Para la parte de visualización en la matriz de correlación, se decidió utilizar el centroide del clúster, debido a que es equidistante de todos los sitios del clúster, por lo que se obtendría el documento más representativo del clúster para comparar con el resto. Otra idea propuesta, era obtener la media de los embeddings del total de documentos de cada clúster, pero esta idea no se llegó a desarrollar, porque cabe la posibilidad de que si dentro de un clúster se hablaba demasiado de un tema y muy poco de otros, el embedding seleccionado no sería representativo. La visualización de la matriz de correlación para ambos casos (6 y 7 clústeres) es la siguiente:

Figura 39: Matriz de correlación para 6 clústeres

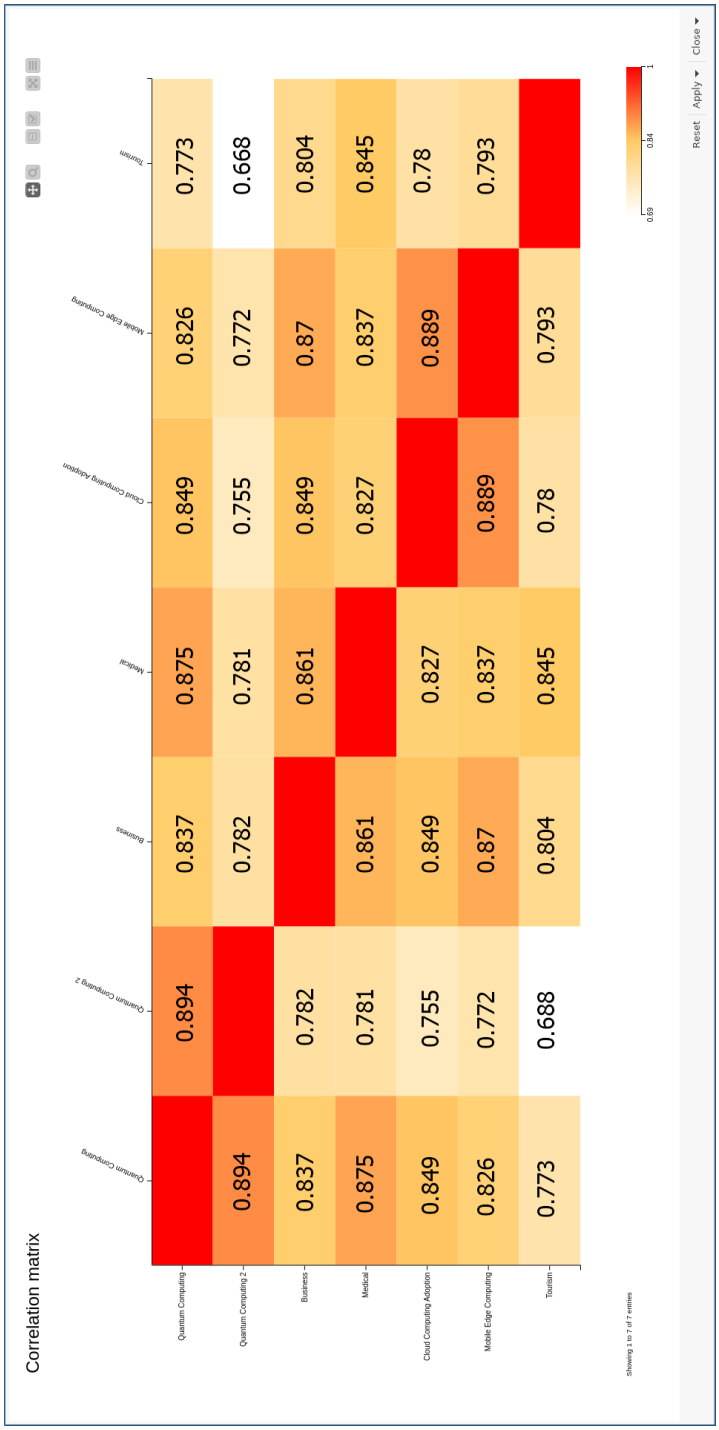

Figura 40: Matriz de correlación para 7 clústeres

Finalmente, al haber dos topics iguales (Quantum Computing), se decidió renombrar la salida del segundo para que así se pudiese representar la matriz (al ser dos iguales cuando se crea el diccionario de Python, la clave estaba repetida).

Con los resultados obtenidos en la matriz de correlación, se aprecia que los topics de turismo y Quantum Computing 2, son los que menos relación tienen con el resto de temas, debido a que entre el resto, no hay ninguna fila que baje de 0.8. El caso de turismo, tiene sentido debido a que este topic, se podría decir que hace referencia al tema de geografía escogido. Por otro lado el topic de Quantum Computing 2, la poca relación puede ser porque los documentos de este topic sean una parte más teórica de la computación cuántica.

5. Licencia software y documental

En este momento, al ser un proyecto realizado en el área de innovación y activos de la empresa, se está realizando un análisis de licencias y dependencias de las librerías utilizadas para poder confirmar el tipo de licencia para los desarrollos realizados.
En cualquier caso, la empresa deberá ser consultada para el acceso al código fuente y su posible utilización más allá de objetivos puramente académicos dentro del desarrollo del Trabajo Fin de Grado del alumno.

Finalmente, no hay que olvidar que este documento, por defecto, está al amparo de la licencia 7.3, por su inclusión en el Repositorio Institucional de Documentos de la Universidad de Zaragoza: ZAGUAN.

Figura 41: Licencia de ZAGUAN

6. Conclusiones y Trabajo futuro

Como conclusión, en este Trabajo de Fin de Grado, se han aprendido muchos conceptos que no se han visto a lo largo de la carrera, por lo que ha requerido un gran proceso de investigación y formación previos a la realización del mismo, dedicándose horas parecidas a todos procesos (implementación de la solución, comprensión de los nuevos conceptos e investigación de las posibilidades). Por otro lado, sin la ayuda de los compañeros de la empresa, no hubiese sido posible la realización del mismo, debido a que muchos de los errores que se dieron a la hora de la implementación, habrían tardado mucho más en resolverse (errores comunes al trabajar con nuevas tecnologías, librerías, instalaciones de dependencias...) o a la comprensión completa de los nuevos conocimientos.

Dentro de las asignaturas que se cursan en la escuela, las más relacionadas con el Trabajo de Fin de Grado, serían Inteligencia Artificial (IA), debido a que en ella, se explican muchos de los conocimientos básicos para llegar a entender las arquitecturas y técnicas más complejas que se han propuesto en el proyecto y Almacenes y Minería de Datos, debido a que en la asignatura se utiliza KNIME, por lo que de esta forma se hizo más sencilla la adaptación a la hora de trabajar con esta herramienta. Otra característica que se aprendió en esta asignatura es analizar grandes cantidades de datos y gráficos, por lo que de esta forma, se pudieron entender las representaciones de los clústeres.

Finalmente, este proyecto tiene mucho trabajo futuro posible, por lo que los apartados en los que más se podría seguir avanzando en el desarrollo serían:

- **Optimización.** Muchas de las zonas del código implementado en Python, se podrían optimizar para evitar mucho tiempo de espera en algunos componentes, evitando bucles o eliminando entornos de Conda en algunos Workflows (utilizando un entorno de Conda para todo el proyecto con todas las dependencias ya instaladas, debido a que la instalación de las mismas en el componente de KNIME es un proceso muy tedioso).

- **Automatización.** Actualmente, el proyecto está separado en varios Workflows de KNIME para cada una de las fases del mismo, pero para una mejor ejecución posterior, se podría unificar todo en un único Workflow, para que de esta forma, fuese más sencilla la ejecución de todo el proyecto de forma completa.

- **Post-procesado.** Se podría realizar una fase de post-proceso en las fases finales de proyecto, como por ejemplo en la fase de extracción de keywords. Aunque actualmente se realiza, podría ser mucho más exacta ya que todavía se siguen introduciendo algún tipo de stopwords o palabras repetidas. Otro lugar en el que se podría realizar un post-procesado, sería después de obtener los títulos de los temas desde GPT-3, ya que actualmente, se utiliza en crudo la salida de GPT-3.

- **Nuevas fuentes de información.** Como fuentes de entrada de este proyecto, se escogieron los documento científico-técnicos de Semantic Scholar, pero para versiones posteriores, se podrían obtener los datos desde cualquier fuente de información (adaptando el modelo de lenguaje a la misma), pudiendo analizar desde cualquier sitio los temas con más relevancia y acotando los documentos, ya que ha podido ser una fuente de problemas como se ha comentado.

Referencias

[1] *BERT (modelo de lenguaje) - Wikipedia, la enciclopedia libre.* (2022). Retrieved 17 June 2022, from https://es.wikipedia.org/wiki/BERT_(modelo_de_lenguaje)

[2] Briceño, B., & Fernandez, E. (2022). *¿Qué son los word embeddings y para qué sirven?.* Retrieved 17 June 2022, from https://blogs.iadb.org/conocimiento-abierto/es/que-son-los-word-embeddings/

[3] Angelov, D. (2022). *Top2Vec: Distributed Representations of Topics.* Retrieved 18 June 2022, from https://arxiv.org/abs/2008.09470

[4] *GPT-3 - Wikipedia, la enciclopedia libre.* (2022). Retrieved 17 June 2022, from https://es.wikipedia.org/wiki/GPT-3

[5] *How to do AVERAGE and MAX word embedding for long sentences?.* (2022). Retrieved 17 June 2022, from https://towardsdatascience.com/how-to-do-average-and-max-word-embedding-for-long-sentences-f3531e99d998

[6] *La guía máxima para el modelo de lenguaje GPT-3 de OpenAI.* (2022). Retrieved 17 June 2022, from https://www.twilio.com/blog/la-guia-maxima-para-el-modelo-de-lenguaje-gpt-3-de-openai

[7] *Modelos de lenguaje en el NLP.* (2022). Retrieved 17 June 2022, from https://monica-echeverrt.medium.com/modelos-de-lenguaje-en-el-nlp-8922dc34753b

[8] *NLP: Word Embedding Techniques Demystified.* (2022). Retrieved 17 June 2022, from https://towardsdatascience.com/nlp-embedding-techniques-51b7e6ec9f92#:~:text=4.-,Doc2Vec,every%20document%20in%20the%20corpus.

[9] *Transformer: domina el mundo (NLP): explicación SENCILLA.* (2022). Retrieved 17 June 2022, from https://www.themachinelearners.com/transformer/#:~:text=que%20nos%20importan.-,Auto-atenci%C3%B3n%20por%20multi-cabeza,para%20despu%C3%A9s%20concatenar%20los%20resultados.

[10] *OpenAI documentation.*(2022). Retrieved 17 June 2022, from https://beta.openai.com/docs/introduction

[11] Cohan, A., Feldman, S., Beltagy, I., Downey, D., & Weld, D. (2022). *SPECTER: Document-level Representation Learning using Citation-informed Transformers.* Retrieved 17 June 2022, from https://arxiv.org/abs/2004.07180

[12] *Text Classification with BERT using Transformers for long text inputs.* (2022). Retrieved 17 June 2022, from https://medium.com/analytics-vidhya/text-classification-with-bert-using-transformers-for-long-text-inputs-f54833994dfd

[13] Beltagy, I., Peters, M., & Cohan, A. (2022). *Longformer: The Long-Document Transformer.* Retrieved 17 June 2022, from https://arxiv.org/abs/2004.05150

[14] *A review of BERT based models.* (2022). Retrieved 17 June 2022, from https://towardsdatascience.com/a-review-of-bert-based-models-4ffdc0f15d58

[15] Sun, C., Qiu, X., Xu, Y., & Huang, X. (2022). *How to Fine-Tune BERT for Text Classification?.* Retrieved 17 June 2022, from https://arxiv.org/abs/1905.05583

[16] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., & Gomez, A. et al. (2022). *Attention Is All You Need.* Retrieved 17 June 2022, from https://arxiv.org/abs/1706.03762

[17] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., & Chen, D. et al. (2022). *RoBERTa: A Robustly Optimized BERT Pretraining Approach.* Retrieved 18 June 2022, from https://arxiv.org/abs/1907.11692

[18] *sentence-transformers/all-roberta-large-v1 · Hugging Face.* (2022). Retrieved 18 June 2022, from https://huggingface.co/sentence-transformers/all-roberta-large-v1

[19] *sentence-transformers/allenai-specter · Hugging Face.* (2022). Retrieved 18 June 2022, from https://huggingface.co/sentence-transformers/allenai-specter

[20] B, C. R. J. G., Davoud, M. & Joerg, S. (2013, 18 de junio). *Density-Based clustering Based on Hierarchical Density Estimates. SpringerLink.* Retrieved 18 June 2022, from https://link.springer.com/chapter/10.1007/978-3-642-37456-2_14