

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
SANTIAGO - CHILE



“DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE BÚSQUEDA SEMÁNTICA SOBRE EL REPOSITORIO USM”

DYLAN OTEIZA CANALES

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN INFORMÁTICA

Profesor Guía: Andrés Moreira Wenzel
Profesor Correferente: Ricardo Ñanculef Alegría

Diciembre - 2024

DEDICATORIA

Dedicado a mi familia; mi papá, mi mamá, mi hermano, quienes me han acompañado y apoyado en cada momento de mi vida.

AGRADECIMIENTOS

Agradezco a mi familia, mis padres, mi hermano, por todo el apoyo, el cariño y la paciencia que han mostrado conmigo a lo largo de todos estos años, durante el transcurso de esta carrera y durante cada etapa en mi vida. Soy feliz todos los días de mi vida gracias a ellos.

Agradezco a mis compañeros y amigos, con quienes hemos compartido y seguimos compartiendo una aventura juntos, ayudándonos y apoyándonos en todo momento.

Agradezco a los profesores, quienes trabajan muy duro cada día para enseñarnos lo que saben de la mejor forma posible y ayudarnos siempre a ser mejores personas. Agradezco particularmente a mi profesor guía, Andrés Moreira, por asumir la responsabilidad de tomar este rol durante el desarrollo de este trabajo, y por impartir las clases que más he disfrutado durante mi paso por la carrera. Agradezco también al profesor Ricardo Ñanculef, por asumir el rol de profesor correferente de este trabajo.

Agradezco a los administradores del Repositorio USM, quienes desde el comienzo se mostraron abiertos a escuchar la propuesta de este trabajo, y fueron pronto a dar su apoyo y asistencia frente a cualquier solicitud. Son personas como ellas las que motivan a construir e inventar cosas nuevas.

Finalmente, agradezco a toda la comunidad y a cada individuo que trabaja día y noche para desarrollar y construir las herramientas que todos utilizamos en nuestros proyectos y en nuestras vidas. Es gracias a su esfuerzo que todo sigue adelante. Que este trabajo sea una demostración de su esfuerzo.

RESUMEN

Resumen— Con el paso del tiempo, el volumen de producción en lenguaje natural se hace cada vez más grande. Como consecuencia, el análisis y procesamiento eficiente de esta información se vuelve cada vez más complejo. Tareas como la recuperación de información sensible al contexto, que requieren de un análisis cuidadoso de la información, se vuelven cada vez más complejas de realizar utilizando sistemas de búsqueda tradicionales. En este trabajo, se expone el diseño e implementación de un sistema de búsqueda semántica, un sistema que utiliza representaciones en vector que capturan el contenido semántico de la información, utilizando el modelo SBERT, con el objetivo de mejorar la calidad de búsqueda del usuario. Este sistema se implementa sobre una muestra del corpus de documentos del Repositorio USM. Se expone, además, la utilización de estas representaciones en vector para obtener un *clustering* de los documentos de este repositorio, utilizando la herramienta BERTopic.

Palabras Clave—Recuperación de información, procesamiento de lenguaje natural, modelo de lenguaje de gran tamaño, búsqueda semántica, SBERT

ABSTRACT

Abstract— As time goes on, the volume of natural language production continues to grow. Consequently, the efficient analysis and processing of this information become increasingly complex. Tasks such as context-sensitive information retrieval, which require careful analysis of the information, are becoming more difficult to perform using traditional search systems. In this work, we present the design and implementation of a semantic search system, which utilizes vector representations capturing the semantic content of information, employing the SBERT model. The goal is to enhance user search quality. This system is implemented on a sample from the Repositorio USM document corpus. Additionally, the use of these vector representations is discussed for clustering documents within this repository, employing the BERTopic tool.

Keywords—Information retrieval, natural language processing, large language model, semantic search, SBERT

GLOSARIO

AKNN: Approximate k-nearest neighbor.

BERT: Bidirectional Encoder Representations from Transformers.

CNN: Convolutional neural network

DI: Departamento de Informática.

IR: Information Retrieval.

KNN: K-nearest neighbor.

PDF: Portable Document Format.

RAM: Random Access Memory.

RNN: Recurrent neural network.

UTFSM: Universidad Técnica Federico Santa María.

VSM: Vector space model.

ÍNDICE DE CONTENIDOS

RESUMEN	IV
ABSTRACT	IV
GLOSARIO	V
ÍNDICE DE FIGURAS	VIII
ÍNDICE DE TABLAS	IX
INTRODUCCIÓN	1
CAPÍTULO 1: DEFINICIÓN DEL PROBLEMA	3
1.1 Descripción e Interés del Problema	3
1.2 El Repositorio USM	5
1.3 Objetivos	6
1.3.1 Objetivo general	6
1.3.2 Objetivos específicos	6
CAPÍTULO 2: MARCO CONCEPTUAL	7
2.1 Recuperación de Información (<i>Information Retrieval</i>)	7
2.2 Búsqueda Semántica (<i>Semantic Search</i>)	9
2.2.1 Secuencia de Codificación (<i>Encoding pipeline</i>)	9
2.2.2 Secuencia de Búsqueda (<i>Search pipeline</i>)	10
2.3 La Arquitectura <i>Transformer</i>	11
2.3.1 El <i>Encoder</i>	14
2.3.2 El <i>Decoder</i>	21
2.4 BERT	23
2.5 SBERT	27
CAPÍTULO 3: PROPUESTA DE SOLUCIÓN	29
3.1 Descripción de la propuesta	29
3.2 Motivación para el uso de LLM	29
3.3 BERT: Uso	30
3.3.1 Búsqueda semántica simétrica	31
3.3.2 Búsqueda semántica asimétrica	32
3.4 Desarrollo de la propuesta	32
3.5 Modelo	33
3.6 Herramienta	35
3.7 Corpus	35
3.7.1 Datos en archivo plano	37
3.7.2 Archivos digitalizados	39
3.8 Modelamiento de tópicos con BERTopic	40

CAPÍTULO 4: VALIDACIÓN DE LA SOLUCIÓN	41
4.1 Modelo	41
4.2 Herramienta	43
4.3 Corpus	45
4.4 Modelamiento de tópicos con BERTopic	47
4.5 Encuesta de opinión	54
4.6 Resultados de la encuesta	55
4.7 Conclusiones de la encuesta	56
CAPÍTULO 5: CONCLUSIONES	58
5.1 Objetivo general	60
5.2 Objetivos específicos	61
5.3 Conclusiones finales	62
REFERENCIAS BIBLIOGRÁFICAS	64

ÍNDICE DE FIGURAS

1	Árbol del problema.	4
2	Proceso de recuperación de información.	8
3	Etapas de la búsqueda semántica.	9
4	La arquitectura <i>transformer</i>	12
5	Abstracción de la arquitectura <i>transformer</i>	12
6	Abstracción de los componentes principales de la arquitectura <i>transformer</i>	13
7	Detalle de los componentes principales de la arquitectura <i>transformer</i>	13
8	Detalle del componente <i>encoder</i> del <i>transformer</i>	14
9	Movimiento de los <i>tokens</i> de entrada a través de los <i>encoders</i>	15
10	Mecanismo de <i>Self-Attention</i> considerando la relación de la palabra "it" con todas las otras palabras que componen la oración, durante su codificación en vector.	16
11	Vectores de los <i>tokens</i> de entrada del <i>encoder</i> , y sus proyecciones en vectores q , k y v , obtenidos luego de multiplicar cada vector por las matrices W_Q , W_K y W_V	17
12	Resultado de las operaciones realizadas por la capa de <i>Self-Attention</i>	18
13	Cálculo realizado por el mecanismo de <i>Multi-Head Attention</i> , utilizando matrices	19
14	Resultado del mecanismo de atención de múltiples cabezas para el token "it" en el ejemplo de entrada de una oración en inglés	19
15	Para cada codificación inicial de cada <i>token</i> de entrada del <i>encoder</i> , la capa de <i>Positional Encoding</i> agrega una noción de posición a cada codificación.	20
16	Operación de adición y normalización realizada por la capa <i>Add & Normalize</i>	21
17	Vista general de la arquitectura <i>transformer</i>	22
18	La salida de la pila de <i>encoders</i> es convertida en un set de vectores que son pasados a cada <i>decoder</i> en la pila de <i>decoders</i>	22
19	Este proceso continúa hasta que el <i>decoder</i> entrega un símbolo especial que indica el fin de la salida.	23
20	Ejemplo de uso de BERT para la tarea de clasificación de correos <i>spam</i>	24
21	Vista general de la arquitectura de BERT, compuesto exclusivamente por una pila de <i>encoders</i>	24
22	Procesamiento de una entrada en BERT, modelada para la resolución de un problema de clasificación de texto.	25
23	Usamos la representación del token [CLS] para entrenar al clasificador.	25
24	Entrenamiento de BERT con la tarea de <i>Masked Language Model</i>	26
25	Entrenamiento de BERT con la tarea de <i>Two-sentence Tasks</i>	27
26	Arquitectura de SBERT	28
27	Comparación entre los modelos BERT, GPT y BART	30
28	Modelos pre-entrenados ofrecidos por SentenceTransformers y sus puntajes de rendimiento en diferentes tareas.	34
29	Rendimiento de los modelos <i>mpnet</i> , <i>minilm</i> y <i>distil</i> para búsqueda simétrica en el corpus de documentos	42

30	Rendimiento de los modelos <i>dot</i> y <i>cos</i> para búsqueda asimétrica en el corpus de documentos	43
31	Implementación de barra de búsqueda y selección de tipo de búsqueda en la herramienta.	44
32	Implementación de filtrado por departamento en la herramienta.	44
33	Vista de resultados de la herramienta	45
34	Búsqueda por secciones de documento	45
35	Mapa de tópicos y la distancia entre ellos.	49
36	<i>Clustering</i> jerárquico de tópicos.	51
37	Puntaje de las palabras más relevantes para cada tópico.	52
38	Matriz de similaridad entre tópicos.	53

ÍNDICE DE TABLAS

1	Atributos de los documentos disponibles en el archivo plano del corpus de datos.	37
2	Tópicos obtenidos con BERTopic.	47

INTRODUCCIÓN

Existen muchas áreas de trabajo donde se requiere del manejo de documentos producidos en lenguaje natural. Una de estas áreas corresponde a la producción académica universitaria, donde alumnos de diversas áreas, tanto de pregrado como de postgrado, están constantemente produciendo nuevos documentos de investigación sobre diversos temas. Sin embargo, el ritmo y volumen actuales de producción tienen como consecuencia el que tareas como la búsqueda de documentos, o la extracción de información relevante, que requieren de una lectura atenta del contenido de cada documento, resulten cada vez más complicadas.

En este trabajo, se expone el diseño e implementación de una aplicación de búsqueda semántica, desarrollada utilizando el modelo Sentence-BERT (SBERT), sobre el corpus de documentos del Repositorio USM, como una propuesta de solución a esta problemática en un contexto local.

La búsqueda semántica es una técnica que considera el sentido semántico tanto de la búsqueda de un usuario, como de la información disponible para la búsqueda, en lugar de solo considerar, por ejemplo, coincidencias literales entre la consulta y la información. Esta metodología de búsqueda se enmarca en el ámbito de la recuperación de información, un área dentro del campo de ciencias de la información que abarca la búsqueda de documentos, o del contenido de estos documentos.

SBERT es una modificación del modelo BERT, un modelo de lenguaje basado en la arquitectura Transformer, con el cual es posible obtener representaciones en vector de oraciones y párrafos escritos en lenguaje natural. Posteriormente, es posible utilizar estas representaciones para la construcción de un sistema de búsqueda semántica.

El Repositorio USM es un repositorio digital administrado por el Área de Archivo Digital, Dirección de Información y Documentación Bibliográfica Institucional de la UTFSM ¹. Para la construcción de esta aplicación, se considerarán trabajos de alumnos de pregrado de la universidad de acceso abierto.

El trabajo expuesto en este documento pretende dar un ejemplo del potencial de un modelo de lenguaje natural del estado del arte, el modelo SBERT, para resolver una problemática relacionada con el manejo de una gran cantidad de documentos producidos en lenguaje natural.

La estructura de este documento es la siguiente: en el Capítulo 1, se entrega un contexto para el problema y una definición en detalle. En el Capítulo 2, se entrega el marco conceptual del problema, partiendo desde el campo más amplio, la recuperación de información, armando hasta lo más particular, el modelo SBERT, con el cual obtendremos nuestras representaciones en vector. En el Capítulo 3, se entrega la teoría y arquitectura detrás de la propuesta de

¹<https://repositorio.usm.cl/>

solución, que incluye la construcción del sistema y el uso de las representaciones para obtener tópicos. En el Capítulo 4, se detalla la construcción del sistema, en base a la propuesta planteada en el capítulo anterior, se expone la obtención de tópicos, y se dan opiniones de usuarios del sistema sobre el mismo. Finalmente, en el Capítulo 5, se entregan las conclusiones finales del trabajo desarrollado, analizando los objetivos iniciales planteados, las áreas de trabajo que quedan pendientes, los principales aportes y limitaciones de la propuesta, y aprendizaje de distintas áreas de desarrollo.

CAPÍTULO 1

DEFINICIÓN DEL PROBLEMA

1.1. Descripción e Interés del Problema

Con el paso del tiempo, la producción de documentos en lenguaje natural crece, y lo hace a un ritmo cada vez más acelerado. Cada nueva producción representa el aporte único de un autor sobre cierto tema en particular, el cual se ha apoyado en el trabajo de otros autores para su desarrollo, y cuyo trabajo servirá a su vez de apoyo para otros autores en el futuro.

Sin embargo, el volumen de producción actual es tan grande, que resulta cada vez más difícil manejarlo de manera eficiente. Cuando se quiere, por ejemplo, investigar el estado del arte de un tema en particular, lo común es utilizar técnicas de búsqueda tradicionales, tales como búsqueda por coincidencia de términos o filtrado simple, para encontrar documentos que sean relevantes para nuestra investigación.

Las técnicas tradicionales de búsqueda resultan muy útiles; sin embargo, no toman en cuenta el contenido o significado semántico detrás del texto de cada documento, el cual solo puede ser extraído luego de una lectura atenta de cada documento por parte del investigador. Realizar esta lectura atenta resulta especialmente difícil cuando se tiene una gran cantidad de documentos, de modo que, muchas veces, se da por terminada una búsqueda por razones de cansancio, lo cual resulta en un trabajo de investigación realizado de manera superficial.

En este trabajo, se propone la elaboración de un sistema de búsqueda semántica sobre un conjunto de información de prueba, con el fin de mejorar la precisión y calidad de las consultas que se realizan sobre este conjunto de información. La búsqueda semántica se define como una búsqueda con significado, la cual considera aspectos como el contexto y la intención del usuario al momento de formular una consulta, en lugar de solo considerar una coincidencia literal en el conjunto de información [Bast *et al.*, 2016]. La búsqueda semántica se enmarca dentro del contexto de la recuperación de información, la cual se define como la búsqueda de información dentro de un documento, búsqueda de documentos en sí mismos, o búsqueda de metadatos que describan a los datos [Luk, 2022].

Para la construcción de este sistema, se hará uso de Sentence-BERT (SBERT), una modificación de la red pre-entrenada BERT que obtiene representaciones en vector de oraciones y párrafos que capturan su significado semántico [Reimers y Gurevych, 2019]. Estas representaciones no solo pueden ser utilizadas para la construcción de un sistema de búsqueda semántica, sino que también pueden resultar útiles para otras aplicaciones. En este trabajo, en particular, aprovechando la obtención de estas representaciones para la construcción del sistema de búsqueda, también se explorará su uso en tareas de *clustering*, para dar cuenta del potencial de este enfoque en el análisis de grandes colecciones de documentos.

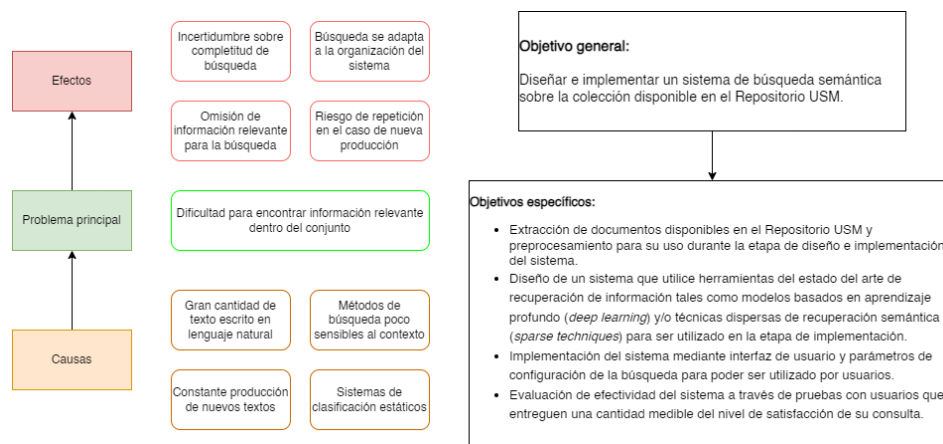


Figura 1: Árbol del problema.

Fuente: Elaboración propia

Como *corpus* de documentos para este sistema, se escoge trabajar sobre la colección disponible en el Repositorio USM, reconociéndola como una fuente de información valiosa para miembros de la comunidad universitaria. El Repositorio USM es un repositorio digital administrado por el Área de Archivo Digital, Dirección de Información y Documentación Bibliográfica Institucional de la UTFSM ². En particular, se considerarán las tesis de alumnos almacenadas en este repositorio.

Así, los usuarios que harán uso de este sistema serán todos aquellos usuarios que tengan interés en las tesis de pregrado y postgrado desarrolladas por alumnos de la universidad. Particularmente, será de interés tanto para alumnos cuyas tesis se encuentren almacenadas dentro del repositorio, como para aquellos alumnos que decidan apoyarse en estas tesis para realizar un trabajo propio.

En la implementación actual del sistema de búsqueda del repositorio, no existe la opción de realizar una búsqueda semántica sobre la colección de documentos disponibles. Creemos que la elaboración de este sistema propuesto puede representar un aporte al sistema actual, ya que puede actuar como un complemento que utiliza un enfoque distinto, y que otorga más opciones de búsqueda al usuario del sistema.

Mencionamos algunos casos de uso particulares para el sistema. Un usuario podría querer realizar una búsqueda cuyo peso principal se encuentre en su significado cuando, por ejemplo, busca información sobre una idea en particular, sin saber exactamente dentro de qué disciplina se podría clasificar (como puede ser el caso de una tesis multidisciplinar). O también en el caso en que conozca bien sobre qué disciplina buscar, pero desconozca la terminología o los conceptos exactos relacionados con ella. Por último, podemos pensar en el caso en que el contenido útil para el usuario puede no necesariamente estar en el título del documento, sino que en su contenido. Estos son algunos de los casos de uso que podrían

²<https://repositorio.usm.cl/>

verse beneficiados con la implementación de un sistema de búsqueda semántica sobre los documentos del repositorio.

Creemos que dar una pronta solución a esta problemática es importante principalmente por dos razones: primero, los sistemas de búsqueda han evolucionado dramáticamente en el último tiempo, de modo que, acostumbrados a sistemas de búsqueda cada vez más sofisticados, usuarios futuros del repositorio podrían sentirse frustrados al momento de utilizar este sistema, pues no responde a los avances actuales en tecnología de recuperación de información. Y segundo, la cantidad de trabajos de tesis producidos por alumnos de la universidad no hará más que aumentar con el tiempo, y será cada vez más difícil hacer un uso eficiente de ellos. El trabajo de esta memoria, entonces, busca contribuir a la visibilidad y usabilidad del trabajo realizado por los estudiantes de la universidad.

1.2. El Repositorio USM

El Repositorio USM, en su implementación actual, tiene disponibles las siguientes colecciones:

- Archivo Histórico USM
- Editorial USM
- Publicaciones de Congresos, Seminarios, Policy Briefs, etc.
- Sistema de Bibliotecas USM

Cada una de estas colecciones se compone, a su vez, de subcolecciones. Como se ha dicho, para el trabajo de esta memoria, nos centraremos en la colección de "Sistema de Bibliotecas USM", y, en particular, en su subcolección "Tesis USM". La elección de esta colección responde a dos razones principales: en primer lugar, es la colección de mayor tamaño del repositorio, lo que la coloca como la colección "principal" del repositorio, además de que así contaremos con una mayor cantidad de documentos para armado del sistema. Y, en segundo lugar, es la colección sobre la que se tiene un mayor interés en trabajar, al tratarse de una colección de tesis de pregrado y postgrado desarrollados por alumnos de distintas carreras de la universidad.

La información disponible en esta colección, para cada documento, corresponde a los metadatos de estos documentos, los cuales incluyen información como título, descripción, autores, etc, además del archivo digital de cada documento en formato PDF, para la gran mayoría de estos documentos. El detalle del procesamiento y análisis de esta información para su uso en el desarrollo del sistema final será expuesto más adelante en este trabajo.

Finalmente, se desea dejar en claro que el objetivo del desarrollo de este sistema no es el de reemplazar a la implementación actual del sistema de búsqueda disponible en el repositorio, sino más bien el de servir como un ejemplo o prueba de concepto para un nuevo enfoque en la búsqueda de información sobre corpus de documentos de gran tamaño, utilizando herramientas del estado del arte en procesamiento de lenguaje natural.

1.3. Objetivos

1.3.1. Objetivo general

- Diseñar e implementar un sistema de búsqueda semántica sobre la colección disponible en el Repositorio USM para mejorar la calidad y precisión de consulta del usuario.

1.3.2. Objetivos específicos

- Extracción de documentos disponibles en el Repositorio USM y preprocesamiento para su uso durante la etapa de diseño e implementación del sistema.
- Diseño de un sistema que utilice herramientas del estado del arte de recuperación de información tales como modelos basados en aprendizaje profundo (deep learning) y/o técnicas dispersas de recuperación semántica (sparse techniques) para ser utilizado en la etapa de implementación.
- Implementación del sistema mediante interfaz de usuario y parámetros de configuración de la búsqueda para poder ser utilizado por usuarios.
- Evaluación de viabilidad del sistema a través de pruebas con usuarios que entreguen una opinión sobre el sistema implementado.

CAPÍTULO 2

MARCO CONCEPTUAL

Este marco conceptual considera, en primer lugar, una explicación sobre el área de recuperación de información (*information retrieval*), área dentro de la cual se enmarca el trabajo de esta memoria.

Luego, se realiza una explicación sobre la búsqueda semántica (*semantic search*), un panorama actual aplicado a las tareas del área de recuperación de información, y que es de interés particular al trabajo desarrollado en esta memoria, por las razones expuestas en el primer capítulo.

Finalmente, se da una explicación de la arquitectura *transformer*, una arquitectura de aprendizaje profundo (*deep learning*) en la cual se basan los modelos de lenguaje de gran tamaño (*large language models*, o LLM) que usaremos en la propuesta e implementación final de la solución de este trabajo. En particular, se exponen los modelos BERT y SBERT (una modificación del modelo BERT).

2.1. Recuperación de Información (*Information Retrieval*)

En esta sección, se expone una visión general del área de recuperación de información, además de una visión de su panorama actual, la cual está basada, principalmente, en el estudio de investigación [Hambarde y Proença, 2023].

El objetivo principal de la recuperación de información moderna es presentar la información más relevante a los usuarios dependiendo de sus consultas. Esta operación involucra, principalmente, dos etapas: la recuperación de la información (*retrieval stage*), y su posterior clasificación (*ranking stage*) (Figura 2). En la etapa de recuperación, se emplean varios modelos y algoritmos, tales como el modelo de espacio vectorial (*vector space model*) [Salton et al., 1975], el modelo booleano (*boolean model*), el indexado semántico latente (*latent semantic indexing*) [Deerwester et al., 1990], la asignación latente de Dirichlet (*Latent Dirichlet Allocation*, o LDA) [Blei et al., 2003], y modelos más recientes como BERT [Kenton y Toutanova, 2019] para recuperar un conjunto inicial de documentos que probablemente sean relevantes para la consulta. Luego, en la etapa de clasificación, se reevalúa y ajusta el orden de estos documentos, según sus puntajes de relevancia.

La recuperación convencional basada en términos (*conventional term-based retrieval*), también conocida como recuperación booleana, es un método tradicional en la recuperación de información [Schütze et al., 2008]. Este enfoque considera la coincidencia directa de términos en una consulta con los términos en un documento, lo cual ofrece simplicidad y velocidad en la operación, pero presenta limitaciones en áreas como el manejo de sinónimos, polise-

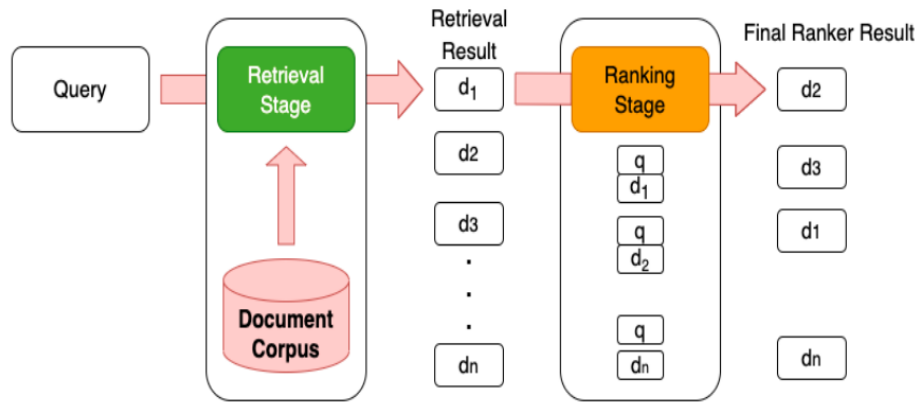


Figura 2: Proceso de recuperación de información.

Fuente: [Hambarde y Proença, 2023]

mia (palabras con múltiples significados) y contexto de la consulta [Baeza-Yates *et al.*, 1999]. A pesar de estos inconvenientes, se utiliza comúnmente en sistemas de recuperación de información a pequeña escala [Schütze *et al.*, 2008].

Los métodos más actuales de recuperación de información incluyen técnicas de aprendizaje profundo, técnicas dispersas de recuperación semántica (*sparse techniques*) y técnicas híbridas, las cuales llevan a cabo una recuperación semántica de la información.

Los enfoques de aprendizaje profundo, basados en redes neuronales tales como redes neuronales convolucionales (*convolutional neural networks*, o CNN), redes neuronales recurrentes (*recurrent neural networks*, o RNN) y la arquitectura *transformer* con mecanismos de atención, permiten representar documentos y consultas como vectores continuos, facilitando el uso de medidas de similitud basadas en redes neuronales para clasificar documentos en función de su relevancia para una consulta [Shen *et al.*, 2014, Huang *et al.*, 2013].

Los métodos de recuperación dispersa se refieren a técnicas utilizadas en la recuperación de información que tienen como objetivo reducir la dimensionalidad de los datos que se están indexando y recuperar sólo los documentos más relevantes. Estos métodos se basan en la idea de que la mayoría de los datos en una colección no son relevantes para una consulta dada, y que no es necesario recuperar todos los documentos cuando solo un pequeño subconjunto es relevante.

En cuanto a la etapa de clasificación, los métodos basados en modelos de aprendizaje profundo han ganado, nuevamente, especial relevancia. El aprendizaje profundo ofrece ventajas para capturar relaciones complejas entre documentos y consultas [Mitra *et al.*, 2017]. El modelo de espacio vectorial (*vector space model*, o VSM) [Salton *et al.*, 1975] es ampliamente utilizado en la recuperación de información, representando documentos y consultas como vectores para clasificarlos en base a similitud.

2.2. Búsqueda Semántica (*Semantic Search*)

En esta sección, se dará una visión general de la búsqueda semántica, enmarcada dentro del contexto de la recuperación de información. Esta sección estará basada principalmente en el artículo *Semantic search with embeddings: index anything*, de Romain Beaumont, escrito en Diciembre de 2020 ³.

Un sistema de búsqueda semántica se compone, principalmente, de dos etapas: una etapa de codificación (*encoding pipeline*) en la cual se obtienen representaciones en vector de la información, e índices para almacenar estos vectores, y una etapa de búsqueda (*search pipeline*) en la cual se obtiene una representación en vector de la consulta, y se utilizan los índices creados anteriormente para retornar información relevante para la consulta (Figura 3).

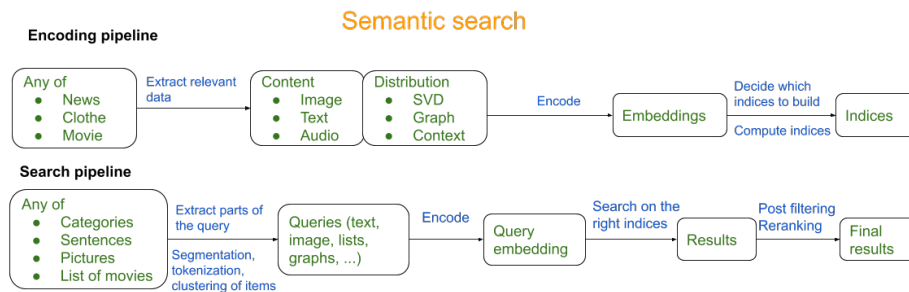


Figura 3: Etapas de la búsqueda semántica.

Fuente: *Semantic search with embeddings: index anything*⁴

2.2.1. Secuencia de Codificación (*Encoding pipeline*)

El primer paso para la construcción de un sistema de recuperación de información basado en búsqueda semántica es la codificación de la información en representaciones en vector. Es posible realizar esta codificación para un gran conjunto de información, de los cuales luego se construirán índices que nos servirán para buscar entre la información.

Los sistemas de recuperación pueden codificar información de distinto tipo, como pueden ser archivos de imagen, audio o texto. Esta información es codificada en representaciones en vector llamadas *embeddings*.

En el caso de texto, las técnicas de aprendizaje profundo modernas son capaces de construir representaciones poderosas para estos objetos. Algunas de estas técnicas son:

- **Word2vec:** Una técnica para obtener la representación en vector de palabras. Estas

³<https://rom1504.medium.com/semantic-search-with-embeddings-index-anything-8fb18556443c>

representaciones capturan información sobre el significado de la palabra basado en palabras cercanas.

- **Transformer:** Una arquitectura de aprendizaje profundo reciente que utiliza mecanismos de atención para obtener representaciones en vector de palabras. Recientemente, se ha convertido en la técnica del estado del arte para multitud de tareas en procesamiento de lenguaje natural (*natural language processing*, o NLP).
- **BERT:** Un modelo de lenguaje de gran tamaño basado en la arquitectura *transformer*, pre-entrenado en una gran cantidad de texto. Puede ser utilizado para una multitud de tareas de procesamiento de lenguaje natural. Entre ellas, la obtención de representaciones en vector de palabras, oraciones y párrafos

Una vez que se obtienen las representaciones en vector de la información, se necesita una manera de revisarlas rápidamente. Esto puede lograrse utilizando al algoritmo de k-vecinos más próximos (*k-nearest neighbors*, o KNN). Básicamente, consiste en el cálculo de la distancia entre un vector y todos los vectores del set de datos. Esta operación puede ser mejorada utilizando el algoritmo de k-vecinos más próximos aproximados (*approximate k-nearest neighbors*, o AKNN).

2.2.2. Secuencia de Búsqueda (*Search pipeline*)

La secuencia de búsqueda es la parte del sistema con la que usualmente interactúa el usuario. Tiene como objetivo recuperar los resultados más relevantes para una consulta dada. Es importante que sea capaz de retornar resultados dentro de un corto periodo de tiempo, para que el sistema pueda ser utilizado de manera eficiente.

Esta secuencia está compuesta por un componente que extrae información relevante de la consulta, un codificador que transforma esta información en representaciones en vector, un sistema de búsqueda que utilice los índices construidos durante la secuencia de codificación y, finalmente, un sistema de post-filtrado que seleccione los mejores resultados de la búsqueda.

La primera parte del sistema consiste en considerar una consulta como entrada y extraer de ella información relevante para poder ser codificada en una representación en vector. Para una buena codificación, se pueden utilizar varias técnicas. En el caso particular del texto, pueden utilizarse las técnicas que se han listado anteriormente en la secuencia de codificación. Generalmente se utiliza la misma técnica tanto para la secuencia de codificación, como para la secuencia de búsqueda.

Dependiendo del tipo de consulta, puede ser relevante construir no solo un único índice, sino que varios. Por ejemplo, si la consulta tiene un filtro para una categoría dada de objeto, podría ser relevante construir un índice específico para este subconjunto de representaciones en vector.

2.3. La Arquitectura *Transformer*

La arquitectura *transformer* es una arquitectura de aprendizaje profundo basada en el mecanismo de atención de múltiples cabezas (*multi-head attention mechanism*). Fue propuesta en [Vaswani et al., 2017]. Es la pieza central para comprender el funcionamiento de los modelos de lenguaje de gran tamaño del estado del arte. A continuación, se da una explicación de esta arquitectura a nivel general, la cual está basada principalmente y hace uso de las ilustraciones presentadas en el artículo *The Illustrated Transformer*, de Jay Alammar, escrito en Junio de 2018 ⁵.

Una vista general de la arquitectura puede observarse en la Figura 4. Explicaremos su funcionamiento primero a nivel general, y luego nos centraremos en el detalle de sus partes. La Figura 5 muestra una abstracción de esta arquitectura, para el ejemplo de una aplicación de traducción de una oración. Como puede observarse en este ejemplo, la entrada corresponde a la oración en francés "Je suis étudiant", y la salida, a su traducción al inglés, "I am a student". Así, observamos dos puntos principales: el *transformer* posee una noción de "comprensión" del lenguaje, y puede utilizar esta noción para resolver tareas que requieren de esta "comprensión" del lenguaje, como es el caso de la traducción de una oración. ¿Cómo logra esto? Lo revisamos a continuación.

⁵<https://jalammar.github.io/illustrated-transformer/>

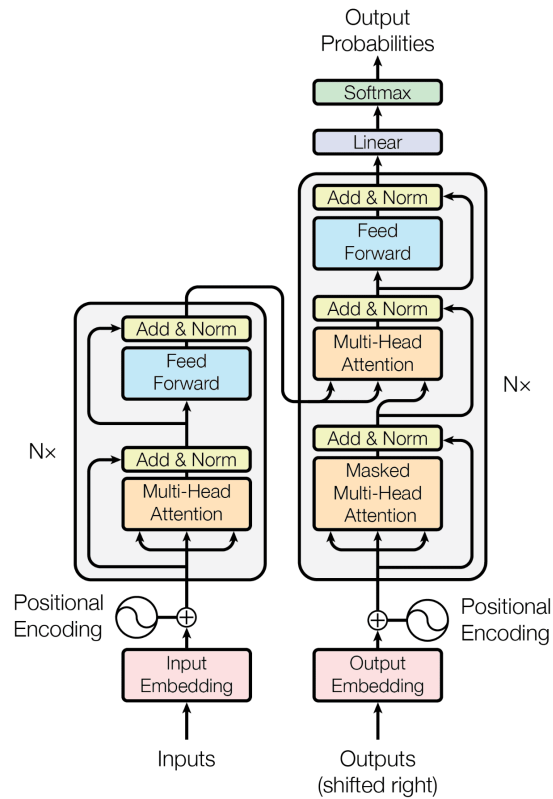


Figura 4: La arquitectura *transformer*.
Fuente: [Vaswani et al., 2017]



Figura 5: Abstracción de la arquitectura *transformer*.
Fuente: *The Illustrated Transformer*

La Figura 6 muestra el *transformer* con un poco más de detalle. Observamos que la arquitectura está compuesta principalmente de dos componentes: los *encoders* y los *decoders*. En la Figura 4, estos componentes están representados por los rectángulos marcados con Nx (junto con todas las capas en su interior, como la capa de *Multi-Head Attention* y la capa de *Feed Forward*). El rectángulo de la izquierda corresponde a los *encoders*, y el de la derecha, a los *decoders*.

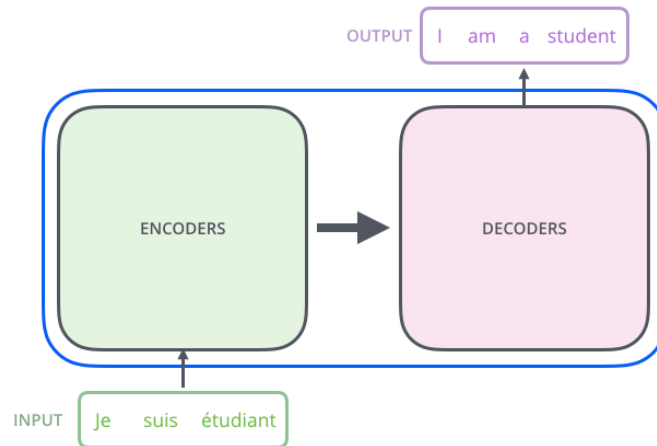


Figura 6: Abstracción de los componentes principales de la arquitectura *transformer*.
Fuente: *The Illustrated Transformer*

En la Figura 7 se muestra la arquitectura con aún más detalle. Vemos que el componente de *encoders* es, en realidad, una pila compuesta por varios componentes *encoder* individuales, como el que aparece en la Figura 4. $N \times$, en esta figura, se refiere a "el componente N una cantidad x de veces". En esta figura, así como también en la Figura 7, la cantidad de componentes *encoder* de la pila es igual a seis, y la cantidad de componentes *decoder* de la otra pila, también seis. Pasemos, ahora, a revisar el *encoder* en detalle.

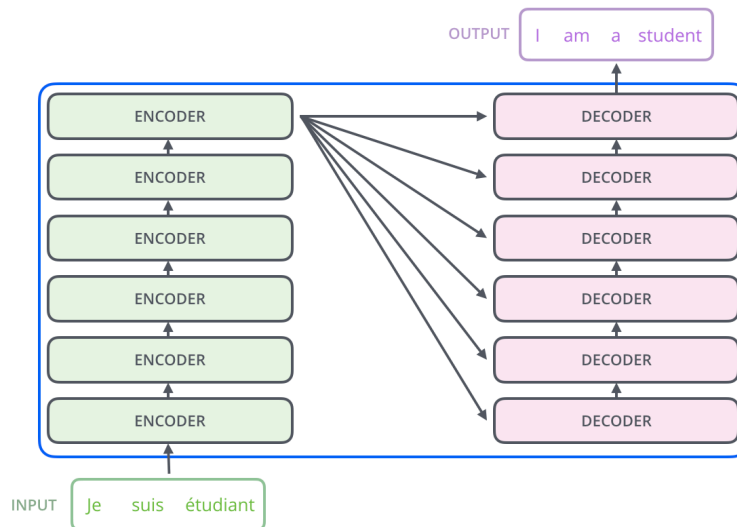


Figura 7: Detalle de los componentes principales de la arquitectura *transformer*.
Fuente: *The Illustrated Transformer*

2.3.1. El *Encoder*

Todos los encoders tienen la misma estructura, que puede observarse en la Figura 8. Está compuesto de dos sub-capas: una capa de *Self-Attention*, y una capa de *Feed Forward Neural Network*. La capa de *Self-Attention* ayuda al *encoder* a mirar otros *tokens* de la entrada mientras obtiene la codificación de un *token* en específico. Explicaremos el funcionamiento de este mecanismo en detalle más adelante. La capa de *Feed Forward Neural Network* es una red neuronal simple que ayuda a transmitir la información desde un *encoder* hasta otro *encoder* en la pila. No entraremos en los detalles del funcionamiento de esta capa. El *encoder* recibe *tokens* de entrada, y estos suben desde un *encoder* a otro, como se aprecia en la Figura 9, hasta llegar a la codificación final de cada *token*. En este ejemplo, se tienen dos *tokens* de entrada (que equivalen a palabras, en este caso), "Thinking" y "Machines". Revisemos ahora la capa de *Self-Attention* en detalle.

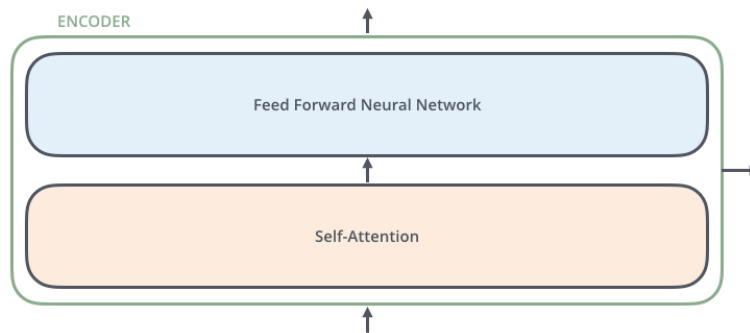


Figura 8: Detalle del componente *encoder* del *transformer*.
Fuente: *The Illustrated Transformer*

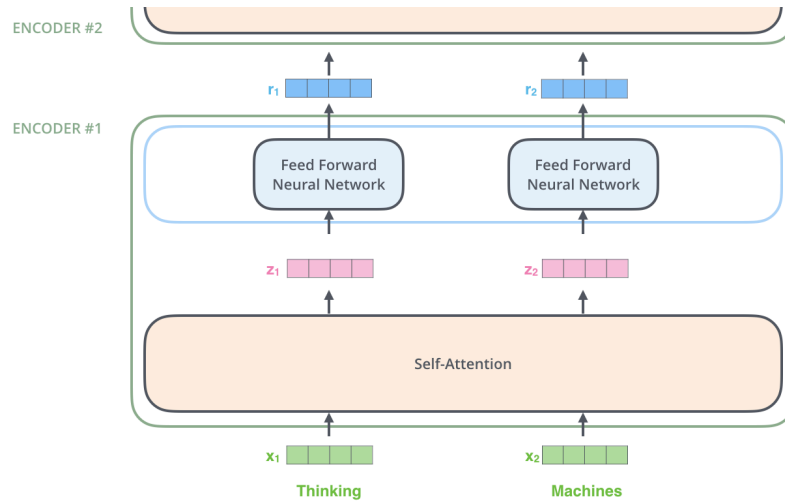


Figura 9: Movimiento de los tokens de entrada a través de los encoders.

Fuente: *The Illustrated Transformer*

La capa de *Self-Attention* le ayuda al *encoder* a ver otros tokens mientras está obteniendo la codificación de un token en particular, con el fin de entender su relación con cada uno de los otros tokens que componen la entrada, y codificar esta relación en la representación en vector final del token. Pensemos, por ejemplo, que queremos traducir la siguiente oración: *The animal didn't cross the street because it was too tired*. A qué se refiere la palabra *it* en esta oración? Acaso se refiere a *animal*, o a *street*? Cuál es su relación con cada una de las otras palabras que componen la oración? La Figura 10 muestra una ilustración del proceso realizado por el mecanismo de *Self-Attention*. En este ejemplo, cuando intentamos obtener la codificación de la palabra *if*, el mecanismo de *Self-Attention* le presta más atención a las palabras *The* y *animal*. Pero, ¿Cómo determina a qué palabras prestarle atención, y cómo dar una métrica para esta atención? Vamos a verlo en detalle.

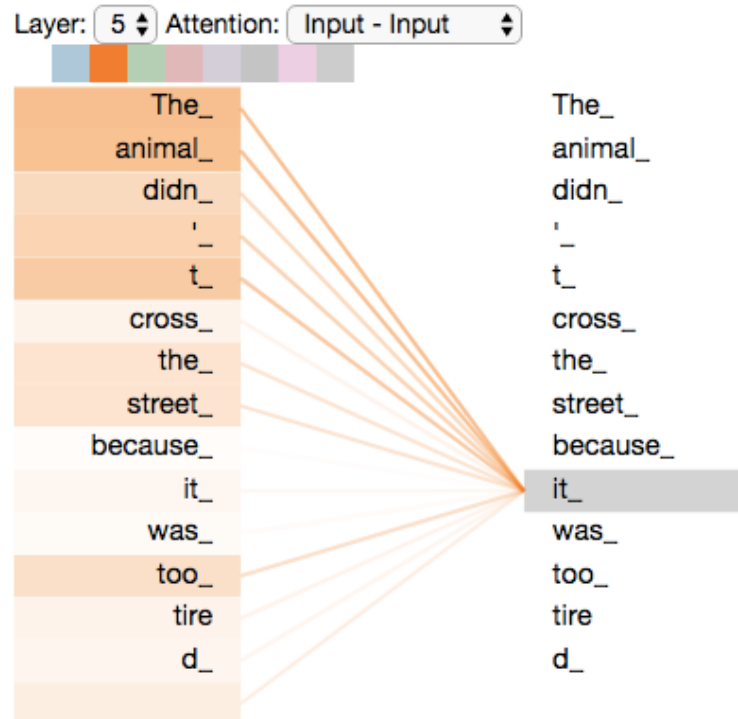


Figura 10: Mecanismo de *Self-Attention* considerando la relación de la palabra "it" con todas las otras palabras que componen la oración, durante su codificación en vector.

Fuente: *The Illustrated Transformer*

El primer paso en el cálculo realizado por el mecanismo de *Self-Attention* es crear tres vectores a partir de cada vector de entrada del *encoder* (en este caso, una representación en vector de cada *token*). En la Figura 4, la capa de nombre *Input Embedding* es la encargada de obtener esta codificación para cada *token* de la entrada. Creamos estos vectores multiplicando cada vector de entrada por tres matrices: W_Q , W_K y W_V , que corresponden a matrices que fueron obtenidas durante la fase de entrenamiento. Con ello, obtenemos tres vectores para cada token x de entrada: un vector q (*queries*), un vector k (*keys*), y un vector v (*values*). La Figura 11 muestra estos vectores y estas matrices.

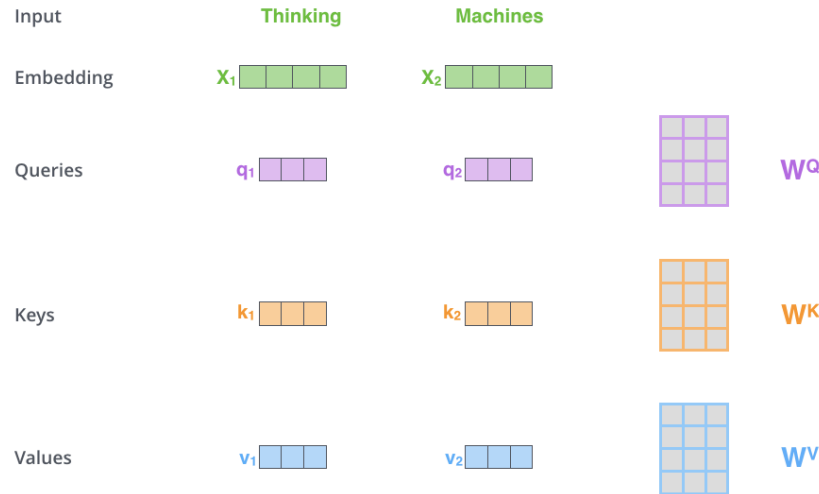


Figura 11: Vectores de los *tokens* de entrada del *encoder*, y sus proyecciones en vectores q , k y v , obtenidos luego de multiplicar cada vector por las matrices W_Q , W_K y W_V .

Fuente: *The Illustrated Transformer*

El segundo paso es el cálculo de un puntaje, que corresponde al puntaje de cada *token* de la secuencia de entrada en comparación con el *token* que estamos codificando actualmente. Este puntaje nos indica cuanta "atención" prestarle a cada *token* de la secuencia de entrada cuando estamos codificando un *token* en una posición específica. Calculamos este puntaje multiplicando el vector q del *token* por cada vector k de cada *token*.

El tercer paso es dividir los puntajes por la raíz cuadrada de la dimensión de los vectores k . Esto nos da gradientes más estables.

El cuarto paso es pasar este resultado por una operación de *softmax*, que normalizará los puntajes para que sean positivos y que para que sumen 1 en total.

El quinto paso es multiplicar cada vector v de cada palabra por el puntaje *softmax* obtenido.

El sexto paso es sumar todos los vectores v con pesos obtenidos en el paso anterior, y esta será la salida de la capa de *Self-Attention* para cada *token*.

La Figura 12 muestra en detalle el resultado de cada una de estas operaciones, para el ejemplo donde nuestros *tokens* de entrada son las palabras "Thinking" y "Machines".

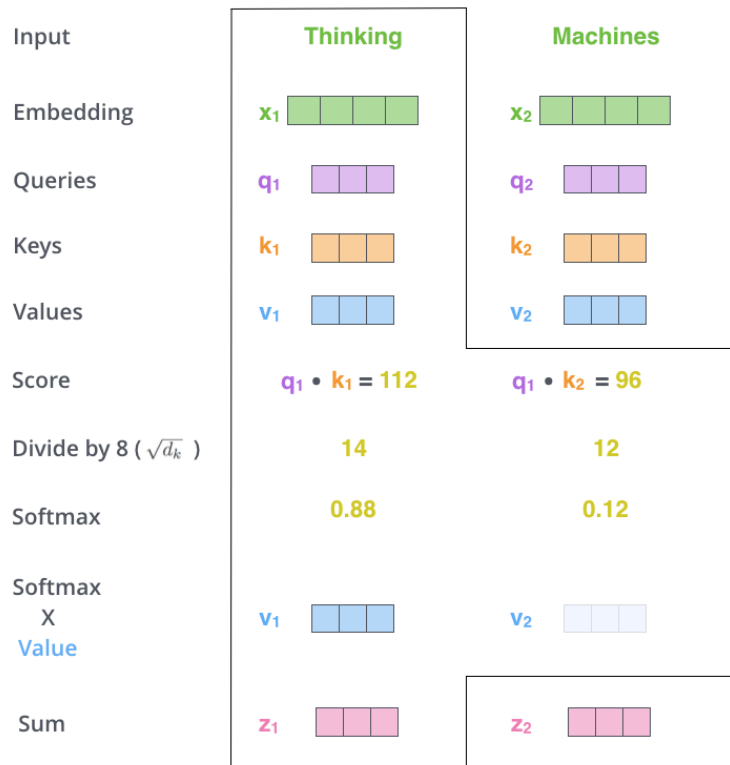


Figura 12: Resultado de las operaciones realizadas por la capa de *Self-Attention*

Fuente: *The Illustrated Transformer*

En la práctica, el *transformer* no trabaja con vectores, sino que con matrices. Debido a esto, se dice que la capa de *Self-Attention* trabaja con múltiples "cabezas" de atención para obtener la representación en vector final de cada *token*. Es decir, hacemos lo que hemos descrito, pero de forma simultánea, en varias capas. Esto aumenta la capacidad de representación del mecanismo. La Figura 13 muestra una vista general de la operación realizada por la capa de *Self-Attention*, considerando cálculo con matrices y múltiples cabezas de atención. Notamos que es una especie de escalamiento del proceso que hemos descrito para vectores individuales.

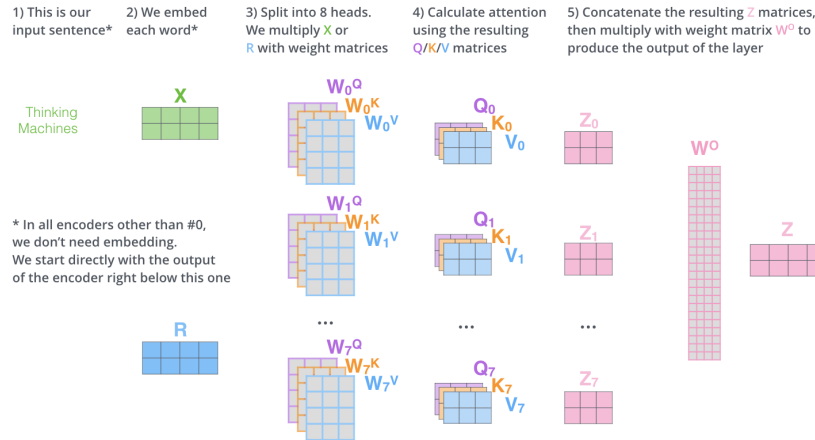


Figura 13: Cálculo realizado por el mecanismo de *Multi-Head Attention*, utilizando matrices
Fuente: *The Illustrated Transformer*

La Figura 14 muestra una representación del resultado de múltiples cabezas de atención, que ilustra las diferentes "atenciones" que presta el modelo en cada cabeza a cada *token* mientras considera su relación con todos los *tokens* que componen la entrada.

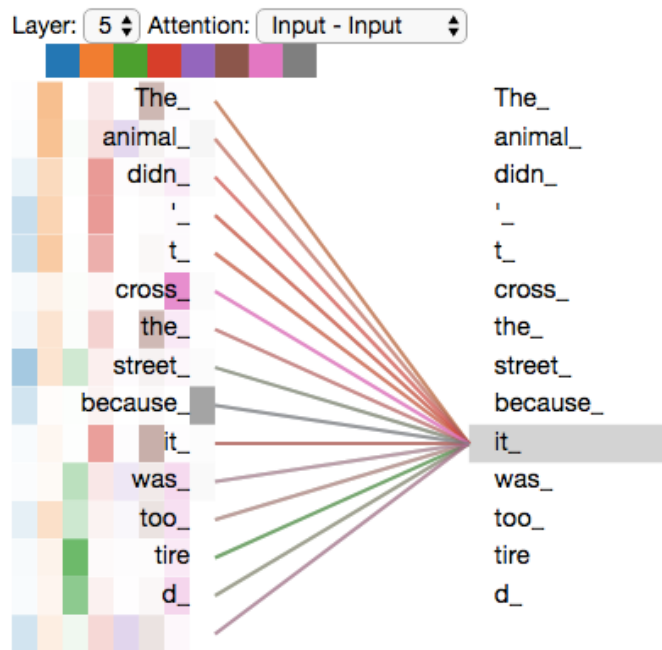


Figura 14: Resultado del mecanismo de atención de múltiples cabezas para el token "it" en el ejemplo de entrada de una oración en inglés
Fuente: *The Illustrated Transformer*

Ahora, de la Figura 4, nos queda explicar la capa con el nombre de *Positional Encoding*. Esta

capa entrega una especie de noción de posición de cada *token* dentro de la entrada, puesto que esta noción no es obvia para la codificación inicial en vector de cada *token*. La Figura 15 ilustra esta operación. No entraremos en mayor detalle sobre su funcionamiento, nos basta quedarnos con esta noción.

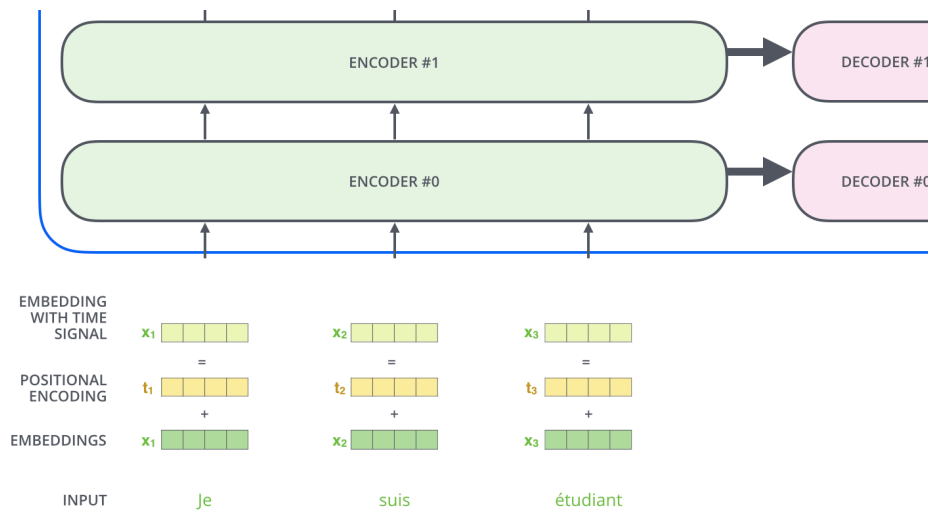


Figura 15: Para cada codificación inicial de cada *token* de entrada del *encoder*, la capa de *Positional Encoding* agrega una noción de posición a cada codificación.

Fuente: *The Illustrated Transformer*

Por último, debemos fijarnos también que en el modelo del inicial de la arquitectura *transformer*, aparece una capa con el nombre de *Add & Normalize* por encima de las capas de *Multi-Head Attention* y *Feed Forward*. Podemos visualizar la operación realizada por esta capa en la Figura 16. De nuevo, no entraremos en mayor detalle sobre esto.

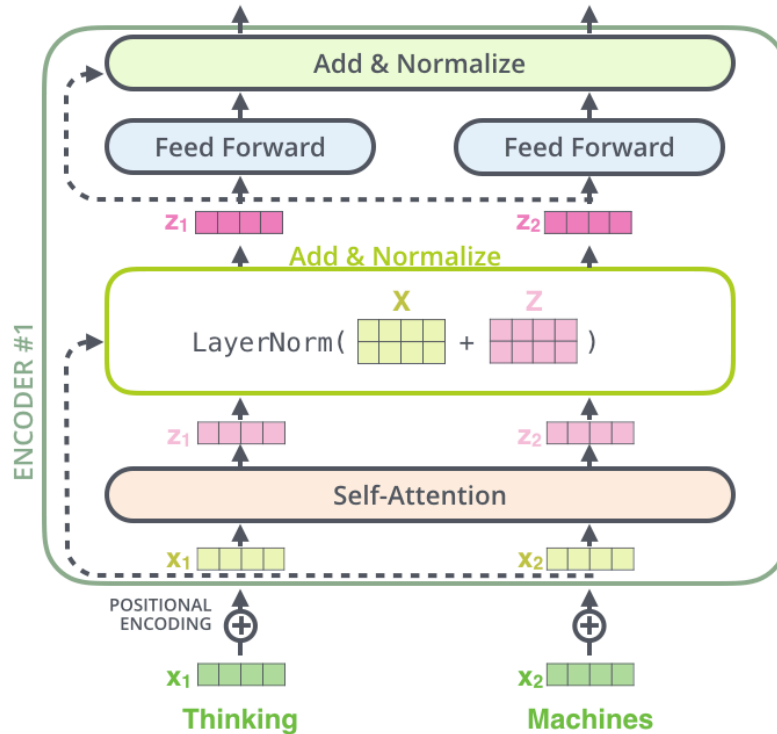


Figura 16: Operación de adición y normalización realizada por la capa *Add & Normalize*.

Fuente: *The Illustrated Transformer*

Y este es el detalle del funcionamiento del *encoder*. Hemos visto lo principal de su funcionamiento, cuáles son sus entradas, y cuáles son sus salidas. Pasamos, ahora, a revisar el segundo componente principal de la arquitectura *transformer*: el *decoder*.

2.3.2. El Decoder

Habiendo visto el funcionamiento general del *encoder* y sus componentes, conocemos también el funcionamiento básico del *decoder* y sus componentes. Revisemos, a continuación, como funcionan todos juntos. La Figura 17 muestra una ilustración de la arquitectura *transformer* como la que revisamos al comienzo, considerando los componentes que hemos revisado hasta ahora. En particular, hacemos notar la relación entre la salida del *encoder*, y la salida del *decoder*.

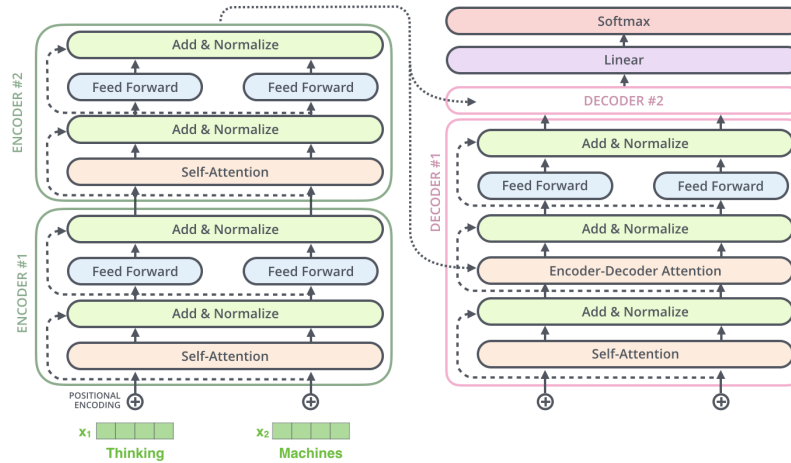


Figura 17: Vista general de la arquitectura *transformer*.

Fuente: *The Illustrated Transformer*

La salida final de la lista de *encoders*, las representaciones en vector de cada *token* dentro de la secuencia de entrada, considerando su relación con cada *token* de la secuencia, se transforma en el set de vectores de atención K_{encdec} y V_{encdec} . Estos vectores son usados por cada *decoder* en su capa de *Encoder-Decoder Attention*, esto con el objetivo de ayudar al *decoder* a prestar atención a las partes apropiadas de la secuencia de entrada. El ultimo *decoder* de la pila de *decoders* entrega como salida una palabra. Este proceso puede visualizarse en la Figura 18.

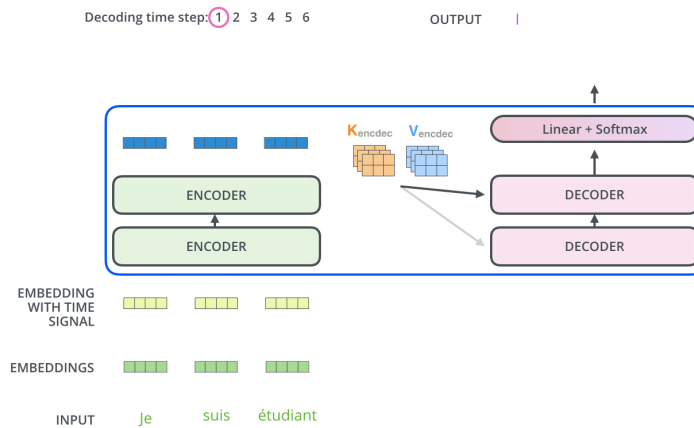


Figura 18: La salida de la pila de *encoders* es convertida en un set de vectores que son pasados a cada *decoder* en la pila de *decoders*.

Fuente: *The Illustrated Transformer*

La salida del *decoder* en cada iteración es entregada de vuelta de manera recurrente al primer *decoder* al comienzo de la pila, y este proceso continúa hasta que el *decoder* entrega un símbolo especial que indica el fin de la salida. Esto puede visualizarse en la Figura 19.

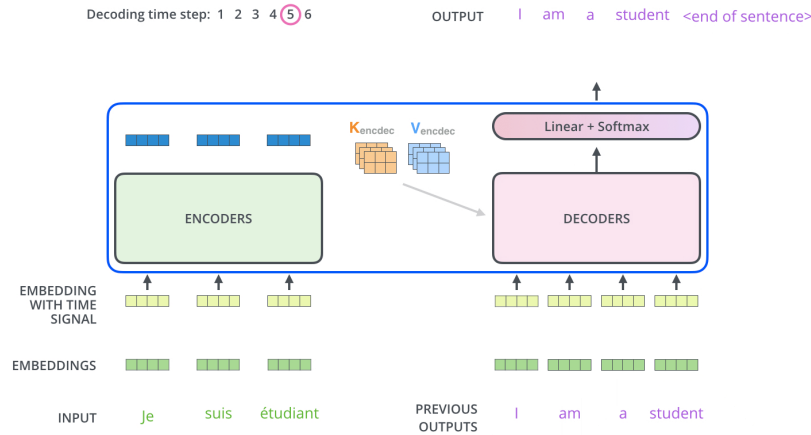


Figura 19: Este proceso continúa hasta que el *decoder* entrega un símbolo especial que indica el fin de la salida.

Fuente: *The Illustrated Transformer*

Este es, básicamente, el funcionamiento del *decoder*. Debemos mencionar una diferencia fundamental del mecanismo de *Self-Attention* con respecto al *encoder*, y esta es: el mecanismo de *Self-Attention* del *encoder* solo puede observar los *tokens* anteriores en la secuencia de salida.

Por último, debemos mencionar el papel que cumplen las capas *Linear* y *Softmax* en la arquitectura del *decoder*. Estas capas se encuentran por encima de la pila de *decoders*, y se encargan de transformar la salida de esta pila, que corresponde a un vector de números decimales, a una palabra, o símbolo reconocible en un lenguaje. No hace falta entrar en el detalle de esto, nos basta con mencionarlo.

2.4. BERT

BERT es un modelo de lenguaje de gran tamaño basado en la arquitectura *transformer*, el cual ha sido pre-entrenado en un set de datos muy grande, de modo que posee una buena capacidad para "comprender" el lenguaje. Fue propuesto por [Kenton y Toutanova, 2019]. A continuación, se da una explicación de BERT a nivel general, la cual está basada principalmente y hace uso de las ilustraciones presentadas en el artículo *The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning)*, de Jay Alammar, escrito en Diciembre de 2018⁶.

La Figura 20 muestra un ejemplo donde se hace uso de BERT para resolver una tarea de clasificación de correo. Debido a esta buena capacidad para "comprender" el lenguaje, BERT es ampliamente utilizado en diversas tareas de procesamiento de lenguaje natural, como

⁶<https://jalammar.github.io/illustrated-bert/>

la que se muestra en el ejemplo. Así, podemos usar BERT como una base para obtener un entendimiento general del lenguaje, y luego utilizar este entendimiento para resolver nuestra tarea en particular. Primero revisaremos una visión general de BERT, y luego veremos su uso en una tarea específica, cuando revisemos el modelo SBERT.

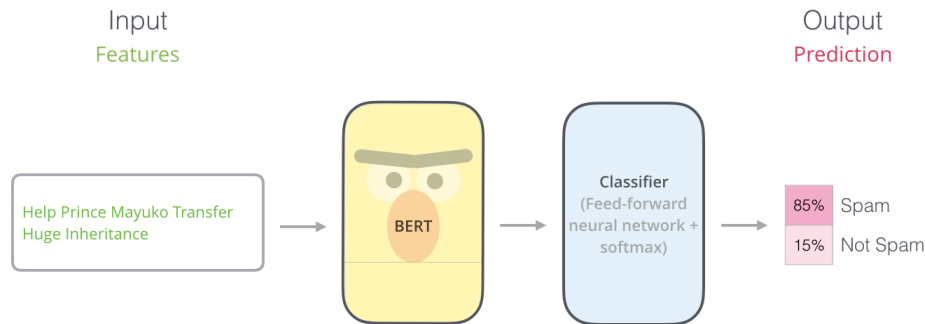


Figura 20: Ejemplo de uso de BERT para la tarea de clasificación de correos *spam*.
Fuente: *The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning)*

Revisemos la arquitectura de BERT. La Figura 21 muestra una visión general. BERT es una pila de componentes *encoder* entrenados. La figura muestra dos versiones: BERT base, con 12 componentes *encoder*, y BERT large, con 24 componentes *encoder*. Estos encoders, a su vez, tienen también una mayor cantidad de cabezas de atención en comparación al *transformer* original.

Así, BERT se construye utilizando exclusivamente la pila de *encoders* que compone la arquitectura *transformer*, excluyendo la segunda mitad, la pila de *decoders*. La razón de esto, es que los desarrolladores de BERT observaron que utilizar esta pila era suficiente para construir un modelo de lenguaje, que luego fuera posible utilizar para diferentes tareas.

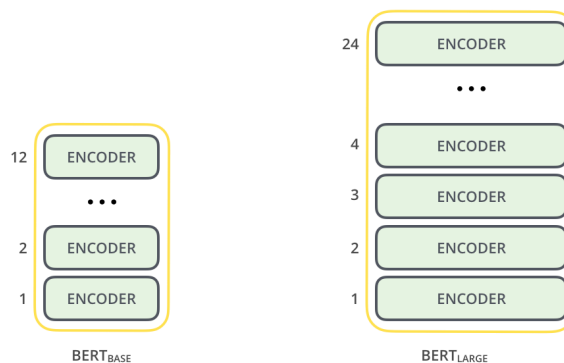


Figura 21: Vista general de la arquitectura de BERT, compuesto exclusivamente por una pila de *encoders*

Fuente: *The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning)*

La Figura 22 muestra un ejemplo del procesamiento de una entrada en BERT. En este caso, se trata de una tarea de clasificación, que tiene como primer *token* [CLS], que quiere decir clasificación. Como hemos visto, el *encoder* tendrá como salida una representación en vector de cada *token* de entrada considerando su relación con el resto de *tokens*. En este caso, la representación del *token* [CLS] codifica una noción de esta tarea de clasificación, de modo que esto será lo que pasaremos al clasificador, y será este clasificador el que entrenaremos para realizar la tarea de clasificación, como puede verse en la Figura 23. Esta es una manera en la que podemos utilizar la salida de BERT para resolver diferentes tareas. Podemos utilizar distintos *tokens*, en distintas posiciones, para resolver distintas tareas, según modelemos las entradas que entregemos a BERT.

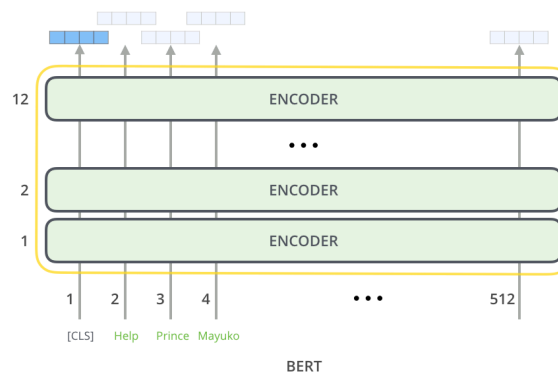


Figura 22: Procesamiento de una entrada en BERT, modelada para la resolución de un problema de clasificación de texto.

Fuente: *The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning)*

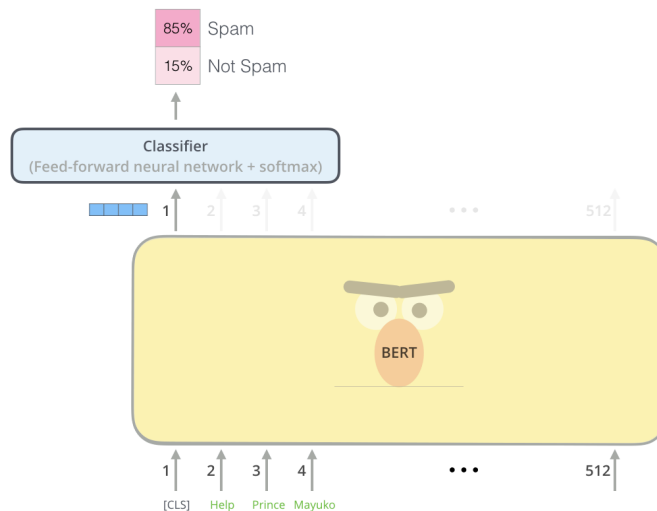


Figura 23: Usamos la representación del token [CLS] para entrenar al clasificador.

Fuente: *The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning)*

Ahora, revisaremos como se realiza el entrenamiento de BERT. Para ello, se realizan dos tareas principales: *Masked Language Model* y *Two-sentence Tasks*.

La Figura 24 ilustra la tarea de *Masked Language Model*. En ella, se enmascaran el 15 % de los *tokens* de entrada, y se pide al modelo que prediga la palabra que falta.

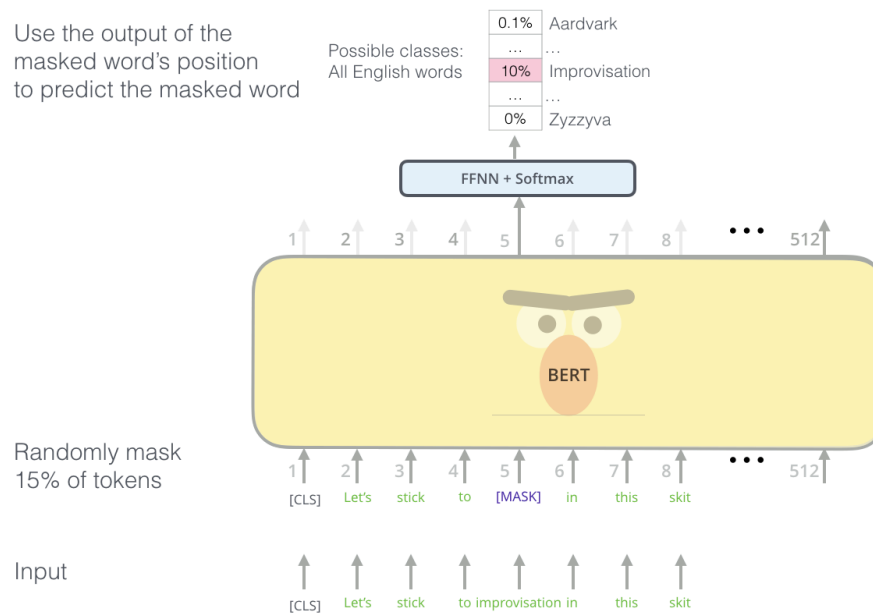


Figura 24: Entrenamiento de BERT con la tarea de *Masked Language Model*
Fuente: *The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning)*

La Figura 25 ilustra la tarea de *Two-sentence Tasks*. Para lograr que BERT pueda manejar bien relaciones entre múltiples oraciones, esta tarea pregunta: dadas dos oraciones, A y B, ¿Acaso B es la oración que sigue a A?

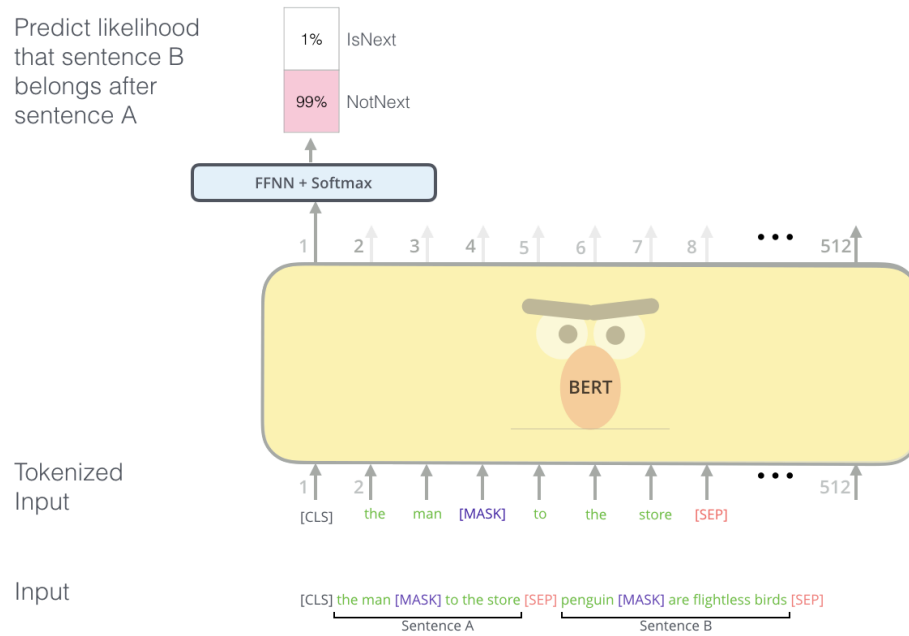


Figura 25: Entrenamiento de BERT con la tarea de *Two-sentence Tasks*
 Fuente: *The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning)*

En otras palabras, estamos entrenando a este modelo de lenguaje realizando tareas como "completar la oración" o "están relacionadas estas oraciones o no". Cada vez que resuelva estas tareas con una puntuación cada vez mayor, entenderemos que es cada vez más hábil para "comprender" el lenguaje. Sin embargo, esto lo podemos hacer para lenguajes cada vez más específicos: es aquí donde observamos la gran capacidad de BERT para ser utilizado en una gran cantidad de distintas tareas de procesamiento de lenguaje natural.

2.5. SBERT

SBERT es una modificación (o *fine-tuning*) de BERT que utiliza redes siameses y redes triples para entregar una representación en vector no solo de *tokens* individuales, sino que de grupos de *tokens*, que, en el caso de palabras, podemos considerar como oraciones o párrafos completos, y que pueden ser comparados utilizando operaciones entre vectores, como la similitud coseno (*cosine similarity*) [Reimers y Gurevych, 2019].

La Figura 26 muestra la arquitectura básica de SBERT. La arquitectura añade una operación de *pooling* a la salida de la entrada procesada por BERT, para obtener una representación en vector única a partir de las representaciones individuales de cada *token* de entrada. SBERT utiliza diferentes estrategias para realizar esta operación de *pooling*. La operación por defecto corresponde al cálculo de la media de todos los vectores entregados por BERT.

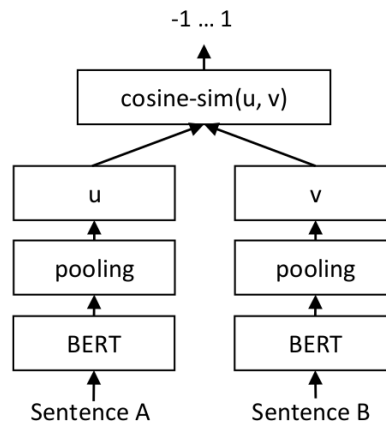


Figura 26: Arquitectura de SBERT
Fuente: [Reimers y Gurevych, 2019]

Una vez obtenidas estas representaciones, se entrena al modelo para que las representaciones que produzca sean buenas representaciones de oraciones o párrafos en el lenguaje. Para este entrenamiento, también es posible utilizar diferentes estrategias. La estrategia que se muestra en la figura hace uso de similitud coseno (*cosine similarity*) entre vectores. Le entregamos al modelo la representación de dos sentencias, y calculamos su similitud coseno, para comparar el valor de esta similitud con un "valor oro", que será alto en el caso de una alta similitud, y bajo en el caso contrario. De esta forma, obtendremos buenas representaciones a nivel de sentencias.

SentenceTransformers es un *framework* (librería) de Python desarrollado por los autores de [Reimers y Gurevych, 2019], pensado para tareas de representación en vector de texto e imágenes⁷. Este *framework* incluye varias funciones que hacen uso de las representaciones entregadas por este modelo. De modo que resulta de gran utilidad, en particular, para la construcción de sistemas basados en búsqueda semántica.

⁷<https://sbert.net/>

CAPÍTULO 3

PROPUESTA DE SOLUCIÓN

3.1. Descripción de la propuesta

Una vez considerada la definición del problema, se procede, ahora, a exponer la propuesta de solución. A grandes rasgos, la propuesta es la siguiente: uso de herramientas basadas en modelos de lenguaje de gran tamaño (LLM) para el manejo de fuentes de información de gran tamaño; en particular, para los documentos alojados en el Repositorio USM como fuente de información. O, dicho de otra manera: ¿Cómo podemos utilizar herramientas que poseen una "noción" de entendimiento del lenguaje para asistir en el manejo de los documentos alojados en el Repositorio USM?

La propuesta es la siguiente: la construcción de un sistema de recuperación de información, que haga uso de una estrategia de búsqueda semántica, y esta búsqueda utilice, en particular, el modelo BERT, y su modificación SBERT, para sus secuencias de codificación y de búsqueda. En las secciones que siguen describiremos, en detalle, la lógica detrás de este sistema y su construcción.

3.2. Motivación para el uso de LLM

La motivación para el uso de modelos de lenguaje de gran tamaño (LLM) en la construcción del sistema propuesto está basada, principalmente, en los buenos resultados que estos modelos han tenido en el último tiempo en su aplicación en diversas tareas de procesamiento de lenguaje natural (NLP) [Naveed *et al.*, 2023], además de una disponibilidad y accesibilidad cada vez mayor a estos modelos gracias a comunidades como Hugging Face⁸.

De los LLM disponibles en la actualidad, se ha escogido uno en particular: el modelo BERT. Este modelo está basado en la arquitectura *transformer*, pero con un enfoque y construcción particular que lo distingue de otros modelos basados en esta arquitectura, como el modelo GPT. La Figura 27 muestra una comparación entre los modelos BERT, GPT, y BART, tres de los modelos más populares en la actualidad para tareas de procesamiento de lenguaje natural.

En esta comparación, observamos que BERT es utilizado principalmente en tareas como análisis de sentimiento (*sentiment analysis*) y reconocimiento de entidades nombradas (*named entity recognition*). Así, adquirimos una buena noción del uso que podemos dar a BERT: tareas de análisis de texto.

En la comparación también observamos la manera de realizar un ajuste fino (*fine-tuning*)

⁸<https://huggingface.co/>

	BERT	GPT	BART
Model Type	Encoder Only	Decoder Only	Encoder-Decoder
Direction	Bidirectional	Unidirectional (left-to-right)	Bidirectional
Pre-training Objective	Masked language modeling (MLM)	Autoregressive (casual) language modeling	Span Corruption (Masking entire spans of words)
Fine-tuning	Task-specific layer added on top of the pre-trained BERT model	Providing task-specific prompts using few-shot or one-shot adaptation and adapting the model's parameters	Versatile and can be used for various NLP tasks
Use Case	Sentiment Analysis Named entity Recognition Word Classification	Text generation Text completion creative writing	Translation Text Summarisation Question & Answer
Original Organisations	Google AI	OpenAI	Facebook AI

Figura 27: Comparación entre los modelos BERT, GPT y BART

Fuente: *BERT, GPT and BART: a short comparison*⁹

a cada modelo. En el caso de BERT, añadimos una capa con una tarea específica por encima de la capa que contiene al modelo pre-entrenado de BERT. Esto es lo que realizaremos en nuestro caso, haciendo uso del modelo ajustado SBERT (*Siamese BERT-Networks*) para la construcción de nuestro sistema. Entraremos en detalle sobre esto en las secciones que siguen.

Esta es la motivación para el uso de LLM; en particular, para el uso del modelo BERT, y la justificación de su uso en la tarea que hemos asignado, esto es, para la codificación de sentencias y párrafos de documentos en representaciones vectoriales. Procedemos, ahora, a explicar en detalle el uso que daremos a este modelo en la construcción del sistema.

3.3. BERT: Uso

Como hemos visto, BERT es usado ampliamente para tareas de análisis de texto. Esto se debe a que BERT es capaz de obtener una buena representación en vector para las palabras (o *tokens*) en el contexto de su oración (o cadena). Sin embargo, ¿Qué tal si quisiéramos obtener una buena representación para oraciones completas, o incluso para párrafos completos? el modelo estándar de BERT, por si solo, no está entrenado para entregar muy buenas representaciones para estos casos. Es por esta razón que [Reimers y Gurevych, 2019] propone SBERT (*Sentence Embeddings using Siamese BERT-Networks*): una modificación del modelo BERT que utiliza redes siamesas para obtener una representación en vector de oraciones y párrafos completos.

El interés principal de tener buenas representaciones en vector para oraciones y párrafos, es la posibilidad de poder compararlos utilizando métricas vectoriales, como puede ser, por ejemplo, la similitud coseno (*cosine similarity*) entre vectores.

Los autores de [Reimers y Gurevych, 2019] han desarrollado también una librería en Python donde se entrega una implementación práctica del modelo, además de algunos ejemplos de las herramientas que podemos construir utilizando el modelo SBERT¹⁰. En la documentación de esta librería, se da una explicación de cómo utilizar las representaciones de SBERT para construir un sistema de búsqueda semántica (*semantic search*)¹¹. Nos basaremos en esta explicación para construir nuestro sistema, y este será el uso principal que haremos del modelo BERT, en particular, de su modelo modificado, SBERT.

Los autores, en la descripción del uso de SBERT para tareas de búsqueda semántica, han definido dos tipos distintos de búsqueda semántica: la búsqueda semántica simétrica (*symmetric semantic search*) y la búsqueda semántica asimétrica (*asymmetric semantic search*). Procedemos, ahora, a explicar los dos tipos, y, para cada uno de ellos, exponer el detalle de cómo usaremos cada uno en la construcción de nuestro sistema.

3.3.1. Búsqueda semántica simétrica

En una búsqueda semántica simétrica, la consulta y los posibles resultados tienen más o menos la misma longitud. Así, por ejemplo, podríamos tener como consulta una oración, y la respuesta sería otra oración con una similitud semántica cercana y de una longitud similar.

Ahora, consideremos nuestro caso particular: los documentos disponibles en el Repositorio USM. ¿Qué cadenas de texto de estos documentos podemos utilizar para realizar una búsqueda semántica simétrica? Consideramos de gran interés tres cadenas en particular: el título del documento, la descripción (o resumen, o *abstract*) del documento, y el texto contenido en su archivo digitalizado (en este caso, archivo en formato PDF). De estas tres cadenas de texto, no podemos usar con SBERT, directamente, el texto completo del archivo digitalizado, ya que SBERT obtiene buenas representaciones para oraciones y párrafos, sin embargo, sus resultados no son tan buenos para cadenas de mayor tamaño, como puede ser, por ejemplo, el texto completo del archivo digitalizado ([Reimers y Gurevych, 2019]). Revisaremos más adelante qué podemos hacer en este caso.

Nos quedamos, entonces, con títulos y descripciones. El usuario podrá entregar un título o descripción de un posible documento que quisiese buscar dentro del repositorio, y la búsqueda retornará el documento con el título, o descripción, más similar a su consulta.

¿Cómo aprovecharemos, ahora, el texto que tenemos disponible en el archivo digitalizado del documento? Debido a que no podemos utilizarlo directamente con SBERT por su gran

¹⁰<https://sbert.net/index.html>

¹¹<https://sbert.net/examples/applications/semantic-search/README.html>

tamaño, proponemos la siguiente estrategia: dividir su contenido en partes más pequeñas. Ahora, ¿Qué división aplicaremos? Proponemos la siguiente: dividir el texto en sus secciones individuales, dado que el texto de cada uno de los documentos posee dicha división de manera orgánica. Así, usaremos el título y el texto de cada sección, de la misma manera que utilizaremos el título y la descripción (*abstract*) de cada documento completo. Así, el usuario tendrá la opción de buscar de forma general entre todos los documentos, o de forma particular entre las secciones de cada documento. De esta manera, se pretende ampliar tanto el rango como la profundidad de la búsqueda, utilizando esta estrategia de búsqueda semántica.

3.3.2. Búsqueda semántica asimétrica

En una búsqueda semántica asimétrica, la consulta y sus posibles resultados tienen una diferencia de longitud notable. Así, por ejemplo, podríamos tener como consulta una oración, y la respuesta sería un párrafo que guarde alguna relación con esta oración. Algunos ejemplos de este par oración-párrafo podrían ser una pregunta y un párrafo que responda a la pregunta, o una lista de palabras claves y un párrafo que guarde alguna relación con estas palabras claves.

Consideremos, ahora, cómo podemos utilizar este tipo de búsqueda con las cadenas de texto que tenemos para cada documento. Para este caso, proponemos, también, utilizar el título y la descripción de cada documento. Así, se tiene que el usuario entrega una oración corta (la cual puede ser una pregunta, o una colección de palabras), y recibe el documento con la descripción que guarde la mayor similitud con esta oración. Del mismo modo, el usuario puede entregar un párrafo extenso (que puede corresponder a la descripción propuesta de un documento), y recibir el título del documento que guarde la mayor relación con este párrafo.

Se utilizará la misma estrategia para cada una de las secciones de cada documento.

3.4. Desarrollo de la propuesta

En esta sección, se explicará cada una de las áreas de trabajo que comprenden la construcción del sistema, que corresponden a tres principalmente:

- **Modelo:** Se refiere a la búsqueda, elección y optimización del modelo a utilizar. Aquí es donde evaluaremos el rendimiento de diferentes modelos basados en BERT (o, en nuestro, caso, SBERT) pre-entrenados por la comunidad. Son estos modelos los que utilizaremos para procesar el texto de cada documento.
- **Herramienta:** Se refiere al diseño y construcción de la herramienta que utilizará el

usuario para realizar la búsqueda. Esta herramienta también estará encargada de almacenar las representaciones en vector de sentencias y párrafos.

- **Corpus:** Se refiere a la obtención y manejo de los datos del corpus de interés: el Repositorio USM. Esto incluye la evaluación de opciones para la obtención de los datos del corpus, la evaluación de opciones para procesarlos, y la evaluación de la información que utilizaremos de ellos.

En las secciones que siguen, se entra en el detalle del desarrollo/análisis propuesto para cada área de trabajo.

3.5. Modelo

El trabajo que realiza el modelo está en el corazón del sistema de búsqueda. Como se ha discutido, esta será la componente que traduzca la información que tenemos disponible (sentencias y párrafos) a una representación en vector. Así, la calidad de nuestra búsqueda dependerá de la calidad de la herramienta que utilizemos para obtener estas representaciones.

Ahora, ¿Cuál será el modelo que utilizaremos? ¿Conviene construir uno desde cero, o apoyarse en uno existente? Sobre esto, se decide: utilizaremos uno existente. En particular, utilizaremos los modelos pre-entrenados de SBERT ofrecidos por la librería SentenceTransformers. Como hemos discutido, SentenceTransformers hace una distinción entre dos tipos de búsqueda semántica: búsqueda semántica simétrica, y búsqueda semántica asimétrica. Por este motivo, también hace la misma distinción entre los modelos ofrecidos, dependiendo del tipo de búsqueda para la cual hayan sido entrenados en particular. Revisaremos, para cada tipo de búsqueda, los modelos propuestos a utilizar en nuestro sistema.

En la Figura 28, se puede ver una lista con los principales modelos pre-entrenados ofrecidos por SentenceTransformers, y su rendimiento en diferentes corpus de datos. Los modelos relevantes para nuestro trabajo, son los que tengan un mayor puntaje en la columna *Performance Semantic Search*

En cuanto a los modelos para búsqueda semántica simétrica, proponemos los siguientes candidatos:

- **paraphrase-multilingual-mpnet-base-v2:** Extensión del modelo *paraphrase-mpnet-base-v2* para más de cincuenta idiomas, entre ellos; español.
- **paraphrase-multilingual-MiniLM-L12-v2:** Extensión del modelo *paraphrase-multilingual-MiniLM-L12-v2*, para más de cincuenta idiomas; entre ellos, español.
- **distiluse-base-multilingual-cased-v1:** Modelo multilinguaje con soporte para más de quince idiomas; entre ellos, español.

Model Name	Performance Sentence Embeddings (14 Datasets) ⓘ	Performance Semantic Search (6 Datasets) ⓘ	Avg. Performance ⓘ	Speed ⓘ	Model Size ⓘ
multi-qa-mpnet-base-dot-v1 ⓘ	66.76	57.60	62.18	2800	420 MB
all-mpnet-base-v2 ⓘ	69.57	57.02	63.30	2800	420 MB
multi-qa-distilbert-cos-v1 ⓘ	65.98	52.83	59.41	4000	250 MB
multi-qa-MiniLM-L6-cos-v1 ⓘ	64.33	51.83	58.08	14200	80 MB
all-distilroberta-v1 ⓘ	68.73	50.94	59.84	4000	290 MB
all-MiniLM-L12-v2 ⓘ	68.70	50.82	59.76	7500	120 MB
all-MiniLM-L6-v2 ⓘ	68.06	49.54	58.80	14200	80 MB
paraphrase-multilingual-mpnet-base-v2 ⓘ	65.83	41.68	53.75	2500	970 MB
paraphrase-albert-small-v2 ⓘ	64.46	40.04	52.25	5000	43 MB
paraphrase-MiniLM-L3-v2 ⓘ	62.29	39.19	50.74	19000	61 MB
paraphrase-multilingual-MiniLM-L12-v2 ⓘ	64.25	39.19	51.72	7500	420 MB
distiluse-base-multilingual-cased-v1 ⓘ	61.30	29.87	45.59	4000	480 MB
distiluse-base-multilingual-cased-v2 ⓘ	60.18	27.35	43.77	4000	480 MB

Figura 28: Modelos pre-entrenados ofrecidos por SentenceTransformers y sus puntajes de rendimiento en diferentes tareas.

Fuente: SBERT.net

Se han escogido estos modelos como candidatos, ya que, según la tabla mostrada anteriormente, son, en el orden entregado, los modelos con mayor rendimiento para búsqueda semántica simétrica, y que además porque poseen soporte nativo para oraciones y párrafos en español.

Ahora, ¿Qué modelos utilizaremos para la búsqueda semántica asimétrica? Estos son los candidatos propuestos:

- **multi-qa-mpnet-base-dot-v1:** Modelo entrenado en pares de pregunta/respuesta de una gran colección de datos, entre ellos, el set de datos MSMARCO y el set de datos de Yahoo respuestas. Especialmente optimizado para similaridad basada en producto punto (*dot product*).
- **multi-qa-mpnet-base-cos-v1:** Modelo entrenado en pares de pregunta/respuesta de una gran colección de datos, entre ellos, el set de datos MSMARCO y el set de datos de Yahoo respuestas. Especialmente optimizado para similaridad basada en similitud coseno (*cosine similarity*).

Al igual que con los modelos de búsqueda semántica simétrica, estos modelos han sido escogidos por ser los de mayor rendimiento según las pruebas realizadas por los autores de SentenceTransformers. En este caso, no se ha encontrado, lamentablemente, algún modelo que haya sido entrenado para búsqueda semántica asimétrica para el idioma español, de modo que estos son, hasta ahora, los mejores candidatos.

3.6. Herramienta

Esta componente del sistema se refiere a la vista que tendrá el usuario para realizar la búsqueda y visualizar los resultados de esta. En esta sección, se da una descripción de los elementos principales de esta vista, junto con sus funcionalidades principales.

En primer lugar, el usuario será capaz de entregar una entrada en cadena de texto, que corresponderá a la cadena de búsqueda. Como ya hemos visto, esta entrada tendrá, principalmente, la forma de un título sugerido para un documento (o alguna de sus secciones) dentro del repositorio, o de una descripción sugerida para alguno de los documentos del repositorio (o alguna de sus secciones), dependiendo del tipo de búsqueda que quisiera realizar, ya sea búsqueda simétrica, o búsqueda asimétrica. Luego, el usuario tendrá también la posibilidad de poder seleccionar entre estos dos tipos de búsqueda, dependiendo del caso de uso.

Además de esto, el usuario tendrá también la posibilidad de filtrar su búsqueda considerando diferentes atributos y categorías de cada documento. Algunos de estos atributos corresponden al tópico del documento, autor del documento, departamento de la universidad al cual pertenece el autor del documento, entre otros. Más adelante, cuando se analice la información disponible sobre cada documento, se entrará en mayor detalle sobre cómo se construirá este filtro.

Una vez realizada la búsqueda, la vista mostrará aquellos documentos (o sus secciones) dentro del repositorio que tengan la mayor relevancia en relación con la búsqueda del usuario. La vista mostrará, para cada documento, un puntaje de relevancia o similaridad con respecto a la búsqueda del usuario, el título de dicho documento, y otros datos relevantes sobre el mismo. El usuario podrá escoger la cantidad de documentos más similares encontrados en la búsqueda, con el fin de poder ampliar el rango de esta, tolerando el retorno de documentos que sean posiblemente cada vez menos relevantes para la búsqueda del usuario.

Esta es, a grandes rasgos, la descripción de la herramienta con la cual interactuará el usuario para acceder a las funcionalidades del sistema. En cuanto a los detalles de su implementación, esta será desarrollada como una aplicación web escrita en lenguaje Python, que hará uso de la implementación del módulo SentenceTransformers dentro de este mismo lenguaje.

3.7. Corpus

En esta sección, se describe el proceso de recolección de los documentos de la colección del Repositorio USM, junto con sus datos. Esta sección comprende, por una parte, la narración del proceso para obtener estos datos de parte de sus administradores y, por otra, la descripción de los datos obtenidos para su uso posterior en la implementación final del sistema.

Así, procedemos a narrar primero la obtención de estos datos: en una primera instancia, se ha consultado en la biblioteca de la sede San Joaquín de la universidad, sede de la que forman

parte el alumno memorista y el profesor guía, sobre la posibilidad de contactar al personal encargado del Repositorio USM, con el fin de poder discutir, por un lado, la posibilidad de obtener los datos (corpus) que serán utilizados en la implementación final del sistema y, por otro, para exponer la propuesta desarrollada en este trabajo, con el fin de averiguar el interés que los administradores del repositorio pudiesen tener en una herramienta como esta. Durante esta consulta, una de las personas encargadas de la biblioteca facilitó al memorista el contacto de José Miguel Vega, administrador del Repositorio USM, quien forma parte del campus casa central de la universidad.

Luego de contactar al administrador vía correo, fue concretada una reunión en modalidad online junto al memorista. En esta reunión se habló, principalmente, un poco sobre la historia del repositorio, los principales formatos y estándares actualmente en uso para el archivo y búsqueda de documentos en el repositorio, los objetivos de trabajo actuales del repositorio y, finalmente, sobre la factibilidad e interés de los administradores en el sistema propuesto. Se ha llegado a un acuerdo donde, una vez armado el prototipo inicial del sistema, esta será presentada a los administradores para que puedan hacer uso de este, y entregar una retroalimentación y opiniones sobre el mismo. Finalmente, con respecto al acceso del corpus de datos para su uso en el desarrollo del sistema, el administrador del repositorio facilitó al memorista el contacto del director de la biblioteca de la universidad, Rimsky Espindola, para solicitar la autorización de acceso a esta información, y poder así concretar detalles sobre los objetivos del desarrollo de este sistema y privacidad de los datos que se manejarán.

El memorista ha contactado al director vía correo electrónico, y ha concretado una reunión en modalidad online. En esta reunión, se ha expuesto la idea principal de la propuesta, y se ha hablado un poco sobre el sistema de almacenamiento de documentos en el repositorio. Principalmente, se ha discutido la existencia de tres categorías de documentos dentro del repositorio: documentos de categoría A, B y C. A continuación, se da una breve descripción de cada una de estas categorías.

Los documentos de categoría A corresponden a documentos de acceso abierto, para cualquier individuo. Esto incluye también a individuos que no forman parte o no tienen relación alguna con la universidad. Los documentos de categoría B corresponden a los documentos de acceso restringido solo a integrantes de la comunidad de la universidad y, por lo tanto, requieren de credenciales institucionales para su acceso. Finalmente, los documentos de categoría C corresponden a documentos de carácter confidencial, para los cuales su acceso debe ser solicitado ya sea por medio del autor del documento, o por medio de la organización con la que pudo haber trabajado el autor durante el desarrollo de su propuesta, y que haya requerido la confidencialidad del documento. Es decir, corresponde a documentos cuyo trabajo ha requerido confidencialidad entre las partes.

Finalmente, se ha acordado la facilitación de un corpus de documentos disponibles en el repositorio, los cuales forman parte de la categoría A de documentos, es decir, documentos de acceso abierto tanto para miembros de la comunidad universitaria como para externos. Este corpus de documentos fue facilitado en formato de archivo de texto plano, en un archivo en formato CSV, donde cada fila del archivo corresponde a un documento del corpus, y cada

columna corresponde a un atributo del documento, como puede ser su título, su descripción, sus autores, su categoría, etc. Serán estos datos los que utilizaremos durante el desarrollo de nuestra herramienta.

Esto concluye la narración de la obtención del corpus de datos por parte de sus administradores. A continuación, se da una descripción breve de los datos obtenidos.

3.7.1. Datos en archivo plano

Como se ha dicho, los datos corresponden a un archivo plano en formato CSV, donde cada fila del archivo corresponde a un documento individual, y cada columna, corresponde a distintos atributos del documento. La Tabla 1 muestra el detalle de cada uno de los atributos disponibles para cada documento.

Tabla 1: Atributos de los documentos disponibles en el archivo plano del corpus de datos.

#	Column	Non-Null Count	Dtype
0	id	2682	object
1	collection	2682	object
2	dc.contributor.advisor	1386	object
3	dc.contributor.advisor[]	1101	object
4	dc.contributor.advisor[es_CL]	1	object
5	dc.contributor.author	1323	object
6	dc.contributor.author[]	1357	object
7	dc.contributor.department[]	13	object
8	dc.contributor.department[eng]	3	object
9	dc.contributor.department[es_CL]	2659	object
10	dc.contributor.other	423	object
11	dc.contributor.other[]	785	object
12	dc.contributor.other[es_CL]	1	object
13	dc.contributor[es_CL]	4	object
14	dc.coverage.spatial[]	2	object
15	dc.coverage.spatial[eng]	2	object
16	dc.coverage.spatial[es_CL]	2618	object
17	dc.creator	2	object
18	dc.date.accessioned	2169	object
19	dc.date.accessioned[]	515	object
20	dc.date.available	2169	object
21	dc.date.available[]	515	object
22	dc.date.issued	1323	object

Continúa en la siguiente página

Tabla 1 – Continúa de la siguiente página

#	Column	Non-Null Count	Dtype
23	dc.date.issued[]	1358	object
24	dc.description.abstract[]	3	object
25	dc.description.abstract[en]	42	object
26	dc.description.abstract[eng]	617	object
27	dc.description.abstract[es_CL]	2619	object
28	dc.description.degree	2	object
29	dc.description.degree[]	2	object
30	dc.description.degree[es_CL]	2678	object
31	dc.description.program[]	774	object
32	dc.description.program[es_CL]	1600	object
33	dc.description.provenance[en]	2681	object
34	dc.description.sponsorship	7	object
35	dc.description[]	3	object
36	dc.description[eng]	3	object
37	dc.description[es_CL]	1071	object
38	dc.format.extent[]	1079	object
39	dc.format.extent[es_CL]	1596	object
40	dc.format.medium[]	1077	object
41	dc.format.mimetype[]	1511	object
42	dc.identifier.barcode[]	1080	object
43	dc.identifier.barcode[es_CL]	1592	object
44	dc.identifier.uri	2079	object
45	dc.identifier.uri[]	602	object
46	dc.language.iso[]	13	object
47	dc.language.iso[es_CL]	39	object
48	dc.publisher[]	17	object
49	dc.publisher[es_CL]	39	object
50	dc.rights.accessRights[]	1500	object
51	dc.rights.accessRights[es_CL]	12	object
52	dc.rights.license[]	1	object
53	dc.rights.uri[]	1	object
54	dc.rights[]	1	object
55	dc.source.uri[]	56	object
56	dc.subject.other	18	object
57	dc.subject.other[]	4	object
58	dc.subject.other[es_CL]	440	object
59	dc.subject[]	8	object
60	dc.subject[eng]	3	object
61	dc.subject[es_CL]	2670	object
62	dc.title[Español]	1	object

Continúa en la siguiente página

Tabla 1 – Continúa de la siguiente página

#	Column	Non-Null Count	Dtype
63	dc.title[]	6	object
64	dc.title[eng]	3	object
65	dc.title[es_CL]	2671	object
66	dc.type.driver	2	object
67	dc.type[]	1596	object
68	dc.type[eng]	3	object
69	dc.type[es_CL]	848	object
70	dcterms.title[]	1	object
71	dspace.entity.type	2682	object
72	usm.campus.sede[]	2	object
73	usm.date.thesisregistration[]	1060	object
74	usm.identifier.rut	36	object
75	usm.identifier.rut[]	1071	object
76	usm.identifier.rut[es_CL]	430	object
77	usm.identifier.thesis[]	1060	float64

Como podemos ver, tenemos varios atributos disponibles. Los que serán de interés principal para la construcción del sistema propuesto, son el título del documento en español (dc.title[es_CL]) y el *abstract* (descripción) del documento en español (dc.description.abstract[es_CL]). Además, nos interesa también el departamento de la universidad al cual pertenece el documento (dc.contributor.department[es_CL]), para así poder filtrar los resultados de la búsqueda en base a esta información.

3.7.2. Archivos digitalizados

La mayor parte de los documentos que forman parte del corpus posee un archivo digitalizado con el contenido del mismo, al cual podemos acceder por medio de los datos disponibles en la planilla. Estos archivos están disponibles en formato PDF.

Debemos considerar una estrategia adecuada para el manejo de estos archivos en formato digital, y de la información contenida en ellos en formato de texto. Esto significa, principalmente, la aplicación de un método adecuado para la extracción del texto de cada documento, lo cual significa la separación del texto extraído en cada una de sus secciones, para luego poder ser procesado correctamente por el modelo, según lo que hemos propuesto.

La extracción adecuada de la información contenida en el documento resulta una tarea particularmente complicada, esto debido a la heterogeneidad del formato y estructura de los documentos disponibles en el corpus de datos del repositorio. Si bien, en general, los documentos presentan un formato y estructura en común, esto es, el formato de documentos de memoria o tesis de grado de alumnos de la universidad, el procesamiento de todos los

documentos en general se dificulta al momento de considerar las pequeñas diferencias individuales de formato para cada documento en particular.

Es por esta razón, que se decide hacer uso de una herramienta que logre capturar la estructura de cada archivo PDF para poder extraer el texto de cada archivo en partes separadas, con su estructura correspondiente. Para ello, se hace uso de la librería PyPDF¹². Esta librería permite manipular archivos PDF para distintas tareas. Una de ellas, corresponde a la extracción de texto y metadatos de los archivos, que es la que utilizaremos en este caso. Mas adelante entraremos en detalle sobre el procesamiento de los textos de los archivos digitalizados de cada documento.

3.8. Modelamiento de tópicos con BERTopic

Por último, una vez obtenidas las representaciones para cada documento, mostraremos cómo pueden utilizarse para algo más que solo búsqueda semántica. Mostraremos su uso con BERTopic, una técnica de modelamiento de tópicos que hace uso de estas representaciones para crear clústeres densos, creando así tópicos fácilmente interpretables para el corpus de documentos [Grootendorst, 2022].

Se presenta esta aplicación como un ejemplo de cómo se pueden utilizar estas representaciones en vector del contenido de los documentos, para asistir en su procesamiento a través de un enfoque distinto a la búsqueda semántica.

¹²<https://pypdf2.readthedocs.io/en/3.x/>

CAPÍTULO 4

VALIDACIÓN DE LA SOLUCIÓN

En esta sección revisaremos, parte por parte, la implementación de cada una de las secciones descritas en la propuesta del capítulo anterior.

4.1. Modelo

En esta primera parte, se describirá la elección del modelo SBERT propuesto, para ambos casos de búsqueda: para la búsqueda simétrica, y para la búsqueda asimétrica.

Para ambos casos, se propone la siguiente métrica para seleccionar entre los modelos propuestos: se evaluará cada modelo en una búsqueda para los primeros k resultados, con un k desde 1 hasta 10. Se escogerán cinco títulos diferentes extraídos desde el corpus de datos, y para cada título, se propondrán diez títulos similares, con el mismo significado semántico, pero escritos de manera distinta. Así, por ejemplo, para el título

"Estudio de Prefactibilidad Técnico Económica para Crear una Empresa Constructora Dedicada a la Construcción de Vivienda Social Modalidad Fondo Solidario de Vivienda"

Se proponen los siguientes títulos similares

"Análisis Técnico-Económico para el Establecimiento de una Constructora de Vivienda Social bajo el Esquema del Fondo Solidario de Vivienda",

"Evaluación de Factibilidad para la Creación de una Empresa Constructora Especializada en Vivienda Social"

y

"Estudio de Viabilidad para Iniciar una Constructora de Vivienda Social con Apoyo del Fondo Solidario"

etc. Para cada título similar, se obtendrá la cantidad de veces que aparece para cada k , para así obtener, para cada valor de k , la cantidad de veces que el modelo logra encontrar el título que se estaba buscando.

En cuanto a la búsqueda simétrica, es decir, para la comparación entre los modelos *mpnet*, *minilm* y *distil*, se obtienen los resultados que se muestran en la Figura 29. Como se puede observar en esta figura, el modelo *mpnet* es el que obtiene la mayor cantidad de resultados precisos para cada valor de k , convergiendo a 44 títulos encontrados cuando la precisión de k es igual a 10 (es decir, se encuentra dentro de los primeros diez títulos). Atendiendo a estos

resultados, se escoge a *mpnet* como nuestro modelo base para la obtención de representaciones en vector para búsqueda simétrica.

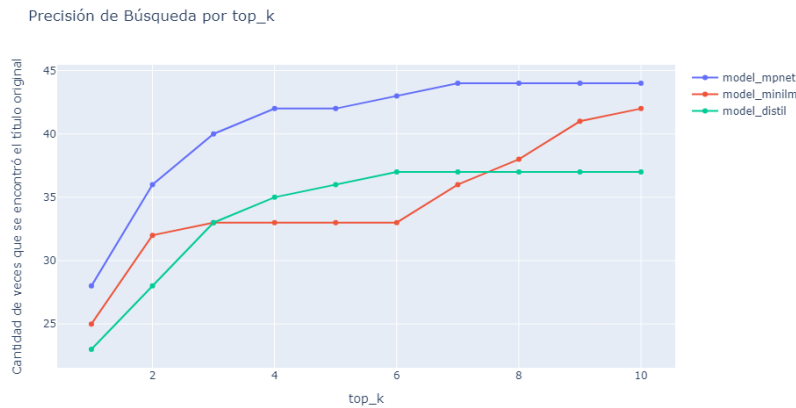


Figura 29: Rendimiento de los modelos *mpnet*, *minilm* y *distil* para búsqueda simétrica en el corpus de documentos

Fuente: *Elaboración propia*

Ahora, con respecto a la búsqueda asimétrica, se hace uso de la misma métrica, pero cambiando los pares de sentencias y párrafos que utilizaremos para la búsqueda. Ahora, escogemos cinco *abstracts* de documentos dentro del corpus, y, para buscarlos, utilizaremos diez conjuntos de palabras clave con mismo un significado semántico, pero con sintaxis diferente para cada documento. Así, por ejemplo, para el *abstract* del documento con el título

"Flujo Óptimo de Potencia Considerando Restricciones y Criterios de Estabilidad de Tensión"

Tendremos los siguientes conjuntos de palabras clave

"Flujo óptimo de potencia, estabilidad de tensión, minimización de costos",

"Algoritmo computacional, despacho de potencia, restricciones de seguridad"

y

"Pyomo, Python, solver ipopt, solver Bonmin"

etc. Utilizando esta comparación para los diferentes modelos de búsqueda asimétrica, es decir, los modelos *dot* y *cos*, obtenemos los resultados que se muestran en la Figura 30. Como se puede observar, la precisión de búsqueda para ambos modelos es bastante similar, pero puede apreciarse una ligera ventaja para el modelo *dot* para algunos valores de *k*. Así, escogemos este modelo para usarlo en nuestra búsqueda asimétrica.

Es importante señalar que, a pesar de que el rendimiento de ambos modelos sea parecido, esto no significa que sea un rendimiento particularmente bueno: observamos que ambos

modelos llegan a una precisión de 26 *hits* cuando se considera el máximo valor de k en 10, lo cual no es un resultado particularmente bueno, considerando un total de cincuenta conjuntos de palabras para cada búsqueda. Esto puede deberse principalmente a que, como se ha mencionado, estos modelos no están particularmente entrenados para ser utilizados con corpus de texto en idioma español, a diferencia de los modelos propuestos para la búsqueda simétrica, los que hemos visto han obtenido mejores resultados en general.

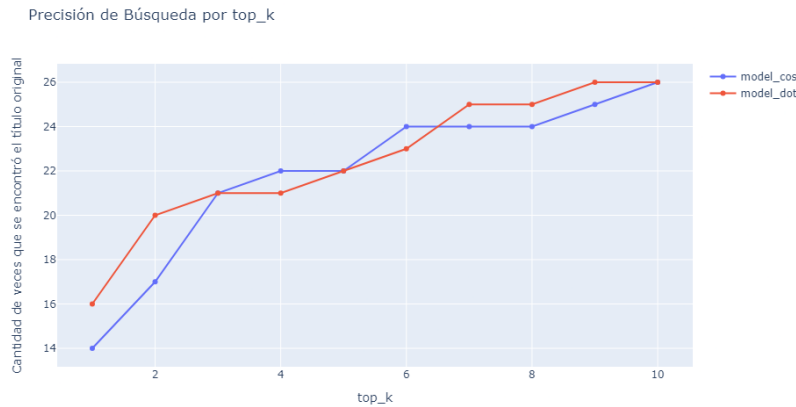


Figura 30: Rendimiento de los modelos *dot* y *cos* para búsqueda asimétrica en el corpus de documentos

Fuente: *Elaboración propia*

Así, finalmente, escogemos nuestro modelo definitivo para ambos casos de búsqueda.

4.2. Herramienta

En esta sección, se muestra el proceso de construcción de la herramienta, en base a la propuesta descrita en el capítulo anterior.

Como se ha mencionado, esta herramienta estará implementada en lenguaje Python. La funcionalidad principal hará uso de la implementación en este lenguaje la librería Sentence-Transformers, desarrollada por los autores del modelo SBERT. Por último, se implementará su funcionalidad principal en una *demo* básica en forma de aplicación web, utilizando el framework Flask en python para esta implementación.

En primer lugar, se implementa una barra de búsqueda básica, donde el usuario puede ingresar la cadena de texto correspondiente a su búsqueda. Junto a este barra, se implementa la selección para el tipo de modelo, es decir, entre modelo de búsqueda simétrica o asimétrica. La Figura 31 muestra la implementación de estos elementos en la herramienta.

Búsqueda semántica documentos Repositorio USM



Figura 31: Implementación de barra de búsqueda y selección de tipo de búsqueda en la herramienta.

Fuente: Elaboración propia

Luego, se implementa el campo de filtrado para los resultados mostrados por el sistema. Luego de realizar el análisis de los datos, se implementa un filtro para especificar el departamento de la universidad al cual pertenece el documento buscado. La Figura 32 muestra la implementación de esta característica en el sistema.

Búsqueda semántica documentos Repositorio USM

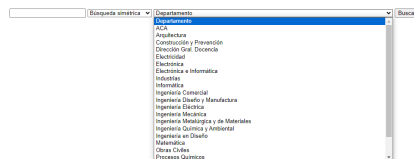


Figura 32: Implementación de filtrado por departamento en la herramienta.

Fuente: Elaboración propia

Se implementa también la vista para los resultados. Se implementa en como una lista simple, con un ítem para cada uno de los elementos encontrados. Los resultados muestran el puntaje de relevancia del documento, junto con el título del documento, y su departamento dentro de la universidad. El usuario tiene la posibilidad de hacer click en cada uno de estos títulos, lo cual lo redireccionará a la página del documento alojado dentro del sitio del repositorio, en caso de que los datos para este documento contengan dicha dirección. Por último, se agrega también la opción de seleccionar la cantidad de documentos que retornará la búsqueda. La Figura 33 muestra la implementación del sistema hasta este punto.

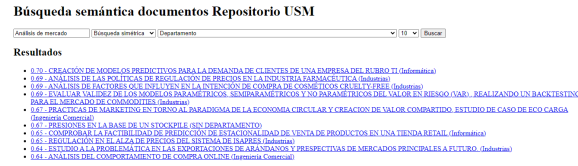


Figura 33: Vista de resultados de la herramienta
Fuente: Elaboración propia

Finalmente, se implementa la selección de búsqueda en el corpus de documentos, o la selección de búsqueda sobre las secciones de cada documento. Ambas opciones de búsqueda pueden hacer uso de búsqueda simétrica y asimétrica, y del filtro por departamento del documento. La Figura 34 muestra la implementación de la búsqueda por secciones de documentos. En esta búsqueda se puede apreciar tanto el título de cada sección, como su contenido.

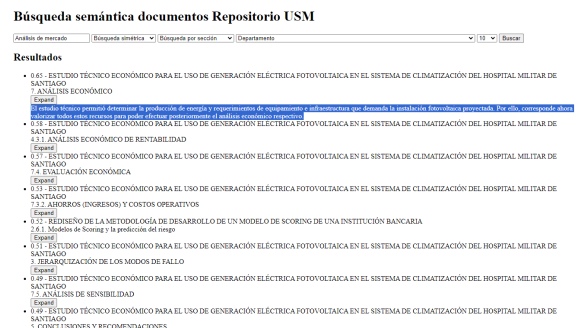


Figura 34: Búsqueda por secciones de documento
Fuente: Elaboración propia

Así, esta es la implementación final de la herramienta. Como se puede observar, corresponde a una aplicación bastante simple, pero que incluye las funcionalidades principales de la propuesta de este trabajo. Se hará uso de esta herramienta para realizar la evaluación final con usuarios de esta propuesta.

4.3. Corpus

En esta sección, se describe el proceso de obtención y manipulación de información de los documentos del corpus.

En primer lugar, se procesa el archivo CSV con los datos y metadatos de cada documento utilizando la librería Pandas en Python. De este archivo, seleccionamos los atributos que nos son de interés: el título, y el abstract de cada documento. También seleccionamos el departamento de cada documento, para poder realizar el filtrado de la búsqueda

Luego, para el procesamiento de los archivos PDF de cada documento, se hace uso de la librería PyPDF2 para extraer los textos de estos archivos. Esta librería extrae directamente el texto de cada archivo. Sin embargo, este texto es extraído en un formato no estructurado, es decir, debemos procesar este texto para darle una estructura que podamos utilizar para su procesamiento final. Debido a la heterogeneidad en el formato y estructura de cada documento, no es posible procesar los archivos digitalizados del corpus completo de forma automatizada para extraer el texto de cada sección de manera precisa. Por este motivo, para el caso de desarrollo de esta memoria, aplicaremos un procedimiento manual para extraer y estructurar el texto de cada sección de una muestra de los documentos del corpus.

Debido a que el procesamiento manual de cada documento debe realizarse con cuidado, el procedimiento para cada documento toma un tiempo considerable, de modo que se ha tomado una muestra total de cinco documentos, de los cuales se han extraído y estructurado un total de 191 secciones. Si bien esta no es una muestra de gran tamaño particularmente, juzgamos que esta cantidad será suficiente para servir como una muestra de la división de documentos y procesamiento de sus textos que se propone en este trabajo. El procesamiento de la totalidad de documentos digitalizados del corpus queda como un trabajo propuesto futuro. Se entrará en mayor detalle sobre esto en las conclusiones del trabajo.

Así, siguiendo este procedimiento, para cada documento se crea de forma manual un archivo en formato JSON, cuya estructura puede observarse en el Listing 1.

Listing 1: Estructura del archivo JSON para cada documento

```
{
  "title": "",
  "sections": {
    "0": {
      "title": "",
      "text": "",
      "sections": {}
    },
    "1": {
      "title": "",
      "text": "",
      "sections": {
        "0": {
          "title": "",
          "text": "",
          "sections": {}
        },
        ...
      }
    },
    ...
  }
}
```

```

    }
    },
    ...
}
}

```

Como se puede apreciar, se trata de una estructura recursiva para cada sección del texto. Cada una de estas secciones será tratada como un documento completo, considerando, en este caso, el título de cada sección (representado por el atributo *title*) y el texto de cada sección (representado por el atributo *text*), los cuales serán utilizados para realizar la búsqueda.

Finalmente, se combinan todos los archivos JSON en un solo archivo plano CSV con tres atributos: nombre del documento, nombre de la sección, y texto de la sección, con el objetivo de procesar esta información de la misma forma en que se ha procesado el archivo CSV del corpus completo.

4.4. Modelamiento de tópicos con BERTopic

En esta sección, se describe otro ejemplo de uso para las representaciones en vector obtenidas con SBERT: el modelamiento de tópicos del corpus de documentos utilizando BERTopic. Así, se pretende dar un ejemplo de cómo se pueden utilizar estas representaciones para realizar una caracterización del contenido de corpus de documentos de gran tamaño, más allá del enfoque principal de este trabajo de memoria, que corresponde a su uso en búsqueda semántica.

BERTopic es una técnica de modelamiento de tópicos que utiliza la representación en vector de texto y c-TF-IDF para crear clústeres de estos y así obtener tópicos fácilmente interpretables, manteniendo palabras importantes del texto en la descripción de estos tópicos [Grootendorst, 2022]. Para nuestro caso de uso, utilizaremos el título de cada documento para la obtención de tópicos.

El detalle de los primeros diez tópicos obtenidos utilizando esta técnica puede observarse en la Tabla 2. En total, se obtuvieron 39 tópicos diferentes. El primero de ellos, con etiqueta -1, es un tópico para *outliers* dentro del corpus: puede ignorarse, ya que no es realmente representativo de ningún agrupamiento particular.

Tabla 2: Tópicos obtenidos con BERTopic.

Topic	Count	Name	Representation
-1	804	-1_de_para_en_la	[de, para, en, la, un, el, una, del, estudio, ...]

Continued on next page

Tabla 2 – continued from previous page

Topic	Count	Name	Representation
0	265	0_de_en_la_para	[de, en, la, para, del, santa, universidad, fe, ...]
1	171	1_chile_en_el_del	[chile, en, el, del, la, chileno, de, chilena, ...]
2	166	2_planta_plan_mantenimiento_de	[planta, plan, mantenimiento, de, preventivo, ...]
3	151	3_energética_energía_eléctrica_en	[energética, energía, eléctrica, en, generación, ...]
4	111	4_solar_fotovoltaicos_radiación_fotovoltaico	[solar, fotovoltaicos, radiación, fotovoltaico, ...]
5	95	5_con_hormigón_fibras_hormigones	[con, hormigón, fibras, hormigones, del, refuerzo, ...]
6	84	6_empresa_prefactibilidad_una_dedicada	[empresa, prefactibilidad, una, dedicada, crear, ...]
7	62	7_minera_compañía_mineros_mineral	[minera, compañía, mineros, mineral, de, doña, ...]
8	61	8_aerolíneas_latam_modelo_airlines	[aerolíneas, latam, modelo, airlines, drones, ...]
9	55	9_utfsm_informática_web_civil	[utfsm, informática, web, civil, datos, ingeniería, ...]

Como puede observarse, cada tópico está construido a partir de palabras claves de cada clúster de títulos. Los títulos vienen a ser las primeras cuatro palabras más representativas de cada tópico.

Los primeros dos tópicos de mayor tamaño corresponden a palabras o conectores que se repiten con mucha frecuencia en los documentos. En el caso del tópico 0, parece tratarse de la estructura general del título de los documentos del repositorio, es decir, de una estructura general para los títulos, además del tópico referente a la universidad federico santa maría, en particular, la cual es mencionada por una gran parte de los documentos del repositorio en su título.

En el tópico 1, ocurre algo similar. Las palabras más importantes parecen corresponder a palabras que forman parte de la estructura general de los títulos de documentos del repositorio. Esta vez vemos también como el país, chile, aparece como una palabra recurrente y representativa, tal como ocurría en el caso del tópico anterior con la palabra universidad.

Ya a partir del tópico 2, comenzamos a notar el agrupamiento en áreas y tópicos de carácter más específico. Este tópico está representado por las palabras "planta", "plan", "mantenimiento", "preventivo", etc. todo lo cual parece tener que ver con el tópico de plantas (en el contexto de industria) y planes. En el tópico 3, aparecen las palabras "energética", "energía",

"eléctrica" y "generación", que parecen tener que ver con el área de energía. En el tópico 4, aparecen las palabras "solar", "fotovoltaico" y "radiación", que parecen tener que ver con el tópico de energía solar. Y así, esto sucede con los diferentes tópicos detectados por BERTopic.

Este análisis nos puede servir para obtener una idea general de los tópicos tratados en un corpus de documentos, utilizando las representaciones de los títulos de cada documento obtenidas por SBERT.

Ahora, también podemos obtener representaciones visuales sobre estos tópicos. En la Figura 35, puede observarse la primera de estas representaciones, la cual corresponde a un mapa de distancia entre tópicos. Este mapa representa cada tópico, su tamaño, y su distancia con respecto a otros tópicos.



Figura 35: Mapa de tópicos y la distancia entre ellos.

Fuente: Elaboración propia

Esta representación nos puede dar una idea visual de la distribución o agrupamiento de documentos del corpus en distintos tópicos, y la distancia semántica que existe entre todos ellos. En esta figura, por ejemplo, observamos un clúster de tópicos de gran tamaño en medio del mapa. Si observamos el detalle de este clúster, nos encontramos con el tópico 2, el

tópico 7, y el tópico 6, cuya descripción puede observarse en la Tabla 2 referida anteriormente. Podemos observar una cercanía semántica entre estos tópicos, la cual tiene que ver con el tópico de plantas, empresas y compañías. Esta cercanía o relación queda representada en este mapa.

Otra área destacada en este mapa aparece en la zona de superior a la derecha, donde aparecen dos tópicos de gran tamaño. Estos son los tópicos 3 y 4, ambos los cuales tienen que ver con energía.

Podemos observar otros agrupamientos destacados en este mapa. Sin embargo, no entraremos en mayor detalle sobre ellos. Nos basta con los mencionados para dar una idea de esta visualización

La segunda representación, corresponde a una estructuración jerárquica de los tópicos. Puede observarse en la Figura 36. Se utiliza una función de *ward linkage* para obtener una jerarquización de clustering basada en la matriz de distancia coseno entre las representaciones de los tópicos.

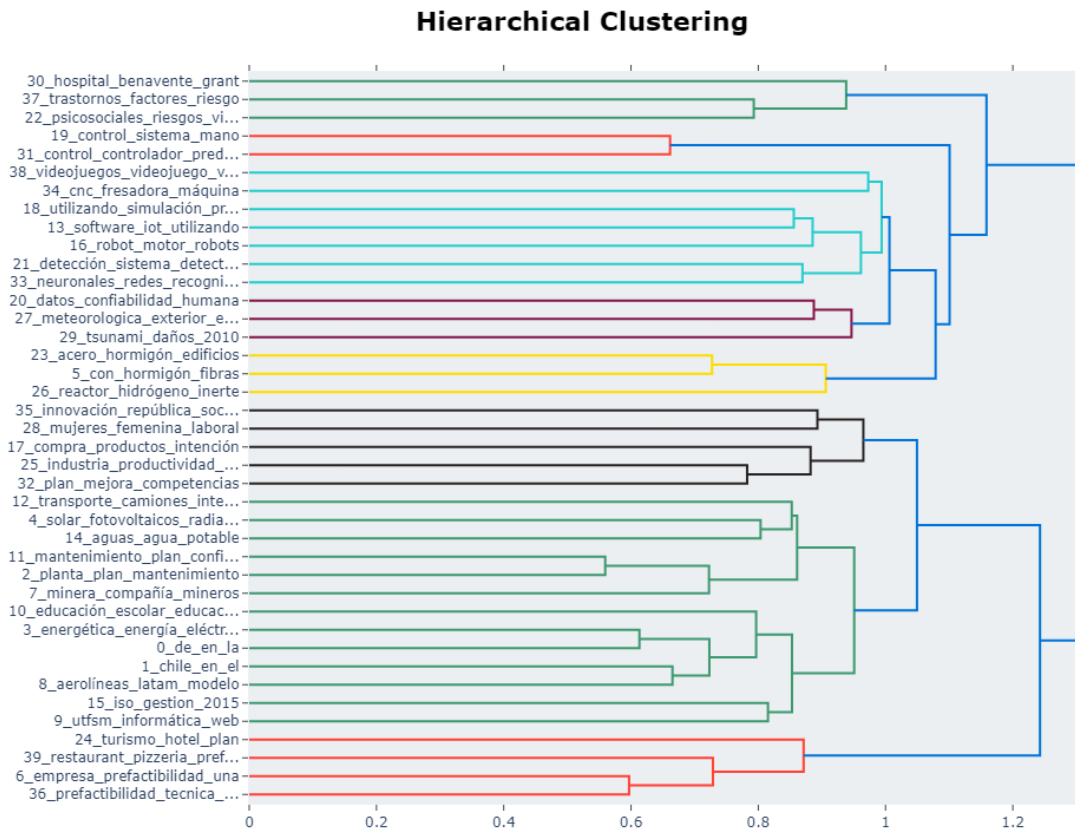


Figura 36: *Clustering* jerárquico de tópicos.

Fuente: Elaboración propia

En esta estructura, observamos como se ordenan los diferentes tópicos. Observamos, por ejemplo, como se unen el tópico 18, que habla de "simulación", y el tópico 13, que habla de "software", a un mismo nivel jerárquico. A su vez, estos dos tópicos se unen al tópico 16, que habla de "robot", en un nivel jerárquico más arriba.

Vemos también como el tópico 4, que habla de "fotovoltaicos", se une en el mismo nivel junto con el tópico 14, que habla de "agua potable".

Y de esta manera podemos observar, en general, la estructura jerárquica que otorga el modelo a los diferentes tópicos del corpus de documentos.

La tercera representación, corresponde a un gráfico de barras para algunos de los tópicos. Puede observarse en la Figura 37. Para cada tópico, se observa el puntaje c-TF-IDF obtenido para cada uno de cada palabra en relación a este tópico. Así, observamos las palabras más relevantes.^{en} la representación de cada tópico.

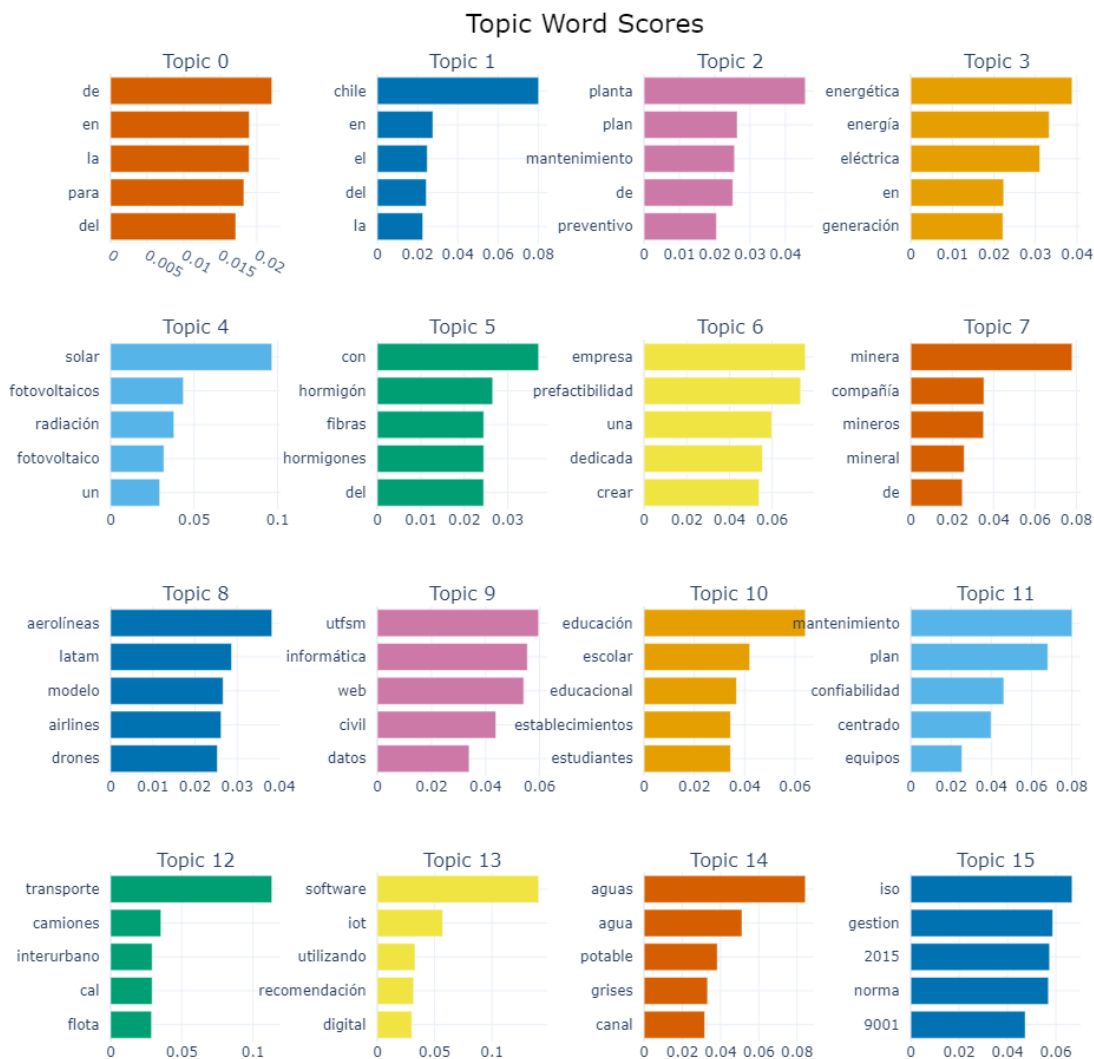


Figura 37: Puntaje de las palabras más relevantes para cada tópico.

Fuente: Elaboración propia

En este ejemplo, observemos algunas de las palabras más representativas para cada tópico, es decir, con un mayor puntaje en comparación al resto de palabras que componen cada tópico. Esto sucede de forma evidente, por ejemplo, en el caso del tópico 2, donde domina, con diferencia, la palabra "planta". También sucede lo mismo con los tópicos 4, 7, 10 y 12, donde dominan las palabras "solar", "minera", "educación" y "transporte", respectivamente, lo cual nos habla de temas muy marcados para cada uno de estos tópicos.

Esta representatividad marcada no ocurre en todos los casos. En el tópico 0, por ejemplo, ninguna palabra o conjunto de palabras parece dominar el tópico con gran diferencia. Lo mismo sucede en el caso del tópico 15.

Por último, se tiene una representación para la matriz de similitud entre tópicos. Puede observarse en la Figura 38. Basado en la similitud coseno entre representaciones de los tópicos, se crea un mapa de calor que muestra la similitud entre tópicos.

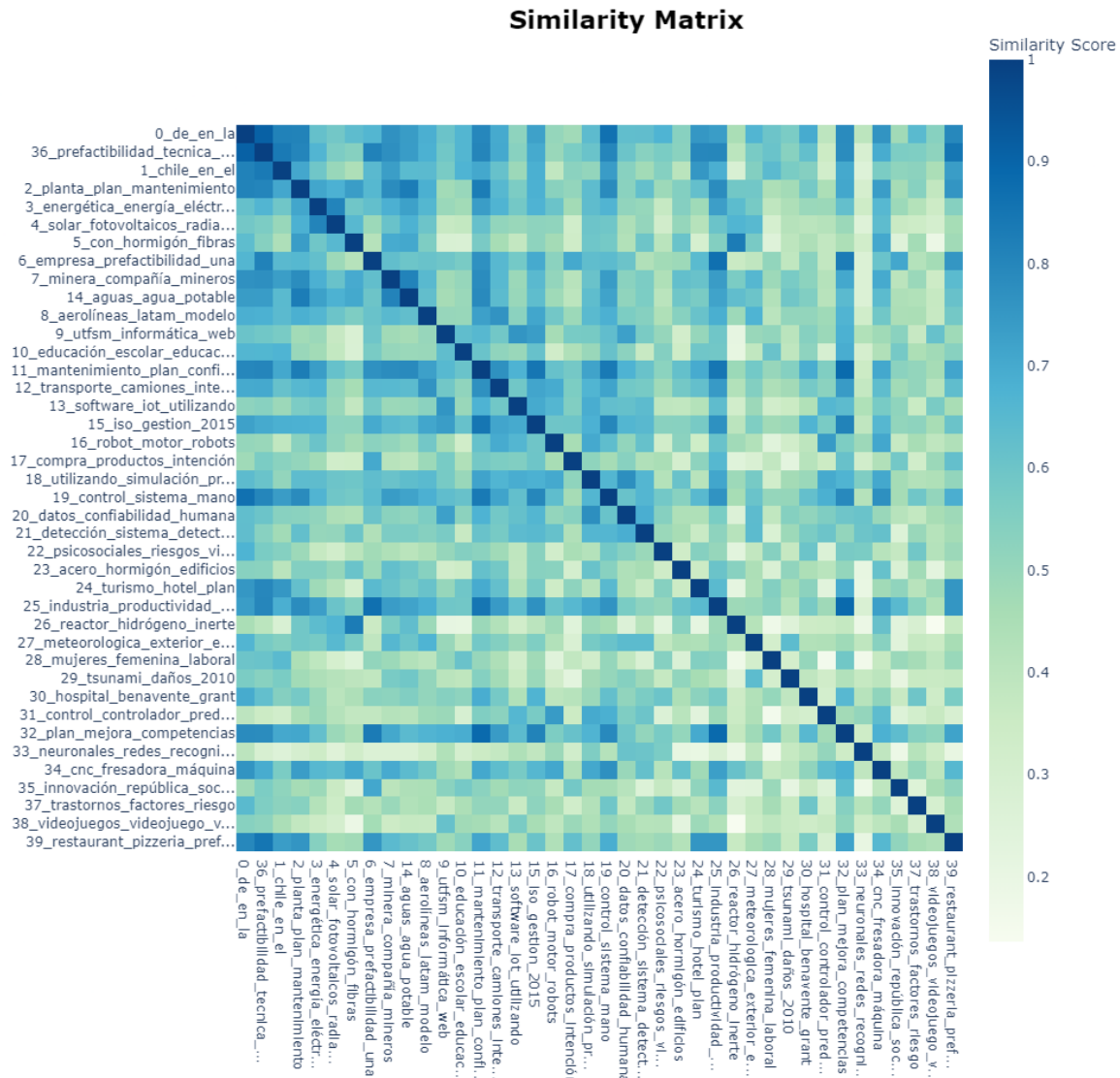


Figura 38: Matriz de similitud entre tópicos.

Fuente: Elaboración propia

En esta matriz, algunos de los tópicos con mayor similitud entre ellos son, por ejemplo, entre los tópicos 25, que habla de "industria", y el tópico 6, que habla de "empresa", o entre el tópico 15, que habla de "iso", y el tópico 32, que habla de "plan".

En general, a partir de este modelamiento, y algunas de sus representaciones, podemos ex-

traer información valiosa sobre el contenido semántico de los documentos de un corpus. Si bien esta caracterización del corpus puede no resultar precisa en todos los casos, particularmente en comparación a otras técnicas tradicionales de clustering, si nos sirve para obtener una idea aproximada, y para obtener una caracterización distinta a la que es posible obtener con métodos clásicos.

4.5. Encuesta de opinión

Se ha desarrollado una encuesta para recolectar la opinión de usuarios con respecto a la metodología de búsqueda implementada en este trabajo.

En esta encuesta, se busca obtener una opinión subjetiva de la experiencia de usuario. No se busca una opinión objetiva puesto que, como se ha dicho, el objetivo de esta propuesta no es el de reemplazar al actual sistema de búsqueda, sino que se desea dar una demostración de esta nueva metodología. Además, no se han encontrado métricas objetivas suficientemente satisfactorias como para realizar esta comparación.

Los usuarios escogidos para realizar esta encuesta corresponden a alumnos de la universidad y posibles externos que deseen hacer uso del Repositorio USM.

En esta encuesta, se realizan las siguientes preguntas:

- En cuanto a la propuesta de búsqueda semántica en general: ¿Cuáles son tus impresiones?
- En cuanto a la propuesta de búsqueda semántica simétrica en particular: ¿Cuáles son tus impresiones?
- En cuanto a la propuesta de búsqueda semántica asimétrica en particular: ¿Cuáles son tus impresiones?
- En cuanto a la propuesta de búsqueda por secciones: ¿Cuáles son tus impresiones?

Además, se pide entregar comentarios finales adicionales.

En cada pregunta, se pide dar una breve impresión de la experiencia del usuario con respecto a un ítem en particular. Adicionalmente, se pide entregar posibles puntos positivos y negativos sobre el ítem de cada enunciado.

Estas preguntas buscan extraer una opinión en base a tres ítems principales del sistema desarrollado: búsqueda semántica en general, búsqueda simétrica/asimétrica en particular y búsqueda por secciones.

En cuanto a la búsqueda semántica en general, se desea conocer la opinión general con respecto a esta metodología de búsqueda. Interesa conocer si acaso existe un interés por esta metodología, si se considera viable, o cualquier otra ventaja/desventaja que pueda hacerse notar.

En cuanto a la búsqueda simétrica/asimétrica, se desea conocer la opinión particular sobre cada modalidad de búsqueda. Interesa conocer si acaso una modalidad es preferida por sobre la otra, qué modalidad es la más adecuada para cada caso de uso, qué dificultades existen en la elaboración de consultas para cada modalidad, etc.

En cuanto a la búsqueda por secciones, ya que esta corresponde a un enfoque particular expuesto y desarrollado por los autores en este documento, se desea conocer la viabilidad de este enfoque para realizar una búsqueda a través de la información contenida en cada documento.

4.6. Resultados de la encuesta

Se obtuvo un total de siete respuestas durante la realización de esta encuesta.

En general, para cada pregunta, se identificaron los siguientes puntos principales sobre cada ítem:

En cuanto a la propuesta de búsqueda semántica en general: ¿Cuáles son tus impresiones?

- Útil
- Búsqueda de carácter intuitivo
- Entrega de resultados relevantes
- Entrega de resultados que de otro modo podrían ser pasados por alto
- Entrega rápida de resultados

En cuanto a la propuesta de búsqueda semántica simétrica en particular: ¿Cuáles son tus impresiones?

- Búsqueda de títulos similares sin necesidad de exactitud
- Búsqueda de títulos complejos a partir de ideas simples
- Entrega rápida de resultados
- Entrega de algunos resultados con baja relevancia

En cuanto a la propuesta de búsqueda semántica asimétrica en particular: ¿Cuáles son tus impresiones?

- Posibilidad de realizar una búsqueda más inexacta
- Menor desempeño en comparación a búsqueda simétrica

En cuanto a la propuesta de búsqueda por secciones: ¿Cuáles son tus impresiones?

- Útil
- Revisión rápida de secciones entre documentos
- Permite una búsqueda específica
- No recomendado con búsqueda asimétrica

Comentarios finales

- Útil
- Relevante
- Necesario
- Rápido
- Promueve colaboración entre autores
- Posible extensión a contexto temático

4.7. Conclusiones de la encuesta

En general, de las respuestas registradas, se extraen las siguientes conclusiones:

En cuanto a la búsqueda semántica en general, se percibe como un enfoque con características únicas, las cuales resultan útiles y además viables, debido a su uso intuitivo y a la rapidez de entrega de resultados.

En cuanto a la búsqueda simétrica en particular, se ve como un enfoque útil, cuyas ventajas particulares se encuentran principalmente en la búsqueda de títulos parecidos, pero no exactamente iguales, con lo cual se puede realizar una búsqueda con múltiples variaciones

semánticas, y aun así lograr encontrar resultados relevantes. También se identifica como viable por su rapidez de respuesta. Sin embargo, también se menciona la entrega de algunos resultados de baja relevancia, lo cual está relacionado con el entrenamiento de ajuste fino para el modelo, el cual no se ha realizado y se ha dejado como trabajo propuesto en este documento.

En cuanto a la búsqueda asimétrica en particular, se identifica como apropiada para entregar consultas más inexactas, es decir, entregando solo palabras o conceptos clave, o una pregunta formulada. Sin embargo, se identifica un menor desempeño en comparación a la búsqueda semántica. Esto puede deberse, principalmente, a que se trata de un enfoque de menor precisión, y que el modelo utilizado para esta búsqueda no fue entrenado en un set de datos en español, de modo que es menos preciso para nuestro corpus de búsqueda.

En cuanto a la búsqueda por secciones, se identifica como una herramienta útil para realizar una búsqueda más específica dentro del contenido de cada documento, debido a la búsqueda transversal que se realiza sobre cada una de las secciones de cada documento, para entregar un resultado de alta relevancia. Sin embargo, se comenta una baja precisión en búsqueda asimétrica.

Finalmente, con respecto a los comentarios finales, se destaca: la relevancia de la implementación de este enfoque, debido a su utilidad identificada en diferentes aspectos de su implementación y a la importancia del repositorio. Particularmente útil, debido a sus características únicas. La posibilidad de promover el trabajo y la referencia cruzada entre autores de la institución, debido al posible descubrimiento de trabajos que podrían haber sido pasados por alto con métodos diferentes. Finalmente, su viabilidad de uso, debido a su rápido tiempo de respuesta y a lo intuitivo de las consultas generadas. Una posible área de mejora mencionada es la posibilidad de reducir la búsqueda a un contexto temático en particular, considerando, por ejemplo, uno o varios departamentos de la universidad, una o varias áreas de investigación particulares, uno o varios temas tratados en cada documento, etc. Esto se deja, precisamente, como un trabajo propuesto en este documento.

CAPÍTULO 5

CONCLUSIONES

En esta sección, se exponen las conclusiones principales del trabajo realizado, considerando los siguientes puntos principales:

- Planteamiento inicial del problema y estado del arte
- Propuesta basada en modelos actuales
- Análisis, obtención y procesamiento del corpus
- Entrenamiento y comparación de modelos
- Ejemplo aplicación de clustering
- Armado del sistema de prueba
- Evaluación del sistema de prueba

El planteamiento inicial de la propuesta comenzó considerando dos ideas principales: el manejo de una gran cantidad de documentos escritos en lenguaje natural, y la utilización de herramientas de procesamiento de lenguaje natural para poder llevar a cabo esta tarea. Se realizó la síntesis de ambas ideas con el planteamiento de un trabajo de diseño e implementación de un sistema de búsqueda semántica; un sistema de búsqueda que considera el significado semántico tanto de la consulta del usuario, como de los documentos sobre los que se realiza esta búsqueda.

En cuanto al de manejo de grandes cantidades de documentos, se cree que el sistema de búsqueda propuesto en esta memoria corresponde a una solución adecuada. Este sistema no solo considera el título y el resumen de cada documento, sino que también considera el texto contenido dentro de cada documento, con la posibilidad de realizar una búsqueda individual sobre cada una de sus secciones. Así, el sistema presenta una solución al manejo de esta gran cantidad de documentos, ya que tiene la capacidad de realizar una búsqueda sobre el corpus completo, y sobre cada sección de cada documento perteneciente al corpus, considerando una representación semántica de cada una de estas secciones.

Por otro lado, en relación al uso de herramientas de lenguaje natural para realizar esta búsqueda, la solución propuesta lo hace satisfactoriamente. Durante el desarrollo de este trabajo, se ha realizado un estudio en profundidad del estado del arte de estas herramientas, considerando métodos actuales y áreas de trabajo. Con el avance de esta investigación, se ha convergido lentamente hacia el marco teórico propuesto en este trabajo. Así, se ha construido un sistema de búsqueda sobre documentos y sus secciones, que se basa en las representaciones semánticas de cada uno de estos elementos, obtenidas con el uso del modelo

SBERT, y gracias a ello somos capaces de buscar entre los textos y su contenido de una manera directa y con una mayor sensibilidad al contexto.

Así, se cree que esta investigación ha convergido hacia un marco teórico adecuado para dar solución al problema inicial. A pesar de no ser la única manera de utilizar este tipo de herramientas para resolver este problema, se desea destacar la importancia de experimentar con esta solución en particular, considerando sus ventajas y desventajas, las cuales serán detalladas más adelante.

En cuanto al corpus de documentos: se logró establecer contacto con los administradores del repositorio, quienes nos facilitaron los datos necesarios sobre el Repositorio USM para realizar la construcción del sistema. En cuanto a la extracción de los datos principales de cada documento, es decir, el título y *abstract* (descripción) de cada uno, no se presentaron mayores dificultades, pudiendo acceder directamente a ellos desde los datos facilitados por los administradores del repositorio. Sin embargo, en cuanto al contenido de los documentos en formato PDF, se tienen varias conclusiones.

Sobre estos documentos, al tratarse de datos no estructurados, se debió realizar un proceso de extracción y estructuración de estos, el cual, luego de avanzada una parte del trabajo, se identificó como un problema complejo, y de difícil aplicación a gran escala para la totalidad del corpus. Debido a esto, se concluye sobre las grandes ventajas que significaría el contar con un formato estructurado para el contenido de cada documento, y no solo como un documento PDF no estructurado. Si bien esto significaría un esfuerzo por estructurar y estandarizar el formato de los trabajos desarrollados por los alumnos a través de cada departamento de la universidad, se cree que esto no solo sería de gran utilidad para el tratamiento del contenido de estos documentos en la construcción del sistema propuesto, sino que también para otras posibles aplicaciones que realicen procesamiento de lenguaje natural.

Otra posible solución para la extracción del contenido de estos documentos digitalizados sería el desarrollo de una herramienta para este propósito. Sin embargo, como se ha mencionado, durante la realización de este trabajo, se ha observado que esta es una tarea compleja, la cual se escapa totalmente del alcance de este trabajo, de modo que aquí se insiste en una solución a nivel de documento, elaborada por la institución.

Con respecto al modelo: en general, se ha observado una gran disponibilidad de modelos pre-entrenados para tareas específicas, y sobre grandes conjuntos de datos de diferentes ámbitos, para una gran cantidad de idiomas diferentes. Sin embargo, durante la realización de este trabajo, se ha observado que la mayor parte de modelos y herramientas disponibles para el procesamiento de textos en lenguaje natural se han desarrollado para el idioma inglés. De esta forma, las principales herramientas de optimización y análisis de modelos utilizados durante el desarrollo de este trabajo han sido implementadas pensando principalmente en documentos en inglés. Así, se propone como trabajo futuro el ajuste fino del modelo utilizado en este sistema para adaptarlo al idioma del corpus utilizado, con el objetivo de mejorar la precisión de búsqueda del sistema. Esto puede ser realizado, por ejemplo, utilizando modelos generativos de lenguaje para recibir como entrada el contenido de los

documentos del repositorio, y entregando como salida textos escritos en este mismo lenguaje. Luego, utilizar estos textos para realizar un ajuste fino del modelo escogido. El detalle de cómo desarrollar este ajuste fino puede encontrarse en la documentación del módulo SentenceTransformers.

Ahora, en cuanto al *clustering* obtenido con el uso de BERTopic, esto queda como una prueba del uso que puede darse a las representaciones en vector de los textos para realizar un análisis del contexto. Este enfoque abre la puerta a nuevas investigaciones sobre el uso de estas representaciones para realizar una caracterización de los datos, de lo cual existen pocos antecedentes aún. Al igual que con el enfoque de búsqueda semántica, el uso de estas representaciones para obtener un *clustering* de los datos no representa un reemplazo a métodos tradicionales, sino más bien un apoyo o un nuevo enfoque en el análisis de corpus de documentos de gran tamaño.

Ahora, con respecto a la herramienta, para esta propuesta se ha desarrollado un sistema de interfaz simple, que incluye las funciones principales de la búsqueda permitida por los modelos. Se cree suficiente para la demostración y evaluación de esta propuesta, pero, en el caso de aplicar esta herramienta a un caso real, es necesario expandir su desarrollo. Se propone como un posible trabajo futuro.

Con respecto a las encuestas realizadas, se obtuvo una opinión general sobre la viabilidad y características de la herramienta. Sin embargo, se hubiera deseado contar con un mayor número de participantes, para obtener una opinión más amplia sobre la propuesta.

Ahora, volvemos a revisar los objetivos planteados al comienzo de esta propuesta, para analizar si se han cumplido satisfactoriamente en relación con el trabajo desarrollado:

5.1. Objetivo general

- Diseñar e implementar un sistema de búsqueda semántica sobre la colección disponible en el Repositorio USM para mejorar la calidad y precisión de consulta del usuario.

En el tercer capítulo de este trabajo, se expone en detalle el diseño del sistema. En el cuarto capítulo, se expone en detalle su implementación.

Se hace uso de búsqueda semántica, puesto que se trabaja dentro del marco teórico de la recuperación de información, utilizando herramientas de procesamiento de lenguaje natural. En particular, se utiliza el modelo BERT, y su variante SBERT, para el diseño e implementación del sistema.

Se utilizan los documentos disponibles en el Repositorio USM, tras ser solicitados y obtenidos de parte de los administradores del repositorio. Se obtuvo una parte de los documentos disponibles; en particular, los documentos de acceso abierto desarrollados por alumnos de

pregrado de la universidad.

Se mejora la calidad de la consulta del usuario, considerando que ahora es capaz de utilizar un nuevo enfoque para la búsqueda, y se mejora la precisión de su consulta, debido a que ahora es capaz de precisar el contexto semántico de su consulta, lo cual no era posible de realizar con la implementación actual del sistema.

5.2. Objetivos específicos

- Extracción de documentos disponibles en el Repositorio USM y preprocesamiento para su uso durante la etapa de diseño e implementación del sistema.

Para la extracción de documentos, se ha contactado al personal responsable de la administración del repositorio, los cuales han facilitado una muestra del corpus del repositorio para la construcción del sistema.

El trabajo de preprocesamiento de los documentos ha sido descrito en los capítulos 3 y 4.

- Diseño de un sistema que utilice herramientas del estado del arte de recuperación de información tales como modelos basados en aprendizaje profundo (deep learning) y/o técnicas dispersas de recuperación semántica (sparse techniques) para ser utilizado en la etapa de implementación.

Se ha llevado a cabo una investigación sobre las herramientas del estado del arte de recuperación de información, en particular, sobre el modelo SBERT, para su uso en búsqueda semántica, que corresponde a un modelo basado en aprendizaje profundo (deep learning).

- Implementación del sistema mediante interfaz de usuario y parámetros de configuración de la búsqueda para poder ser utilizado por usuarios.

Se ha implementado el sistema como una aplicación web desarrollada utilizando el *framework* Flask, en Python. Este sistema permite el establecimiento de parámetros de configuración para la búsqueda, tales como el tipo de búsqueda, el tipo de documento a buscar, el departamento de cada documento, y la cantidad de documentos a buscar.

- Evaluación de viabilidad del sistema a través de pruebas con usuarios que entreguen una opinión sobre el sistema implementado.

Se ha desarrollado y aplicado una encuesta de opinión a usuarios pertenecientes a la comunidad universitaria y a posibles externos que puedan tener interés en hacer uso del repositorio. En esta encuesta, se ha obtenido una opinión subjetiva sobre la experiencia de cada usuario con respecto a diferentes aspectos del sistema, considerando posibles puntos positivos y negativos.

5.3. Conclusiones finales

El alcance principal del trabajo realizado en esta memoria es la utilización de herramientas basadas en grandes modelos de lenguaje, específicamente en BERT, para tareas de búsqueda. En particular, se pueden utilizar estas herramientas para obtener representaciones de los metadatos de documentos, tales como título y descripción (*abstract*), y utilizarlas tanto para tareas de búsqueda, principalmente, como para otras tareas de caracterización de los datos, como se ha mostrado en el ejemplo de clusterización con BERTopic. No solo esto, sino que se también se ha buscado extender el uso de estas herramientas al contenido de los archivos digitalizados de cada documento, proponiendo y haciendo uso de una estrategia de particionamiento basada en las secciones de cada documento. Previo al desarrollo de esta propuesta, no se han encontrado trabajos que hiciesen uso de esta u otra estrategia similar para realizar búsqueda semántica sobre los textos que componen estos archivos.

Por otro lado, la limitación principal de esta propuesta consiste en la metodología utilizada para realizar la búsqueda semántica: esta herramienta considera la similitud entre el vector del término de búsqueda y el vector del párrafo o sentencia dentro del corpus para dar un puntaje a la búsqueda. Sin embargo, quizás sea posible implementar una estrategia diferente para hacer uso de estas representaciones, la cual puede servir para realizar una búsqueda en otro contexto.

Además, si bien se ha realizado un trabajo inicial para el particionamiento del contenido de los documentos del corpus, esto solo se ha realizado para una baja cantidad de ellos, debido a la dificultad de estructurar el contenido de cada uno de estos documentos de forma manual, trabajo que ha quedado propuesto y que completaría el análisis para el corpus completo. Finalmente, si bien se ha utilizado un modelo con un rendimiento aceptable para este caso de uso, es posible mejorar la precisión de este modelo realizando un ajuste fino basado en el contenido particular de los documentos del corpus del Repositorio USM, de modo que esto también ha quedado como un posible trabajo futuro.

Por último, el autor desea expresar su gran interés por los avances recientes en el área de procesamiento de lenguaje natural utilizando modelos de lenguaje de gran tamaño, e insiste en el desarrollo de nuevas herramientas capaces de captar no solo la estructura sintáctica del lenguaje natural, sino que también su estructura semántica, con el objetivo de apoyar en las diversas áreas que requieren del procesamiento y manejo de la información disponible en grandes volúmenes de documentos escritos en lenguaje natural.

El autor espera que este trabajo pueda servir como un mínimo acercamiento y motivación de las capacidades de los modelos de lenguaje del estado del arte, y, en particular a una de sus aplicaciones más interesantes, la búsqueda semántica, la cual ha tratado de exponerse mediante un ejemplo práctico en este documento.

REFERENCIAS BIBLIOGRÁFICAS

- [Baeza-Yates et al., 1999] Baeza-Yates, Ricardo and Ribeiro-Neto, Berthier and others (1999). *Modern information retrieval*, volumen 463. ACM press New York.
- [Bast et al., 2016] Bast, Hannah and Buchhold, Björn and Haussmann, Elmar and others (2016). Semantic search on text and knowledge bases. *Foundations and Trends® in Information Retrieval*, 10(2-3):119–271.
- [Blei et al., 2003] Blei, D. M., Ng, A. Y., y Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022.
- [Deerwester et al., 1990] Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., y Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407.
- [Grootendorst, 2022] Grootendorst, M. R. (2022). Bertopic: neural topic modeling with a class-based tf-idf procedure. *ArXiv*, abs/2203.05794.
- [Hambarde y Proença, 2023] Hambarde, K. A. y Proença, H. (2023). Information retrieval: recent advances and beyond. *IEEE Access*, 11:76581–76604.
- [Huang et al., 2013] Huang, P.-S., He, X., Gao, J., Deng, L., Acero, A., y Heck, L. (2013). Learning deep structured semantic models for web search using clickthrough data. En *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pp. 2333–2338.
- [Kenton y Toutanova, 2019] Kenton, J. D. M.-W. C. y Toutanova, L. K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. En *Proceedings of naacl-HLT*, volumen 1, p. 2. Minneapolis, Minnesota.
- [Luk, 2022] Luk, R. W. (2022). Why is information retrieval a scientific discipline? *Foundations of Science*, pp. 1–27.
- [Mitra et al., 2017] Mitra, B., Diaz, F., y Craswell, N. (2017). Learning to match using local and distributed representations of text for web search. En *Proceedings of the 26th international conference on world wide web*, pp. 1291–1299.
- [Naveed et al., 2023] Naveed, H., Khan, A. U., Qiu, S., Saqib, M., Anwar, S., Usman, M., Barnes, N., y Mian, A. S. (2023). A comprehensive overview of large language models. *ArXiv*, abs/2307.06435.
- [Reimers y Gurevych, 2019] Reimers, N. y Gurevych, I. (2019). Sentence-bert: sentence embeddings using siamese bert-networks. En *Conference on Empirical Methods in Natural Language Processing*.

- [Salton *et al.*, 1975] Salton, G., Wong, A., y Yang, C.-S. (1975). A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620.
- [Schütze *et al.*, 2008] Schütze, H., Manning, C. D., y Raghavan, P. (2008). *Introduction to information retrieval*, volumen 39. Cambridge University Press Cambridge.
- [Shen *et al.*, 2014] Shen, Y., He, X., Gao, J., Deng, L., y Mesnil, G. (2014). A latent semantic model with convolutional-pooling structure for information retrieval. En *Proceedings of the 23rd ACM international conference on conference on information and knowledge management*, pp. 101–110.
- [Vaswani *et al.*, 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., y Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.