



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA



UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Búsqueda de Respuestas utilizando redes neuronales

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Salyuk Kulinich, Mykola

Tutor/a: Hurtado Oliver, Lluís Felip

Cotutor/a: González Barba, José Ángel

CURSO ACADÉMICO: 2021/2022

Resum

Entre les tasques més importants de la Lingüística Computacional i la Recuperació d'Informació, es troba la Recerca de Respostes o Question Answering (QA). Un sistema de QA ha de ser capaç de respondre a les preguntes realitzades pels usuaris en llenguatge natural buscant les respostes en un repositori de documents o base de coneixements.

En els sistemes de QA clàssics es poden identificar tres etapes: 1) Anàlisi de la Pregunta, per identificar el “què”, “on” o “qui” pregunta l'usuari; 2) Recuperació dels Passatges on pugui estar la resposta; 3) Extracció de la resposta típicament utilitzant plantilles sobre els passatges recuperats. La irrupció en els darrers anys dels models de llenguatge neuronal ha capgirat el sistema clàssic, permetent la proliferació d'algoritmes de comprensió lectora i tècniques que integren l'ús de representacions vectorials denses per captar la semàntica lingüística de la millor forma possible. Els sistemes de Recerca de Respostes neuronals són generalment l'estat de l'art per l'idioma anglès, excloent l'espanyol i el català del panorama científic.

En el present Treball Final de Grau es proposa l'estudi dels models de llenguatge neuronal de l'estat de l'art per a construir un sistema de Recerca de Respostes en castellà i català. Com a base de coneixement s'utilitza la Wikipedia. El sistema realitza un anàlisi bàsic de la pregunta i, amb l'ajuda de llenguatge neuronal, extrau la resposta a partir de fragments de text que probablement la continguin.

Paraules clau: Recerca de Respostes, Processament del llenguatge natural, Recuperació d'informació, Aprenentatge Profund, Extracció de respostes

Resumen

Entre las tareas más interesantes de la Lingüística Computacional y la Recuperación de Información se encuentra la Búsqueda de Respuestas o Question Answering (QA). Un sistema de QA debe ser capaz de responder a preguntas realizadas por los usuarios en lenguaje natural buscando las respuestas en un repositorio de documentos o base de conocimiento.

En los sistemas de QA clásicos se pueden identificar tres etapas: 1) Análisis de la Pregunta, para identificar sobre “qué”, “dónde” o “quién” pregunta el usuario; 2) Recuperación de Pasajes en los que puede estar la respuesta; 3) Extracción de la Respuesta típicamente utilizando plantillas sobre los pasajes recuperados. La irrupción en los últimos años de los modelos de lenguaje neuronales le ha dado un vuelco al esquema clásico, permitiendo así la proliferación de algoritmos de comprensión lectora y técnicas que integran el uso de representaciones vectoriales densas para captar de la mejor manera posible la semántica lingüística. Los sistemas de Búsqueda de Respuestas neuronales son generalmente el estado del arte para el idioma inglés, excluyendo al español y catalán del panorama científico.

En el presente Trabajo Final de Grado se propone el estudio de los modelos de lenguaje neuronales del estado del arte para construir un sistema de búsqueda de respuestas en español y catalán. Como base de conocimiento se utiliza la Wikipedia. El sistema realiza un análisis básico de la pregunta y con la ayuda de modelos de lenguaje neuronal extrae la respuesta a partir de fragmentos de texto como posibles candidatos a contenerla.

Palabras clave: Búsqueda de Respuestas, Procesamiento del lenguaje natural, Recuperación de información, Aprendizaje Profundo, Extracción de respuestas

Abstract

Question Answering (QA) is one of the most interesting tasks in Computational Linguistics and Information Retrieval. A QA system must be able to answer questions asked by users in natural language by searching for answers in a document repository or knowledge base.

In classical QA systems, three stages can be identified: 1) Question Analysis, to identify "what", "where" or "who" the user is asking about; 2) Retrieval of passages in which the answer can be found; 3) Extraction of the answer, typically using rule-based models on the retrieved passages. The irruption in recent years of neural language models has turned the classical scheme on its head allowing the development of Machine Reading Comprehension algorithms and techniques that integrate dense vector representations for capturing in the best possible way linguistic semantics. Question Answering systems are generally the state of art for English language, excluding Spanish and Catalan from this approach.

In the present project we propose the study of the state-of-the-art of neural language models for building a basic Question Answering system in Spanish and Catalan. Wikipedia will be used as a knowledge base. The system will perform a basic analysis of the question and with the help of neural language models extracts the answer from fragments of text as possible candidates containing it.

Key words: *Question Answering, Natural Language Process, Information Retrieval, Deep Learning, Text Extraction*

Índice general

Índice general	VII
Índice de figuras	IX
Índice de tablas	X
<hr/>	
1 Introducción	1
1.1 Motivación	1
1.2 Objetivos	2
2 Introducción a los sistemas de búsqueda de respuestas	3
2.1 Dominio del sistema	3
2.2 Categorización de la pregunta	3
2.3 Etapas de la búsqueda de respuestas	4
2.3.1 Análisis de la pregunta	5
2.3.2 Recuperación de documentos	6
2.3.3 Extracción de la respuesta	6
3 Estado del arte y fundamentos teóricos	9
3.1 Representación numérica del lenguaje	9
3.1.1 Modelo de espacio vectorial	10
3.1.2 <i>Word Embeddings</i>	11
3.2 Redes neuronales para QA	11
3.2.1 Redes Neuronales Recurrentes	11
3.2.2 LSTM (<i>Long Short-Term Memory</i>)	13
3.2.3 DrQA	16
3.2.4 BERT	18
3.2.5 RoBERTa	20
3.2.6 <i>Word embeddings</i> basados en BERT	21
4 Diseño y desarrollo de la solución	23
4.1 Arquitectura del sistema	23
4.2 Diseño detallado	24
4.2.1 Corpus	24
4.2.2 Procesamiento de la pregunta y recuperación de documentos	24
4.2.3 Procesamiento del texto	25
4.2.4 <i>Document Retriever</i>	26
4.2.5 <i>Document Reader</i>	27
4.2.6 Implementación de interfaz gráfica	28
4.3 Tecnologías utilizadas	29
4.3.1 Lenguaje de programación	29
4.3.2 Entorno de desarrollo	30
4.3.3 Paquetes y librerías	30
5 Evaluación y resultados	33
5.1 Métricas utilizadas	33
5.1.1 <i>F1-score</i>	33
5.1.2 <i>Mean Reciprocal Rank</i>	35

5.2	Evaluación aislada	35
5.2.1	Castellano	35
5.2.2	Catalán	35
5.2.3	Comparativa recalls	36
5.3	Evaluación conjunta	37
5.3.1	Castellano	37
5.3.2	Catalán	37
5.4	Tiempos	38
6	Conclusiones y trabajos futuros	41

Apéndice

A	OBJETIVOS DE DESARROLLO SOSTENIBLE	45
----------	---	-----------

Índice de figuras

2.1	Arquitectura clásica de los sistemas de búsqueda de respuestas [23].	5
2.2	Expansión de consulta mediante WordNet.	6
3.1	Estructura de una red neuronal recurrente. En la izquierda, su versión comprimida y en la derecha la versión desplegada.	12
3.2	Estado de celda [16].	13
3.3	<i>Forget gate</i>	14
3.4	<i>Input gate</i>	14
3.5	Cómputo del estado de celda.	15
3.6	<i>Output gate</i>	15
3.7	Modelo LSTM bidireccional. Las flechas verdes representan el flujo de la información hacia delante mientras que las rojas hacia atrás.	16
3.8	Arquitectura del modelo DrQA [13].	18
3.9	Arquitectura de BERT basada en el codificador del Transformer con sus dos versiones.	18
3.10	Ejemplo de MLM.	19
3.11	Fase de pre-entrenamiento y fine-tuning utilizando diferentes datasets [8].	20
3.12	Arquitectura del DPR.	22
4.1	Arquitectura del sistema.	23
4.2	Wikipedia.	24
4.3	Ejemplo de <i>Part of Speech Tagging</i>	24
4.4	Ejemplo de distribución del tamaño de los párrafos.	26
4.5	Interfaz para el español.	29
4.6	Interfaz para el catalán.	29
4.7	Pipelines de la librería SpaCy.	30
4.8	Desglose del funcionamiento de la librería Gradio.	32
5.1	Matriz de confusión.	34
5.2	Recall obtenido del document retriever para el español.	36
5.3	Recall obtenido del document retriever para el catalán.	36
5.4	Comparativa de los recalls de ambos modelos.	37
5.5	Tiempo medio de análisis y recuperación de documentos.	38
5.6	Tiempo de inferencia del <i>document retriever</i> en castellano.	39
5.7	Tiempo de inferencia del <i>document retriever</i> en catalán.	39

Índice de tablas

5.1	Ejemplo de <i>Mean Reciprocal Rank</i>	35
5.2	Resultados del <i>document retriever</i> para el español.	35
5.3	Resultados <i>document reader</i> para español.	35
5.4	Resultados <i>document retriever</i> para catalán.	36
5.5	Resultados <i>document reader</i> para catalán.	36
5.6	Resultados del <i>document reader</i> para el español.	37
5.7	Resultados <i>document reader</i> para catalán.	37
5.8	Tiempo en segundos medio de la recuperación de documentos.	38

CAPÍTULO 1

Introducción

El procesamiento del lenguaje natural es una rama de estudio cuyo objetivo es explorar cómo los ordenadores pueden ser usados para entender y manipular el lenguaje de forma escrita y hablada de la misma manera que el ser humano lo hace. Hay dos grandes enfoques al procesamiento del lenguaje natural: el enfoque computacional y el enfoque lingüístico. El enfoque computacional se centra en el desarrollo de algoritmos y técnicas de procesamiento de lenguaje natural que puedan ser utilizados por ordenadores para realizar tareas como el reconocimiento de voz o el análisis de texto. El enfoque lingüístico, por otro lado, se centra en el estudio de la estructura y el significado del lenguaje y cómo se puede representar de manera eficiente y precisa en un ordenador.

Con el paso de los años esta área ha ido experimentando numerosos avances a la vez que abarcando nuevas áreas de aplicación, siendo en la actualidad una de las más prometedoras y enriquecedoras de la informática y la inteligencia artificial. Una de las numerosas aplicaciones que está teniendo este campo es por ejemplo en el sector del marketing. Las empresas están cada vez más interesadas en analizar el sentimiento de los clientes hacia sus productos y servicios a través de las redes sociales, y el procesamiento del lenguaje natural es una herramienta invaluable para esta tarea. Gracias a él, las empresas pueden monitorizar de forma eficiente las conversaciones en línea y obtener una visión general de la opinión de los clientes. Esto les permite tomar decisiones de marketing más acertadas y, en última instancia, mejorar sus resultados.

Con la aparición de la web y su rápido crecimiento, se ha reintroducido la necesidad de crear técnicas de consulta que permitan acceder fácilmente a grandes cantidades de datos y que a su vez reduzcan el exceso de información. Dentro de estas técnicas encontramos la búsqueda de respuestas, una de las tareas más antiguas dentro del marco del procesamiento natural del lenguaje natural. Dar contestación de forma coherente y concisa a todas las preguntas que el ser humano le impone al computador no es tarea fácil, ya que involucra el reconocimiento y comprensión del lenguaje, la identificación de la pregunta y la generación de una respuesta adecuada.

1.1 Motivación

El desarrollo de los últimos años en el campo del procesamiento natural del lenguaje ha sido increíble. A partir de 2012, el uso de técnicas de aprendizaje automático con grandes volúmenes de datos ha impulsado un avance significativo en la capacidad de los ordenadores para comprender el lenguaje natural. El impacto de estos avances se ha sentido en una amplia gama de áreas, desde el reconocimiento de voz y el procesamiento de imágenes hasta la traducción automática. No obstante, un problema que radica dentro de esta disciplina y que no encontramos en otras es la gran cantidad de lenguas existentes

y por ende, las diferencias sintácticas y semánticas presentes en cada una de ellas. Esto supone un gran reto a la hora de desarrollar cualquier algoritmo capaz de comprender y expresarse en todas ellas, traduciéndose en un rendimiento menor.

1.2 Objetivos

El principal objetivo que persigue este proyecto es el estudio de modelos del estado del arte basados en redes neuronales para el posterior desarrollo de un sistema de búsqueda de respuestas en español y catalán. Mediante el uso de modelos preentrenados disponibles en los repositorios de código abierto, se implementará un sistema capaz de generar una respuesta a partir de una pregunta. Por último, se diseñará una interfaz gráfica de usuario que permita introducir preguntas por teclado.

CAPÍTULO 2

Introducción a los sistemas de búsqueda de respuestas

En este capítulo se introducen los sistemas de búsqueda de respuestas y sus diferentes matices para posteriormente detallar las diversas tecnologías utilizadas en los últimos años para su desarrollo. En primer lugar se describe el dominio del sistema y el tipo de preguntas existentes. Por último, se detallan las tres etapas clásicas de búsqueda de respuesta, que son: análisis de la pregunta, recuperación de los documentos más relevantes y extracción de la respuesta.

2.1 Dominio del sistema

La cantidad de temas sobre los que podemos hacer preguntas es desmesurada. Por ello, los sistemas de búsqueda de respuestas operan dentro de un dominio que se les proporciona. El dominio representa todo el conocimiento sobre el cuál el sistema basa sus respuestas. Podemos clasificar los sistemas en función del dominio, pudiendo ser de dominio cerrado o de dominio abierto.

Los sistemas de búsqueda de respuestas de dominio cerrado o *closed domain question answering systems* tienen un alcance limitado y centran su atención en un tema o régimen específico. Un ejemplo de este tipo de sistemas puede encontrarse en numerosas páginas web en la sección de FAQ, acrónimo de la expresión inglesa "Frequently Asked Questions", preguntas más frecuentes, dónde el usuario puede realizar preguntas relacionadas con dudas que le surgen a la hora de utilizar los servicios de cualquier empresa.

Por otro lado, los sistemas de dominio abierto o *open domain question answering systems* basan su conocimiento en proveedores de recursos, como Wikipedia o la World Wide Web, para responder preguntas de conocimiento general. El motor de búsqueda Google proporciona un sólido sistema de búsqueda de respuestas a cualquier tipo de tema.

2.2 Categorización de la pregunta

Una vez establecido el dominio del sistema, podemos analizar qué tipo de preguntas es capaz de responder. En este aspecto, es necesario enumerar los posibles formatos en los que puede presentarse una pregunta, ya que dependiendo de este, la complejidad del sistema variará considerablemente.

Preguntas factuales

Este tipo de preguntas se pueden definir como aquellas que requieren una respuesta objetiva, concreta y verificable, pudiendo ser de carácter verdadero o falso. Suelen ser lo suficientemente sencillas como para que una máquina las comprenda, y pueden construirse directamente sobre bases de datos estructurales así como extraerse directamente de textos no estructurados. Responden a aquellas preguntas que comienzan por: qué, quién, cuál, cómo, dónde, cuándo y cuánto. Un ejemplo de este tipo de preguntas sería ¿quién es el rey de España?

Preguntas de confirmación

Pueden ser respondidas en el formato de si o no. Por ejemplo, ¿es posible viajar más rápido que la luz? Estas preguntas requieren procedimientos complejos, conocimientos generales, razonamiento de sentido común y mecanismos de inferencia para dar respuestas, especialmente cuando las preguntas implican subjetividad.

Preguntas descriptivas

Suelen empezar con las palabras clave "por qué" y "cómo". La respuesta puede ser muy detallada y puede ser una sola frase, un párrafo o un documento entero. Las razones, explicaciones y elaboraciones suelen ser las respuestas a este tipo de preguntas.

Preguntas hipotéticas

Estas preguntas se utilizan a menudo en el proceso de toma de decisiones, ya que permiten explorar las consecuencias de una acción antes de que se lleve a cabo, planteando escenarios en los que se pueden dar varias respuestas. Algunos ejemplos de preguntas hipotéticas son: "¿Qué pasaría si...?", "¿Cómo sería si...?" y "¿Qué harías si...?".

Preguntas complejas

Las preguntas complejas son las que generalmente no se abordan en las otras cuatro categorías. Estas preguntas requieren procedimientos y mecanismos de inferencia complejos, y las respuestas a estas preguntas normalmente requieren sintetizar información de múltiples documentos.

2.3 Etapas de la búsqueda de respuestas

Los sistemas de búsqueda de respuestas han ido evolucionando a lo largo de los años, aumentando su complejidad y abordando de distintas maneras el problema propuesto. Sin embargo, todos ellos siguen manteniendo de una manera u otra el mismo diagrama de flujo, el cuál consiste en tres etapas: análisis de la pregunta, recuperación de documentos, extracción de la respuesta.

Dada una pregunta en lenguaje natural, el análisis de la pregunta tiene como objetivo entender primero la pregunta para facilitar la recuperación de documentos y la extracción de respuestas en las etapas siguientes. A continuación, la etapa de recuperación de documentos busca los documentos relevantes para la pregunta, basándose en un sistema de recuperación de información propio o en un motor de búsqueda web, utilizando las consultas de búsqueda generadas en el análisis de la pregunta. Por último, la extracción de respuestas se encarga de extraer las respuestas finales a las preguntas del usuario a

partir de los documentos extraídos en el paso anterior. A continuación se describe cada etapa individualmente.

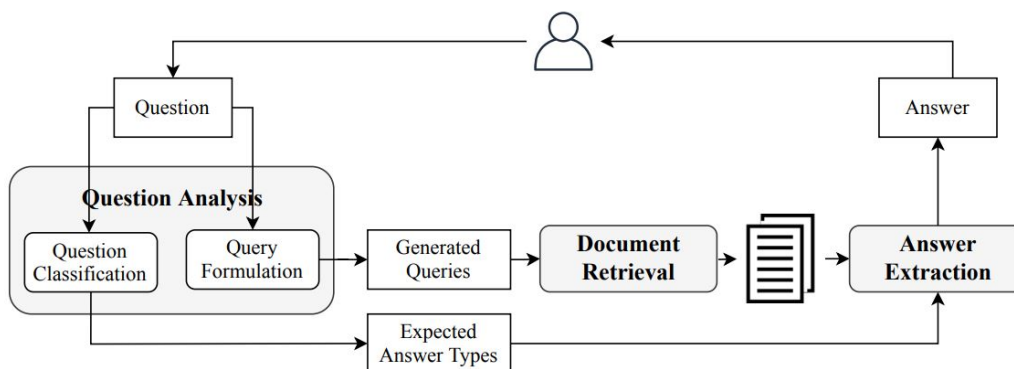


Figura 2.1: Arquitectura clásica de los sistemas de búsqueda de respuestas [23].

2.3.1. Análisis de la pregunta

El objetivo de esta primera etapa consiste en facilitar la recuperación de documentos relevantes, para lo que se suele reformular la pregunta convirtiéndola en una consulta, extrayendo aquellas palabras que consideremos relevantes y descartando aquellas que no lo son. Además, algunos sistemas incorporan en esta etapa un proceso de clasificación de la pregunta para mejorar el rendimiento en la etapa de extracción de respuestas, ya que conociendo el tipo de pregunta al que nos enfrentamos, podemos reducir el conjunto de respuestas esperadas.

Formulación de la consulta

Esta tarea consiste en crear una lista de *tokens* para enviarla a un sistema de recuperación de información con el fin de recuperar documentos que puedan contener cadenas de respuestas. Entendemos la palabra *token* como una secuencia de caracteres que constituyen una unidad semántica. Cuando el recuperador de información es un buscador web, es posible suministrar la pregunta directamente, eliminando los adverbios y pronombres relativos que la inician (dónde, cuándo, quién, etc). Cuando se trata de un conjunto reducido de documentos ofrecidos por páginas corporativas, por ejemplo Wikipedia, se utilizan técnicas lingüísticas para extraer las palabras clave a buscar. Sin embargo, a menudo los términos utilizados en las preguntas no suelen coincidir con los que aparecen en los documentos. Esto se conoce como "desajuste de términos" o *term mismatch*. Para solucionar este problema, típicamente se emplean técnicas de expansión de la consulta y paráfrasis para añadir palabras o frases de búsqueda adicionales con el fin de recuperar documentos más relevantes.

A veces es necesario validar la respuesta extraída cuando no es lo suficientemente segura antes de presentarla al usuario. Además, en algunos casos se pueden producir múltiples respuestas candidatas a una pregunta y hay que seleccionar una de ellas. La validación de respuestas se aplica para resolver estos problemas. Uno de los métodos de validación más comunes consiste en adoptar una fuente de información adicional, como un motor de búsqueda en Internet, para validar la confianza de cada respuesta candidata.

CAPÍTULO 3

Estado del arte y fundamentos teóricos

Enseñar a las máquinas a leer documentos en lenguaje natural sigue siendo un desafío difícil de resolver. Los sistemas de lectura automática evalúan su capacidad de comprensión respondiendo a preguntas sobre el contenido visualizado. TREC (*Text Retrieval Conference*) es un evento anual organizado por el NIST (*National Institute of Standards and Technology*) donde diferentes grupos de investigación se reúnen para mostrar sus modelos desarrollados para la tarea de la recuperación de información y la búsqueda de respuestas. En esta conferencia todos los modelos se evalúan utilizando una colección de documentos a recuperar y una serie de preguntas con sus debidos pasajes para luego comparar resultados y ver qué modelo obtiene los mejores resultados [22]. Con el paso de los años, los modelos expuestos en esta conferencia han ido obteniendo mejores resultados, aunque los mejores resultados alcanzados a día de hoy siguen siendo conseguidos por el ser humano.

Con la aparición de SQuAD (*Stanford Question Answering Dataset*) [17], la evaluación de los sistemas de QA se ha hecho mucho más sencilla, pudiendo compartir los modelos al mundo entero.

En el presente capítulo, se expondrán las diferentes técnicas que existen para representar la información para que los recuperadores de información sean capaces de desempeñar su correcto funcionamiento. Por otro lado, se expondrán varios modelos para la extracción de respuestas diseñados a lo largo de los años.

3.1 Representación numérica del lenguaje

El lenguaje es una facultad del ser humano para expresarse y comunicarse con los demás a través del sonido articulado o de sistemas de signos ¹. Los ordenadores no son capaces de procesar el lenguaje en las mismas modalidades que usa el ser humano. Por ello, es necesario representarlo de manera que sea legible para las máquinas. En este aspecto podemos encontrar dos vertientes bien diferenciadas a la hora de representar la información. Por un lado encontramos el modelo de espacio vectorial. Por otro lado, están las representaciones basadas en la semántica distribucional, conocidas como *word embeddings*. La idea se fundamenta en la hipótesis distribucional, la cual afirma que “elementos lingüísticos con distribuciones similares tienen significados similares”.

¹(<https://dle.rae.es/lenguaje>)

3.1.1. Modelo de espacio vectorial

Esta representación describe el contenido de un documento en términos de las frecuencias de aparición de un conjunto de palabras. Es decir, se describe el documento en términos de ocurrencias de palabras, descartando la gramática y el orden de las palabras. Este método de representación de textos es muy utilizado en problemas de clasificación y búsqueda de información, ya que es una forma muy sencilla y eficaz de representar el contenido de un documento.

El modelo de espacio se construye como un vector de d dimensiones, donde d es el número de palabras únicas en el vocabulario. Cada una de estas dimensiones contendrá la frecuencia de aparición de la palabra en la pregunta o el documento. El vocabulario está compuesto por la unión de todas las palabras presentes en la pregunta y cada documento.

Se trata de un método muy simple y por ello tiene algunas limitaciones:

- No tiene en cuenta la gramática ni el orden de las palabras, por lo que puede ser difícil interpretar el significado de un documento a partir de su representación.
- Las palabras clave de la búsqueda deben coincidir exactamente con los términos del documento.
- Sensibilidad semántica. Los documentos con un contexto similar pero un vocabulario diferente no se relacionarán.

Aunque estas limitaciones pueden ser un problema en algunos casos, este modelo sigue siendo un método muy utilizado y eficaz para representar el contenido de un documento.

TF-IDF

Tf-idf es una medida estadística que se utiliza para evaluar la importancia de una palabra en un documento. Basado en el modelo de espacio vectorial, el objetivo del modelo Tf-idf es asignar un peso a cada palabra de un documento, de tal forma que se refleje su importancia en el documento. Las palabras con un peso más alto son aquellas que son más específicas del documento y, por tanto, las que más nos ayudan a comprender el contenido del documento.

Este modelo se utiliza en los motores de búsqueda para indexar y recuperar documentos, ya que permite que los documentos sean ordenados en función de su relevancia para una consulta dada. Una encuesta realizada en 2015 mostró que el 83% de los sistemas de recomendación basados en texto en las bibliotecas digitales utilizan esta técnica.

²

Está compuesto por dos elementos: la frecuencia del término (tf) y la frecuencia inversa del documento (idf). En primer lugar, la frecuencia del término es el número de veces que aparece una palabra en el documento. Se representa utilizando la escala logarítmica, con el objetivo de minimizar la importancia de palabras con mayor frecuencia, ya que la relevancia no se incrementa proporcionalmente con su frecuencia, de lo contrario se primarían aquellos documentos de mayor talla. De esta manera, la fórmula se define como:

$$tf(t, d) = \begin{cases} 1 + \log_{10} f(t, d) & \text{si } f(t, d) > 0 \\ 0 & \text{en cualquier otro caso} \end{cases} \quad (3.1)$$

²<https://en.wikipedia.org/wiki/Tf-idf>

A continuación, la frecuencia inversa del documento para un término se define como el ratio entre $|N|/df_t$, donde N es el número de total documentos y df_t es el número de documentos en los que aparece la palabra t . Cuanto menor sea el número de documentos en los que aparece un término, mayor será la ponderación. El valor máximo, 1, se asigna a los términos que aparecen en todos los documentos, ya que $df_t \leq |N|$.

$$idf_t = \log_{10} \left(\frac{|N|}{df_t} \right) \quad (3.2)$$

Por último, el peso $tf-idf$ es calculado como el producto entre la frecuencia del término y la frecuencia inversa del documento para dicho término.

$$w_{t,d} = tf_{t,d} \cdot idf_t \quad (3.3)$$

Para el cómputo de múltiples términos en una consulta, bastaría con calcular el peso de cada uno de los términos para la consulta dada Q .

$$score(Q, d) = \sum_{t \in Q} w_{t,d} \quad (3.4)$$

Okapi BM25

BM25[12] es una versión mejorada del modelo Tf-idf. Este modelo tiene en cuenta la longitud de los documentos para ponderar el peso que tiene cada palabra dentro del documento. El cálculo de dicho peso se calcula como:

$$score(Q, d) = \sum_{t \in Q} idf_t \cdot \frac{tf_{t,d} \cdot (k_1 + 1)}{tf_{t,d} + k_1 \cdot (1 - b + b \cdot \frac{L_d}{L_{avg}})} \quad (3.5)$$

En este caso, los valores L_d y L_{avg} representan la longitud del documento y la longitud media de todos los documentos de la colección respectivamente. Por otro lado, k_1 y b son parámetros que se ajustan manualmente. Sus valores generalmente suelen oscilar entre $0,5 < b < 0,8$ y $1,2 < k_1 < 2$, aunque dichos valores dependen de las características concretas de la colección.

3.1.2. Word Embeddings

Word Embedding es una representación que atribuye a los vectores una mayor riqueza sintáctica, semántica, contextual, etc, permitiendo así poder capturar mejor el significado de las palabras y frases. Usaremos el término *embedding* para referirnos a los vectores obtenidos tras la representación.

La idea detrás de los *embeddings* es representar cada palabra como un vector de números reales, también conocido como vector denso. De esta forma, las palabras similares tendrán vectores similares y las palabras más diferentes tendrán vectores más diferentes.

3.2 Redes neuronales para QA

3.2.1. Redes Neuronales Recurrentes

Las redes neuronales recurrentes son un tipo de redes neuronales especiales donde el resultado del estado previo es proporcionado como valor de entrada al estado actual,

lo que permite crear cierta temporalidad en la red. A diferencia de las redes neuronales convencionales, donde la información se transmite solo hacia adelante entre capa y capa, las redes neuronales recurrentes permiten que la información se propague entre las neuronas de la misma capa.

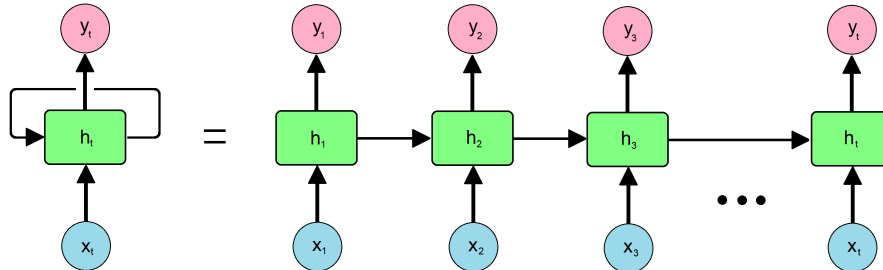


Figura 3.1: Estructura de una red neuronal recurrente. En la izquierda, su versión comprimida y en la derecha la versión desplegada.

Una red neuronal recurrente está compuesta por tres elementos: una secuencia de longitud variable x , un estado oculto h y un valor opcional de salida y . Dicho estado oculto vendrá definido por el valor de entrada x y el valor previo del estado para un instante de tiempo t de la siguiente manera:

$$h_t = f_h(x_t, h_{t-1}) \quad (3.6)$$

$$y_t = f_o(h_t) \quad (3.7)$$

Estas dos fórmulas se entienden cómo, dada una función de activación f_h para el estado oculto h_t en el instante de tiempo t , el valor de h_t se calcula mediante el cómputo del valor x_t y el valor del estado oculto h_{t-1} . El valor de salida y_t será calculado a través de la función de activación de salida f_o .

Este tipo de red es muy útil a la hora de procesar secuencias de datos en los que el orden importa, ya que la red es capaz de recordar (hasta cierto punto) el orden de entrada de los datos. Un uso muy habitual de esta estructura de ciclo es en la predicción de la siguiente palabra dada una secuencia de palabras.

Las redes neuronales recurrentes presentan ciertas desventajas. Una de ellas es la dependencia a largo plazo [3]. Supongamos que queremos predecir la última palabra dentro la oración "Pasé mi infancia en Inglaterra con mis abuelos y por eso hablo fluidamente...". La información que nos precede nos indica que la palabra que queremos encontrar es el nombre de un idioma, pero si queremos saber que idioma es, necesitamos retroceder dentro de la secuencia para encontrar la palabra Inglaterra y saber que la palabra que estamos buscando es inglés. A medida que la oración va creciendo, la distancia entre la palabra Inglaterra y la palabra que queremos predecir es mayor, lo dificulta poder predecir con exactitud de qué idioma estamos hablando, ya que no somos capaces de recordar con precisión todas las palabras.

Por este motivo surgen las redes LSTM, que se presentan en la siguiente sección.

3.2.2. LSTM (Long Short-Term Memory)

Las redes neuronales de memoria a largo y corto plazo, llamadas LSTM, son un tipo de redes neuronales recurrentes que fueron introducidas por primera vez por Hochreiter y Schmidhuber en el año 1997 [10] y resuelven el problema de la dependencia a largo plazo. Su estructura interna es un poco más compleja, ya que la característica particular que presentan es que son capaces de recordar aquella información que es relevante y olvidar lo irrelevante a través de un conjunto de “puertas” que hace que la información se preserve o se descarte en función de su importancia.

Los elementos que componen una red LSTM dado un instante de tiempo t son:

- Un valor de entrada x_t
- Un estado de la celda C_t
- Una forget gate f_t
- Una input gate i_t
- Una output gate o_t
- Un valor de salida h_t , que es el estado oculto (siguiendo la misma nomenclatura que en el apartado anterior)

El estado de la celda, o cell state, es un componente vital dentro de la red LSTM. Alberga gran parte de la información que será transmitida entre cada iteración. Esta información es alterada únicamente a través de interacciones lineales, por lo que no sufre grandes cambios en cada iteración, lo que hace que información del pasado se preserve de mejor.

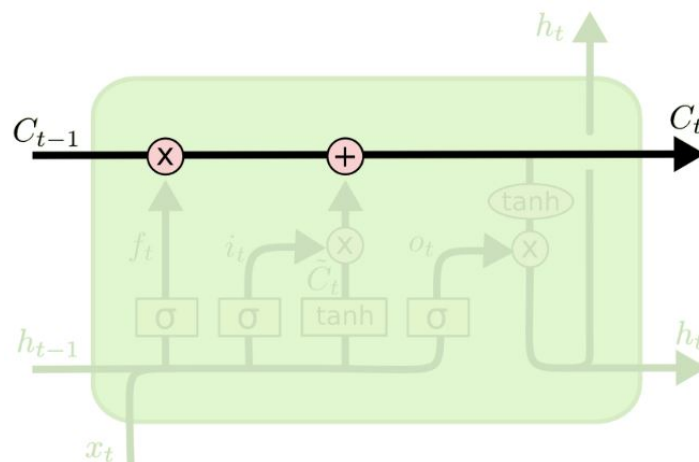


Figura 3.2: Estado de celda [16].

La *forget gate* es una “puerta” encargada de olvidar la información intrascendente de la iteración anterior. Esta “puerta” no es más que una función de activación de tipo sigmoidea que procesa la información del estado oculto h_{t-1} y la información procedente del valor de entrada x_t . La razón de utilizar una función de activación de tipo sigmoidea es que permite generar valores comprendidos entre 0 y 1, los cuáles se asignan a cada elemento del estado oculto h_{t-1} , haciendo que olvide aquella información multiplicada por 0 y que recuerde el resto en menor o mayor medida.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad (3.8)$$

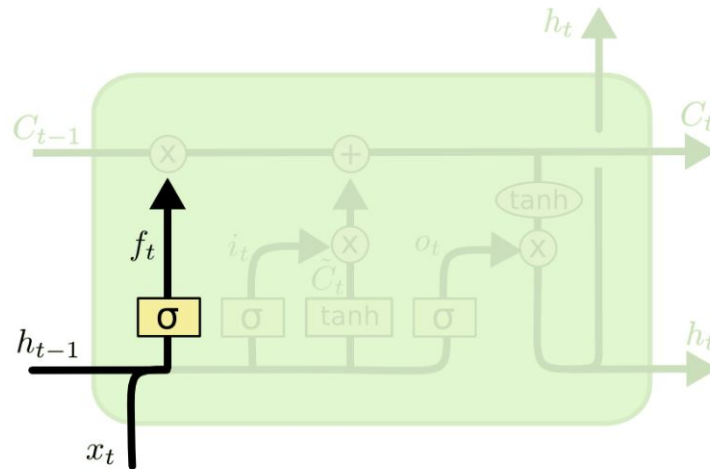


Figura 3.3: Forget gate.

La *input gate* es responsable de elegir la nueva información procedente del valor de entrada x_t que vayamos a preservar, utilizando otra función de activación de tipo sigmoidea. Tras elegir los candidatos, es necesario actualizar los valores del estado de celda, lo cual se hace gracias al vector \tilde{C}_t a través de una función de activación de tipo tangente hiperbólica, que permite que la información se añada al estado de celda en el formato correcto.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (3.9)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \quad (3.10)$$

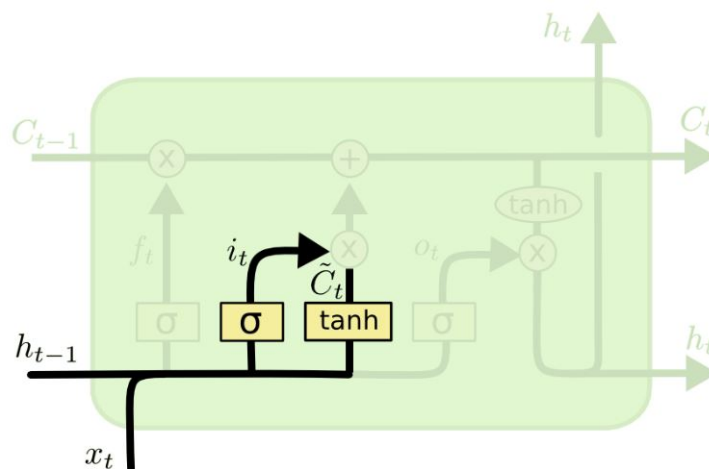


Figura 3.4: Input gate.

Tras procesar la información proveniente del estado de la celda t_1 y del valor de entrada x_t , es necesario actualizar el estado de la celda. Como hemos mencionado antes, el

estado de la celda se calcula a través de operaciones lineales, como la suma y la multiplicación, por lo que la información prácticamente no se altera al actualizar los valores del estado de celda, ya que ha sido modificada al pasar por las *forget gate* e *input gate*. El valor del estado de celda se calculará como:

$$C_t = f_t \times C_{t-1} + i_t \times \tilde{C}_t \quad (3.11)$$

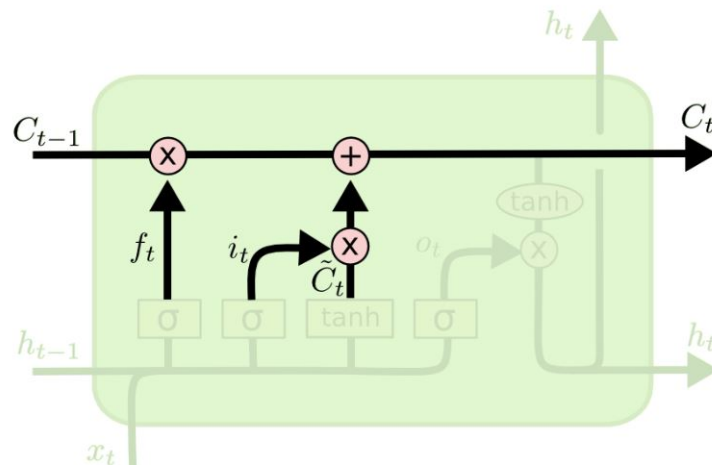


Figura 3.5: Cómputo del estado de celda.

Por último, es necesario calcular el valor del estado oculto h_t . Este valor es el resultado de la información procedente del estado de la celda C_t (que ha sido actualizado), la información del estado oculto h_{t-1} y el valor de entrada x_t . Estos dos últimos parámetros son procesados a través de la *output gate* utilizando una función de activación de tipo sigmoide para luego ser multiplicados por el valor del estado de la celda.

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad (3.12)$$

$$h_t = o_t \times \tanh(C_t) \quad (3.13)$$

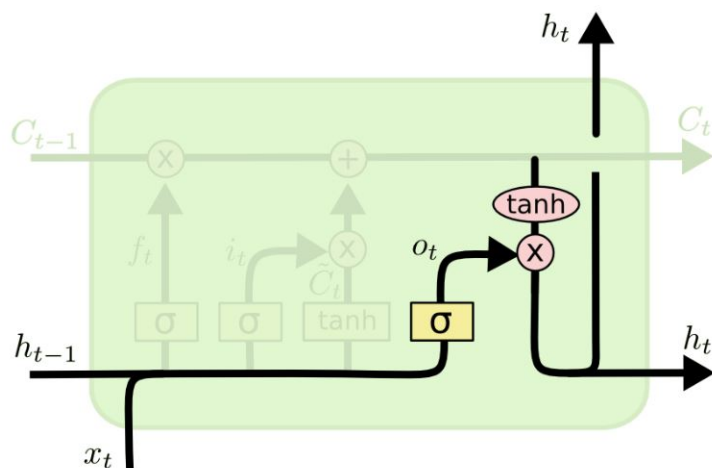


Figura 3.6: Output gate.

Las redes *LSTM* consiguen solucionar el problema de las dependencias a largo plazo. Sin embargo, otro problema que surge a la hora de utilizar las redes neuronales recurrentes y las redes *LSTM* es la unidireccionalidad. Supongamos que queremos predecir la palabra oculta dentro de la sentencia: "Me gustan mucho los ratones. Utilizo mi ratón para ... por el ordenador". Leyendo toda la oración podemos entender que la palabra que estamos buscando es "desplazarme". Esta capacidad que tenemos de entender el contexto de una oración es algo que las redes neuronales recurrentes no son capaces de hacer, ya que la información se procesa de forma unidireccional de izquierda a derecha y es necesario leer toda la secuencia para entender que nos referimos a un periférico del ordenador y no a un animal. Por ello, las redes neuronales recurrentes bidireccionales surgen para solucionar este problema.

Bidirectional LSTM

Las redes neuronales de memoria a largo y corto plazo bidireccionales son la solución al problema de la unidireccionalidad. Tienen la misma estructura que las redes neuronales recurrentes bidireccionales, que fueron introducidas por Mike Schuster y Kuldip Paliwal [19], salvo que en vez de utilizar neuronas recurrentes, estas se sustituyen por neuronas *LSTM*. Esta red está compuesta por dos redes neuronales *LSTM*: una recorre la secuencia de entrada hacia adelante (de izquierda a derecha) y la otra hacia atrás (de derecha a izquierda), como se muestra en la figura 3.7. Este tipo de arquitectura se entrena prácticamente de la misma manera que una red *LSTM*, ya que no hay interacción entre las dos redes, por lo que la información pasa directamente la capa de salida.

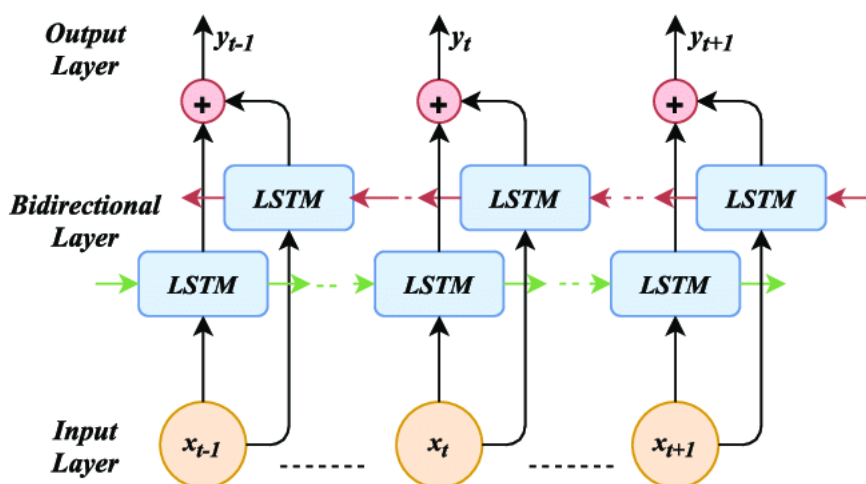


Figura 3.7: Modelo *LSTM* bidireccional. Las flechas verdes representan el flujo de la información hacia adelante mientras que las rojas hacia atrás.

3.2.3. DrQA

DrQA [7] es un sistema de búsqueda de respuestas basado en las redes LSTM bidireccionales. Dada una pregunta q compuesta por l tokens q_1, \dots, q_l y un párrafo p compuesto por m tokens p_1, \dots, p_m , el objetivo de este modelo es predecir para cada token p_i la probabilidad $p_{start}(i)$ de que p_i sea el inicio de la respuesta y la probabilidad $p_{end}(i)$ de ser el final de la respuesta.

Este modelo representa cada token de la pregunta y del párrafo mediante *embeddings*, calcula la similitud entre la pregunta y el párrafo a través de una función de similitud y utiliza dicho valor para predecir el inicio y fin de la respuesta.

Tras representar cada token de la pregunta $E(q_1), \dots, E(q_l)$, estos son tomados como input por la red LSTM bidireccional para calcular la suma ponderada convirtiendo estos *embeddings* en un único vector q .

$$q = \sum_j b_j q_j \quad (3.14)$$

El valor b_j es una medida de relevancia de cada token de la pregunta y se basa en un vector de pesos aprendido w :

$$b_j = \frac{\exp(w \cdot q_j)}{\sum_{j'} \exp(w \cdot q_{j'})} \quad (3.15)$$

Para el obtener los *embeddings* de los diferentes pasajes $\{p_1, \dots, p_m\}$, se hace uso de una representación $\tilde{p} = \{\tilde{p}_1, \dots, \tilde{p}_m\}$ teniendo en cuenta cuatro elementos:

- Un *embedding* $E(p_i)$ para cada token del pasaje, utilizando la técnica GloVe.
- Entidades reconocidas para cada token p_i usando las técnicas de POS o NER.
- Coincidencia exacta del token p_i en la pregunta $q \mathbb{1}(p_i \in q)$
- Además de la coincidencia exacta entre token del pasaje y token de la pregunta, se utiliza un mecanismo de atención para encontrar aquellas palabras que tienen un significado semántico parecido. Se usa una similitud ponderada $\sum_j a_{i,j} E(q_j)$, donde $a_{i,j}$ es el vector de atención que codifica la similitud entre p_i y cada token de la pregunta q_j . Este vector de atención se puede calcular como el producto escalar entre dos funciones α de los *embeddings* de la pregunta y el pasaje:

$$a_{i,j} = \frac{\exp(\alpha(E(p_i)) \cdot \alpha(E(q_j)))}{\sum_{j'} \exp(\alpha(E(p_i)) \cdot \alpha(E(q'_j)))} \quad (3.16)$$

$\alpha(\cdot)$ se entiende como un red neuronal.

Tras obtener los *embeddings* de los pasajes, estos se suministras a la red LSTM bidireccional.

$$\{p_1, \dots, p_m\} = RNN(\tilde{p}_1, \dots, \tilde{p}_m) \quad (3.17)$$

El resultado final es un *embedding* para la pregunta q y uno para cada palabra dentro del pasaje. Con el fin de predecir dónde se encuentra la respuesta, se entrenan dos redes LSTM por separado, una para calcular la posición de inicio de la respuesta y otra para el final. Finalmente, el cálculo de las posiciones se hace a través de una capa de atención bilineal W .

$$\begin{aligned} P_{start}(i) &\propto \exp(p_i W_{start} q) \\ P_{end}(i) &\propto \exp(p_i W_{end} q) \end{aligned} \quad (3.18)$$

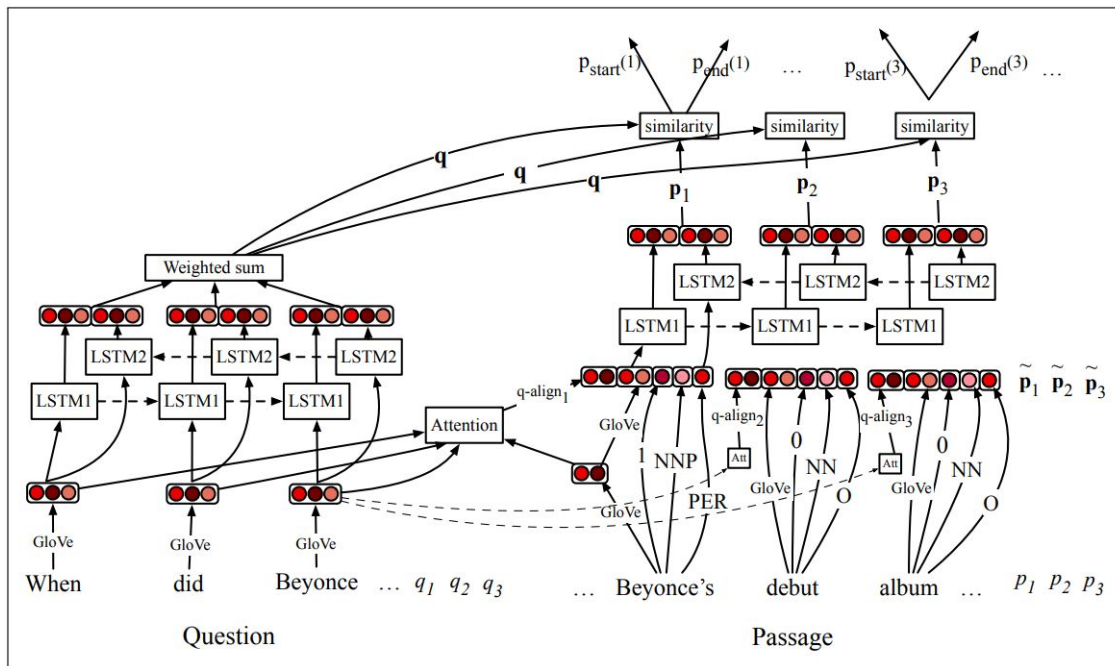


Figura 3.8: Arquitectura del modelo DrQA [13].

3.2.4. BERT

Procedente de las siglas *Bidirectional Encoder from Transformers Representations*, BERT es un modelo de lenguaje basado en la arquitectura Transformer [21] introducido por primera vez en el año 2018 por el grupo de investigación de Google [8]. Compuesto únicamente por bloques del codificador del Transformer, lo que hace a BERT tan especial es la capacidad de procesar el texto en ambos sentidos simultáneamente, proporcionándole así bidireccionalidad, algo que las redes neuronales recurrentes no son capaces de hacer, ya que estas lo hacen de izquierda a derecha o viceversa, pero no al mismo tiempo,

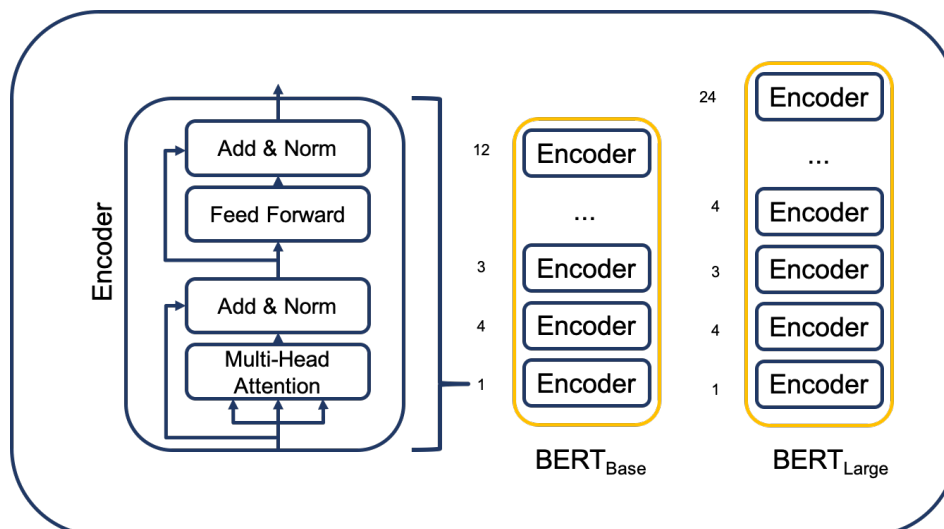


Figura 3.9: Arquitectura de BERT basada en el codificador del Transformer con sus dos versiones. URL: humboldt-wi.github.io/blog/research/information_systems_1920/bert_blog_post/

Este modelo ha sido pre-entrenado mediante dos tareas no-supervisadas: *Masked Language Model* (MLM) y *Next Sentence Prediction* (NSP). Tras esta fase de pre-entrenamiento,

el modelo es capaz de interpretar el lenguaje y pasa por una última fase de afinado o *Fine-tuning* donde es entrenado de nuevo usando los mismos parámetros de la fase de pre-entrenamiento lo que le permite al modelo desempeñar diferentes tareas.

Masked Language Model

Esta primera tarea de entrenamiento consiste en enmascarar el 15 % de los tokens seleccionados aleatoriamente y remplazarlos por la etiqueta **[MASK]**. El modelo intenta entonces predecir el valor original de la palabra enmascarada en función de las palabras que se encuentran a su alrededor. Mediante esta técnica, el modelo es condicionado a comprender el contexto de la frase, proporcionándole así bidireccionalidad.

Aunque esto permite obtener un modelo bidireccional preentrenado, un inconveniente se está creando un desajuste entre la fase de pre-entrenamiento y *Fine-tuning*, ya que durante esta última fase el token **[MASK]** no aparece. Con el fin de evitar dicho problema, los tokens "enmascarados" no siempre se sustituyen por **[MASK]**. Dentro del 15 % de los tokens "enmascarados", realmente el 80 % es enmascarado, el 10 % se sustituye por palabras aleatorias y el 10 % restante no es modificado.

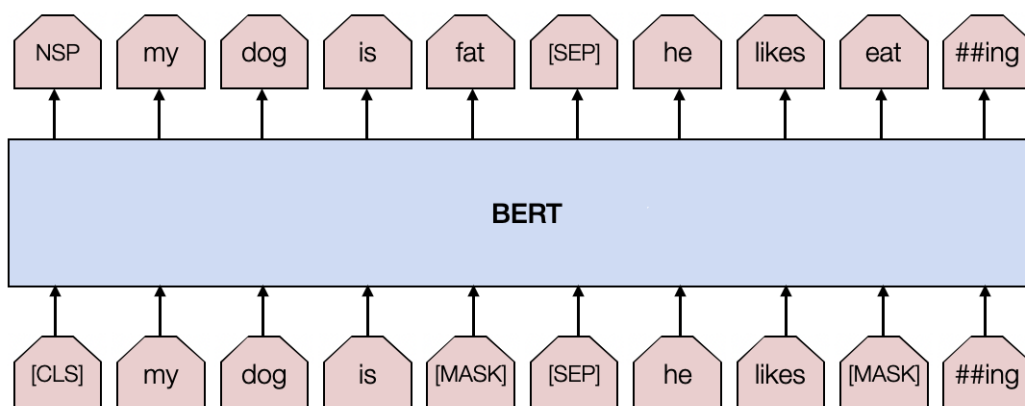


Figura 3.10: Ejemplo de MLM.

URL: <https://medium.com/dair-ai/a-light-introduction-to-bert-2da54f96b68c>

Next Sentence Prediction

Tras predecir la palabra enmascarada, la siguiente tarea con la que es entrenado el modelo BERT es NSP. Como se muestra en la figura 3.11, la secuencia de entrada se representa mediante pares de frases utilizando las etiquetas especiales **[CLS]**, que indica el comienzo de una nueva secuencia, y **[SEP]** que separa las dos frases. De esta manera, la secuencia se representa como: **[CLS] A [SEP] B**.

El objetivo de esta tarea es indicar si existe relación alguna entre las dos oraciones de la secuencia, como si de un clasificador binario se tratase. El 50 % de las veces la frase A guarda relación la frase B y el otro 50 % se escogen dos frases de párrafos diferentes. Conocer este tipo de relaciones resulta útil para tareas como la búsqueda de respuestas, donde la secuencia A representa la pregunta y la secuencia B el contexto donde se encuentra la respuesta.

Fine-Tuning

Esta última fase de entrenamiento de BERT consiste en entrenar el modelo utilizando otro corpus para que desempeñe una tarea específica, ya que es posible utilizar a BERT para diferentes tareas dentro del PNL: análisis de sentimientos, búsqueda de respuestas, clasificación de texto, similitud entre texto, resumen de textos, etc.

Tomando los valores que se han obtenido en la fase de pre-entrenamiento, se modifica la capa de salida para que el resultado que devuelve el modelo se ajuste a la tarea específica. Para el análisis de sentimientos el modelo devolverá un vector con probabilidades indicando si la secuencia es positiva o negativa. En el caso de la búsqueda de respuestas el valor esperado será un vector de probabilidad indicando las posiciones de comienzo y fin de la respuesta.

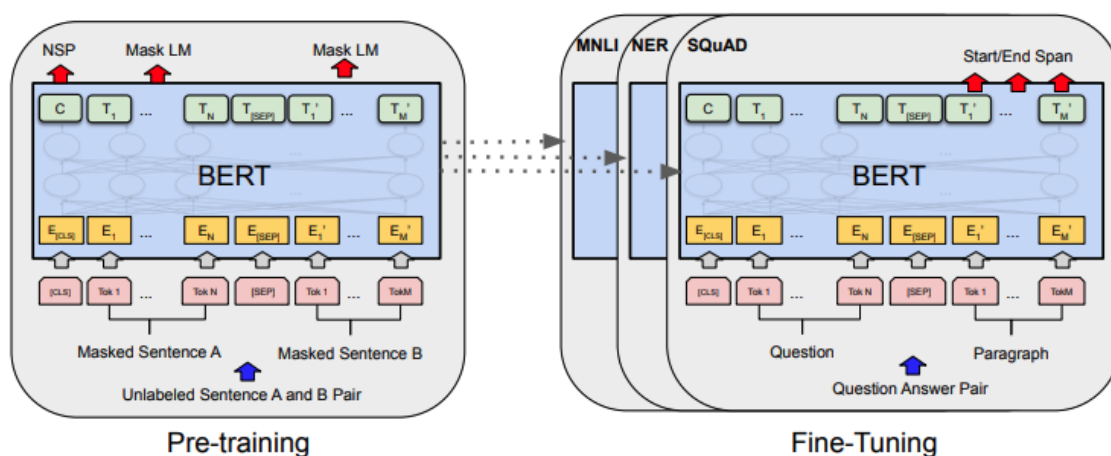


Figura 3.11: Fase de pre-entrenamiento y fine-tuning utilizando diferentes datasets [8].

3.2.5. RoBERTa

Robustly optimized BERT approach es una versión mejorada de BERT introducida por el equipo de investigación de Facebook [15]. Este modelo presenta ciertas modificaciones a la hora de entrenarlo, lo que le permite conseguir ligeramente mejores resultados que BERT en ciertas tareas del PNL.

Los cambios más importantes introducidos en este modelo consisten en la modificación de la tarea de MLM y la eliminación de la tarea de NSP. BERT aplica un enmascaramiento estático a las secuencias de entrada, es decir, se seleccionan tokens aleatorios de la frase A y B que se sustituyen por el token [MASK] y estos no son modificados durante toda la fase de pre-entrenamiento. El cambio que se propone con RoBERTa es modificar la secuencia de entrada 10 veces, de manera que cada vez se escoge un token nuevo que se sustituye por el token [MASK]. Esto implica que el dataset aumente 10 veces su tamaño.

Por otro lado, se ha eliminado la tarea de NSP ya que a través de una serie de experimentos donde se entrena a BERT con y sin dicha tarea, se llega a la conclusión de que su impacto no es relevante a la hora de evaluar la precisión del modelo. Se modifican ligeramente las secuencias de entrada y en vez de utilizar pares de frases se utilizan directamente párrafos enteros hasta alcanzar el límite máximo de tokens que el modelo puede procesar en una iteración, observando así que no es necesario entrenar al modelo con esta tarea.

3.2.6. *Word embeddings* basados en BERT

La búsqueda de respuestas depende estrechamente de la correcta recuperación de los documentos a la hora de encontrar la respuesta, lo cuál se realiza utilizando los modelos Tf-idf o BM25. Sin embargo, la aparición de BERT ha permitido la aparición de diferentes modelos de lenguaje no sólo en el campo del PNL, sino también en el campo de la recuperación de información. Recientes estudios han demostrado que es posible realizar la recuperación de pasajes dada una pregunta mediante el uso de *Word-embeddings* aplicando una función de similitud entre pregunta y párrafo.

Dense Passage Retriever

Dense Passage Retriever [14] es un modelo compuesto por dos codificadores BERT entrenados separadamente. Uno de estos codificadores procesa la pregunta y el otro procesa los párrafos donde uno de ellos contiene la respuesta.

Es necesario seleccionar pasajes que no contengan la respuesta para que así el modelo aprenda cual de ellos es el correcto. Esta selección se hace mediante tres técnicas: seleccionando pasajes aleatorios entre toda la colección de pasajes disponibles, recuperando los k pasajes más relevantes para la pregunta que no contienen la respuesta mediante el recuperador BM25 [12] y recuperando aquellos pasajes que contienen la respuesta de otra pregunta. El modelo codifica la pregunta y los diferentes pasajes en un espacio vectorial con el objetivo de que los pares de preguntas y pasajes relevantes tengan una menor distancia, para luego aplicar una función de similitud, en este caso, el producto escalar.

$$\text{sim}(q, p) = E_Q(q)^T E_P(p) \quad (3.19)$$

Para el cálculo de la función de pérdida o *Loss function* utilizamos esta función de similitud para recalcular los pesos de la red. Sea $D = \{q_i, p_i^+, p_{i,1}^-, \dots, p_{i,n}^-\}_{i=1}^m$ el conjunto de entrenamiento formado por m instancias. Cada instancia contiene una pregunta q_i , un pasaje relevante p_i^+ y n pasajes irrelevantes $p_{i,j}^-$. Optimizamos la función de pérdida como la probabilidad logarítmica negativa del pasaje positivo:

$$L(q_i, p_i^+, p_{i,1}^-, \dots, p_{i,n}^-) = -\log \frac{e^{\text{sim}(q_i, p_i^+)}}{e^{\text{sim}(q_i, p_i^+)} + \sum_{j=1}^n e^{\text{sim}(q_i, p_{i,j}^-)}} \quad (3.20)$$

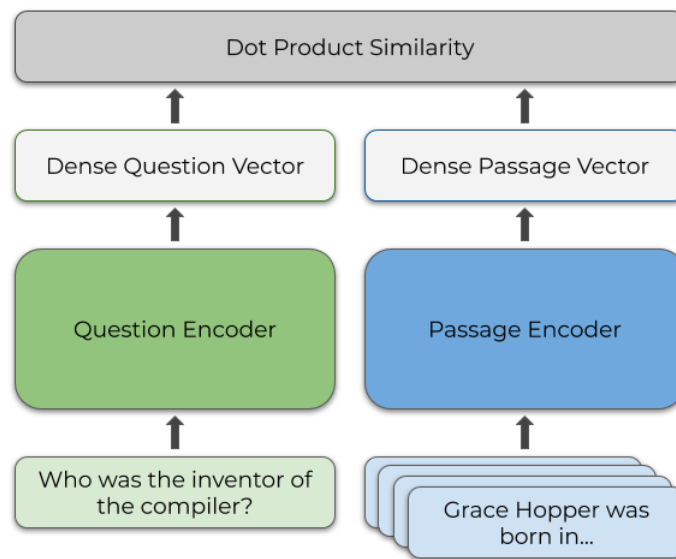


Figura 3.12: Arquitectura del DPR.

URL: <https://www.pragmatic.ml/language-modeling-and-retrieval/>

Diseño y desarrollo de la solución

Como se ha mencionado en el capítulo 2, la búsqueda de respuestas consta de 3 etapas: análisis de la pregunta, recuperación de documentos y extracción de la respuesta. En este capítulo se describirá la solución propuesta al problema explicando la arquitectura del sistema, los procesos implicados y las tecnologías empleadas para su desarrollo.

4.1 Arquitectura del sistema

Los elementos que componen el sistema de búsqueda de respuestas son los siguientes:

- **Colección de documentos (Corpus):** Repositorio o base de conocimiento de donde recuperaremos los documentos relacionados con la pregunta.
- **Document Retriever:** Componente encargado de ordenar los documentos en función de la relevancia que tienen para la pregunta dada.
- **Document Reader:** Elemento cuyo único propósito es extraer la respuesta a partir de los párrafos extraídos aplicando algoritmos de comprensión de lectura mencionados en el capítulo 3.

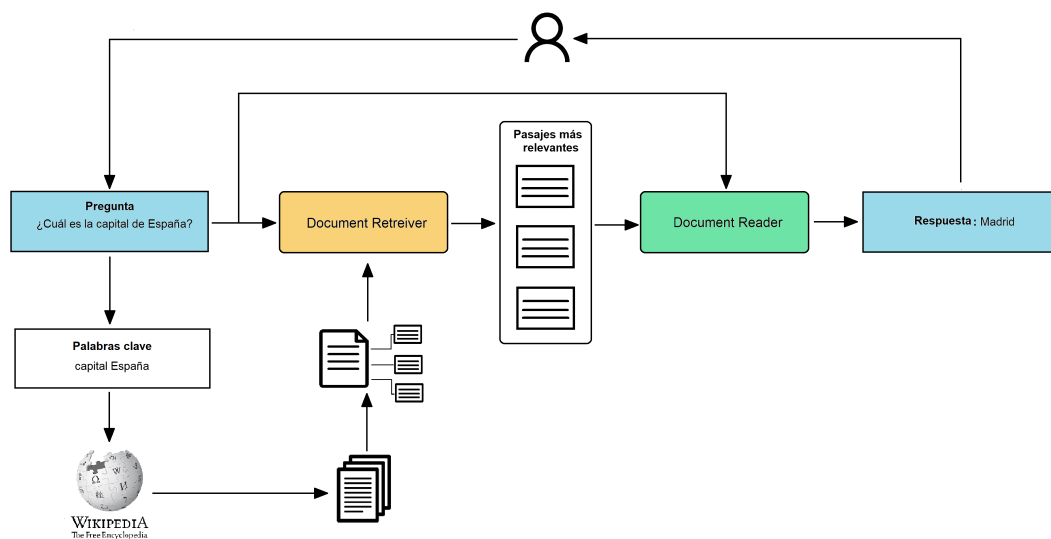


Figura 4.1: Arquitectura del sistema.

4.2 Diseño detallado

4.2.1. Corpus

Wikipedia

Wikipedia ¹ es una enciclopedia en línea gratuita multilingüe escrita y mantenida de manera colaborativa por una comunidad de voluntarios. Posee más de 59 millones de artículos en 329 idiomas, incluyendo el español y catalán, y es considerada la obra de referencia más leída y más grande de la historia.

Se ha optado por escoger a Wikipedia como base de conocimiento a la hora de buscar documentos relacionados con las consultas, ya que es uno de los repositorios de información más grandes disponibles de manera gratuita y se adapta muy bien a las necesidades de los sistemas de búsqueda de dominio abierto (2.1).

Debido al gran tamaño de dicho repositorio y a las constantes actualizaciones, se realizan peticiones a los servidores de Wikipedia para recuperar los documentos en vez de almacenarlos de forma persistente en memoria. La consulta será extraída tras analizar la pregunta y será utilizada para recuperar los documentos mediante el motor de búsqueda de Wikipedia.



Figura 4.2: Wikipedia.

4.2.2. Procesamiento de la pregunta y recuperación de documentos

Cuando el usuario introduce una pregunta al sistema, esta es analizada para extraer la información más relevante. Para dicho objetivo, la pregunta es dividida en *tokens* que se procesan individualmente. Cada uno de estos *tokens* pertenece a una categoría gramatical que se debe identificar. Para este fin se hace uso de la técnica de *Part-of-speech tagging*.

Part-of-Speech Tagging es el proceso de clasificación de las palabras en función de su categoría gramatical. Esta tarea de clasificación incluye habitualmente grupos gramaticales como verbos, sustantivos, adjetivos, adverbios, determinantes, preposiciones, etc.

Ella	escucha	música	clásica
pronombre	verbo	sustantivo	adjetivo

Figura 4.3: Ejemplo de *Part of Speech Tagging*.

El mayor reto a la hora de clasificar dichas palabras es la ambigüedad. Las palabras se comportan de manera diferente en función del contexto en el que se encuentran y por

¹<https://es.wikipedia.org/>

ello la mayor dificultad es etiquetar correctamente las palabras en base a la frase en la que se encuentran.

La librería *SpaCy* proporciona modelos entrenados en español y catalán para la tarea de *Part-of-Speech Tagging*. En concreto, se han usado los modelos *es_core_news_sm* y *ca_core_news_sm*, que son versiones optimizadas para la CPU. Mediante la llamada al método *load()*, los modelos son cargados en memoria creando por defecto una *pipeline* que incluye todos los procesos de análisis de texto. La pipeline devuelve un objeto de tipo *Doc* cuyos atributos son los resultados de cada uno de estos procesos. Accediendo al atributo *Doc.pos* obtenemos la categoría gramatical de cada *token*.

Luego de identificar la categoría gramatical, es necesario descartar aquellas palabras que no aportan valor a la consulta. Estas palabras se denominan *stopwords* y no tienen un significado por sí solas, sino que modifican o acompañan a otras. Este grupo de palabras está conformado por artículos, conjunciones, preposiciones, pronombres y adverbios. La categorías restante tras la eliminación de *stopwords* es la de los sustantivos (comunes y propios) que son los que aportan más información a la recuperación de documentos.

Tras esto, se realiza una consulta al motor de búsqueda de Wikipedia con los *tokens* remanentes por medio de la librería Wikipedia-API 4.3.3, que facilita la recuperación del texto del documento, dejando fuera a las imágenes, tablas y etiquetas HTML. A continuación se detallan los todos lo métodos necesarios para recuperar los documentos:

El contenido de los documentos se organiza por secciones donde se encuentra el texto y a su vez puede contener más subsecciones. Los pasos seguidos para obtener el texto final han sido:

- Obtención de secciones de cada artículo.
- Supresión de aquellas secciones que no aportan información a la hora de buscar respuestas (Referencias, Enlaces Externos, Bibliografía, Véase también, etc).
- Análisis recursivo de cada sección extrayendo el texto.
- Eliminación de expresiones regulares, caracteres especiales y saltos de línea.

El resultado final es un diccionario de documentos cuya clave es el título del documento y el valor el texto recuperado.

4.2.3. Procesamiento del texto

Una vez obtenido el texto de cada documento, se lleva a cabo su procesamiento. El tamaño de los documentos recuperados es muy variable y los modelos de lenguaje son capaces de procesar el texto hasta cierto límite de longitud, por lo que estos documentos deben ser descompuestos en conjuntos más pequeños.

Cada documento es dividido en frases con la ayuda de la librería *Spacy*, que permite usar como criterio de separación los signos de puntuación, garantizando así que la separación de las frases no caiga en medio. Estas frases se van concatenando hasta alcanzar un número máximo de tokens. Se ha establecido como número máximo de *tokens* por cada conjunto un total de 120, aunque en algunos casos estos serán más por el hecho de respetar los signos de puntuación. El criterio empleado para la selección de dicho valor se ha hecho en base a la media ponderada del *dataset SQuAD*, el cuál es 126 [17].



Figura 4.4: Ejemplo de distribución del tamaño de los párrafos.

4.2.4. Document Retriever

Inmediatamente después de indexar el texto, el *document retriever* se encarga de extraer aquellos pasajes entre toda la colección con probabilidad de contener la respuesta, ordenándolos de forma ascendente en función de su relevancia. Para ello, se han utilizado dos modelos basados en *Dense Passage Retriever* 3.2.6, uno para el castellano y otro para el catalán.

A pesar de existir diversas soluciones para esta etapa, como por ejemplo el uso de *Sentence-BERT* [18] que calcula la similitud semántica textual usando pares de frases, *DPR* está optimizado para maximizar las similitudes entre pregunta y pasaje usando dichos pares para el entrenamiento.

Castellano

Para el lenguaje español se ha utilizado un DPR desarrollado por el Instituto de Ingeniería del Conocimiento (IIC) ² que ellos mismos consideran como «el mejor *DPR* en español hasta la fecha». Este modelo ha sido entrenado con los datasets *SQUAD-ES* [6], *Spanish Question Answering Corpus* [9] y *BioAsq22-ES* consiguiendo un rango promedio de 0.117 sobre 1, lo que es equivalente a decir que el pasaje relevante se encuentra en la posición 11 de los 100 pasajes que se recuperan.

Catalán

Se ha realizado en este proyecto el "afinado" o *fine-tuning* de un modelo de lenguaje español preentrenado basado en la arquitectura BERT llamado BETO [5]. Empleando el corpus de ViquiQuAD [2], el modelo resultante ha sido compartido en la plataforma *HuggingFace* ^{3,4} a disposición de la comunidad. A continuación se especifican los detalles.

- **Corpus:** ViquiQuAD es adaptación de SQuAD al catalán creado a partir de 576 artículos de Wikipedia originalmente escritos en catalán y no traducidos de otro idioma. De estos 576 artículos se han escogido aleatoriamente 3129 párrafos para

²<https://www.iic.uam.es/>

³https://huggingface.co/Koslav/dpr-catalan-question_encoder-viquiquad-base

⁴https://huggingface.co/Koslav/dpr-catalan-passage_encoder-viquiquad-base

elaborar hasta 5 preguntas de cada uno, lo que supone un total de más de 15000 preguntas para entrenar a nuestros codificadores.

- **Modelo:** BETO está basado en la arquitectura BERT siendo entrenado con datos de diferentes fuentes de información como Wikipedia y el proyecto OPUS [20], haciendo que el corpus de entrenamiento sea comparable al de BERT.
- **Fine-tuning:** Siguiendo el trabajo desarrollado en [14], se ha reestructurado el corpus para afinar el DPR, de manera que por cada pregunta y su pasaje relevante seleccionamos un conjunto de pasajes no relevantes como se explica en la sección 3.2.6. Se ha realizado un total de 10 *epochs* utilizando un *batch-size* de 16 muestras con un *learning rate* de $1e-12$, obteniendo así un *recall* de 54,62 % en la fase de test.

Los pasajes indexados pasan por el *document retriever* que calcula el *embedding* para cada pasaje siendo representado por un vector denso. Cuando el usuario realiza una pregunta, esta también se representa como un *embedding* para poder calcular la similitud mediante el producto escalar. Cuando se procesa el cómputo de todos los pasajes en relación a la pregunta, se ordenan descendientemente en función del valor obtenido con el fin de seleccionar una cantidad reducida del conjunto que tendrán mayor probabilidad de albergar la respuesta.

4.2.5. Document Reader

La última etapa de este proceso es la extracción de respuestas. El *document reader* es el responsable de analizar la pregunta y los pasajes proporcionados por el *document retriever* mediante un modelo de lenguaje. El resultado devuelto por el *document reader* son dos posiciones dentro del pasaje que indican el inicio y fin de la respuesta.

Se han escogido dos modelos preentrenados (MarIA y BERTa), uno para el castellano y otro para el catalán, desarrollados por el grupo de investigación *Barcelona Super Computing*⁵. La elección de dichos modelos se debe al buen desempeño que presentan a la hora de extraer respuestas, considerándose de los mejores dentro de los modelos en castellano y catalán.

Castellano

MarIA [9] es una familia de modelos de lenguaje en español compartidos de manera pública para la industria y comunidad científica. Este modelo de lenguaje se basa en la arquitectura de RoBERTa, que ha sido entrenado con un corpus masivo de 570GB de texto limpio y deduplicado, extraído del Archivo Web del Español construido por la Biblioteca Nacional de España entre 2009 y 2019. Tras preentrenar este modelo con dicho corpus, este es afinado con diferentes conjuntos de datos para diferentes tareas. En el caso de la búsqueda de respuestas el modelo se ha afinado con SQAC (*Spanish Question Answering Corpus*) consiguiendo un valor para la métrica F1-Score del 79.23 %.

Se ha utilizado la versión base del modelo RoBERTa, aunque la versión *large* presente ligeramente mejores resultados, el tiempo de carga e inferencia del modelo es notablemente mayor. Las especificaciones de la arquitectura de este modelo son:

- **Número de parámetros:** 125 millones
- **Número de capas ocultas:** 12

⁵<https://www.bsc.es/>

- **Número de cabezales de atención:** 12
- **Tamaño del vector oculto:** 768

La configuración que se ha seguido durante la fase *fine-tuning* es la siguiente:

- **Batch size:** 16
- **Epochs:** 5
- **Learning rate:** 1e-5
- **Tamaño máximo de la secuencia de entrada** 512

Catalán

BERTa [2] es modelo de lenguaje basado también en RoBERTa. En su caso, ha sido entrenado con otro corpus (CaText) creado por el BSC que agrupa diferentes *datasets* disponibles en el idioma catalán y el suyo propio. Para su evaluación, se ha utilizado CLUB (*Calatan Language Understanding Benchmark*)⁶. La evaluación de la búsqueda de respuestas se realiza con el corpus ViquiQuAD, obteniendo una precisión exacta del 73.25 % y un 86.99 % como F1-Score.

También se ha utilizado el modelo base de RoBERTa con las siguientes especificaciones:

- **Número de parámetros:** 110 millones
- **Número de capas ocultas:** 12
- **Número de cabezales de atención:** 12
- **Tamaño del vector oculto:** 768

En la fase de *fine-tuning*, la configuración del modelo ha sido:

- **Batch size:** 32
- **Epochs:** 10
- **Learning rate:** 5e-5
- **Tamaño máximo de la secuencia de entrada** 384

4.2.6. Implementación de interfaz gráfica

Con el fin de poder examinar las respuestas obtenidas por el *document reader* de manera sencilla y sin necesidad de contemplar todos los procesos subyacentes que tienen lugar a la hora de encontrar una respuesta, se ha desarrollado una interfaz gráfica web a modo de demo con la librería Gradio⁷. Para seleccionar el idioma en el que se quiera realizar la pregunta, disponemos de dos pestañas que al ser seleccionadas cambian de idioma. La ejecución comienza cuando el botón de "Buscar" o "Trovar" es presionado, tomando como valor de entrada la pregunta que se introduce en el campo de texto "Pregunta" situado en la parte superior de la pantalla principal. Tras extraer la respuesta de

⁶<https://github.com/projecte-aina/club>

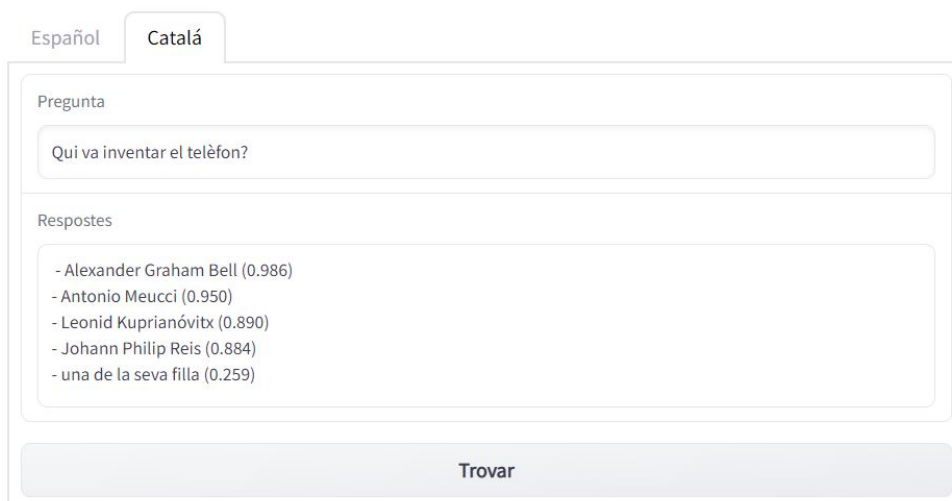
⁷<https://gradio.app/>

los posibles pasajes que la contienen, el resultado junto con la probabilidad de acierto, se imprime por pantalla en el campo de texto "Respuestas" o "Respostes" situado justo debajo de donde se introduce la pregunta.



The screenshot shows a web interface for Spanish. At the top, there are two tabs: "Español" (selected) and "Catalá". Below the tabs is a "Pregunta" (Question) section with a text input field containing "¿Cuál es la capital de Rusia?". Below that is a "Respuestas" (Answers) section with a list of answers and their probabilities: "- Kurgán (0.999)", "- San Petersburgo (0.999)", "- Moscú (0.999)", "- Moscú (0.999)", and "- Moscú (0.999)". At the bottom is a "Buscar" (Search) button.

Figura 4.5: Interfaz para el español.



The screenshot shows a web interface for Catalan. At the top, there are two tabs: "Español" and "Catalá" (selected). Below the tabs is a "Pregunta" (Question) section with a text input field containing "Qui va inventar el telèfon?". Below that is a "Respostes" (Answers) section with a list of answers and their probabilities: "- Alexander Graham Bell (0.986)", "- Antonio Meucci (0.950)", "- Leonid Kuprianóvitx (0.890)", "- Johann Philip Reis (0.884)", and "- una de la seva filla (0.259)". At the bottom is a "Trovar" (Find) button.

Figura 4.6: Interfaz para el catalán.

4.3 Tecnologías utilizadas

4.3.1. Lenguaje de programación

Python

Python⁸ es un lenguaje de programación multiparadigma ampliamente utilizado por investigadores debido a su sencillez, versatilidad y sintaxis cercana al lenguaje natural. La ventaja que ofrece Python en comparación con otros lenguajes es la gran cantidad de librerías desarrolladas por diferentes organismos de investigación que facilitan la tarea del procesamiento del lenguaje natural y que se adaptan perfectamente con los requisitos de este proyecto.

⁸<https://www.python.org/>

Además se han utilizado en el proyecto los siguientes módulos nativos de Python:

- *Re*: Utilizado para el procesamiento de expresiones regulares dentro del texto.
- *Json*: Gestión de ficheros en formato *.json*.
- *Time*: Medición del tiempo de ejecución del código

4.3.2. Entorno de desarrollo

Google Colab

Google Colab ⁹ es un servicio de notebooks basado en el proyecto de código abierto de Jupyter ¹⁰ ofrecido de manera gratuita por la empresa Google. Este servicio brinda recursos computacionales sin coste alguno y sin ningún tipo de instalación para todo el público.

Debido a las limitaciones computacionales encontradas a la hora de desarrollar este proyecto (como falta de memoria RAM o GPU obsoleta) se ha decidido realizar todo el proyecto utilizando esta plataforma.

4.3.3. Paquetes y librerías

A continuación se especifican las librerías implementadas en el lenguaje de programación Python que se han utilizado durante este proyecto.

SpaCy

SpaCy ¹¹ es una librería destinada al PNL que se enfoca en el procesamiento y clasificación de texto. Entre las diferentes tareas que esta librería proporciona destacan: tokenización, lematización, análisis morfológico, análisis sintáctico, análisis semántico, reconocimiento de entidades (*NER*).

Esta librería está disponible en numerosos idiomas, incluido el catalán y por ello se ha elegido para realizar el preprocesamiento de la pregunta.

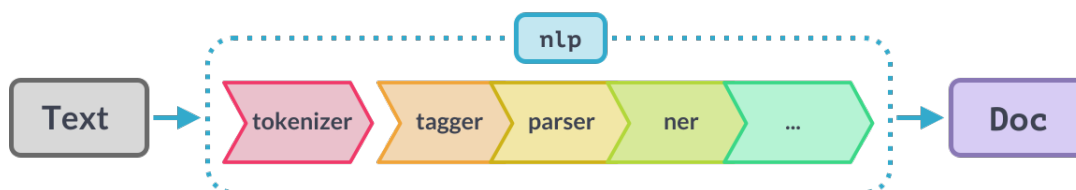


Figura 4.7: Pipelines de la librería SpaCy.

Hugging Face

Hugging Face ¹² es una plataforma de código abierto cuya finalidad es proporcionar herramientas para la construcción, entrenamiento y despliegue de modelos de *Deep Learning*

⁹<https://colab.research.google.com/>

¹⁰<https://jupyter.org/>

¹¹<https://spacy.io/>

¹²<https://huggingface.co/>

Empresas de renombre como Facebook, Microsoft o Google utilizan esta plataforma para compartir sus modelos a disposición de la comunidad para utilizarlos sin necesidad de entrenarlos desde cero, ya que la cantidad de recursos computacionales necesarios para dicho cometido es desmesurada y no está al alcance de cualquiera.

Para el desarrollo de este proyecto se han utilizado varios modelos en castellano y catalán disponibles dentro del repositorio:

– Castellano

- *IIC/dpr-spanish-question_encoder-squades-base*: Codificador de preguntas preentrenado basado en el modelo *Dense Passage Retriever*
- *IIC/dpr-spanish-passage_encoder-squades-base* : Codificador de pasajes preentrenado basado en el modelo *Dense Passage Retriever*.
- *PlanTL-GOB-ES/roberta-base-bne-sqac*: Modelo RoBERTa preentrenado en español y afinado para la tarea de QA.

– Catalán

- *Koslav/dpr-catalan-question_encoder-viquiquad-base*: Codificador de preguntas basado en el modelo *Dense Passage Retriever* entrenado en este proyecto y compartido con la comunidad
- *Koslav/dpr-catalan-passage_encoder-viquiquad-base*: Codificador de pasajes basado en el modelo *Dense Passage Retriever* entrenado en este proyecto y compartido con la comunidad.
- *projecte-aina/roberta-base-ca-cased-qa*: Modelo RoBERTa preentrenado en catalán y afinado para la tarea de QA.

Haystack

Haystack ¹³ es un *framework* que permite la construcción de sistemas de búsqueda eficientes sobre grandes colecciones de documentos. Hace uso de diferentes librerías como Hugging Face, NLTK o scikit-learn para usarlos conjuntamente mediante la creación de nodos, que corresponden con los diferentes módulos que un sistema de búsqueda de respuestas puede tener.

Wikipedia-API

Wikipedia-API ¹⁴ es un *wrapper* que permite realizar peticiones a los servidores de Wikipedia para recuperar texto sin necesidad de utilizar las sofisticadas peticiones HTTP que la API oficial de Wikipedia ofrece. Esta librería ha sido muy útil a la hora de procesar el texto de la consulta evitando la necesidad de analizar completamente la petición. Los métodos utilizados de esta librería han sido:

- *set_lang()*: establece el idioma en el que se devuelven los documentos.
- *search()*: método que devuelve una lista de títulos iguales o semejantes en base a una consulta. Este método es de gran utilidad, ya que proporciona una expansión de la consulta y a su vez tolerancia a las faltas ortográficas, ya que si se tratase de una petición HTTP, el servidor devolvería un error.
- *page()*: lista de secciones contenidas en un artículo.

¹³<https://haystack.deepset.ai>

¹⁴<https://pypi.org/project/Wikipedia-API/>

Gradio

Gradio [1] es una librería que facilita el acceso a demostraciones de aprendizaje automático a usuarios que carecen de *hardware/software* especializado para su construcción. Esta librería permite la rápida creación de interfaces gráficas que pueden ser compartidas con un simple enlace URL.

Uno de los principales retos del uso de modelos neuronales es la accesibilidad. Estos modelos son típicamente construidos por investigadores experimentados que tienen acceso a grandes centros de cómputo, lo que hace que colaboradores no técnicos y usuarios finales tengan dificultades para dar su opinión sobre el desempeño del modelo en escenarios realistas.

Se ha optado por el uso de Gradio debido a la compatibilidad que tiene con los cuadernos Jupyter, en los cuáles se basa el entorno de desarrollo de este proyecto.

Los modelos neuronales son albergados en el sistema del anfitrión, el cuál comparte con los usuarios la URL que la propia librería genera automáticamente. Cuando un usuario introduce un parámetro de entrada por alguno de los componentes de la interfaz, se crea una conexión SSH con el anfitrión que recibe los datos y ejecuta una función que engloba todos los cálculos que el modelo neuronal debe realizar. El valor que obtiene el modelo es devuelto al usuario a través del túnel SSH y se imprime por pantalla.

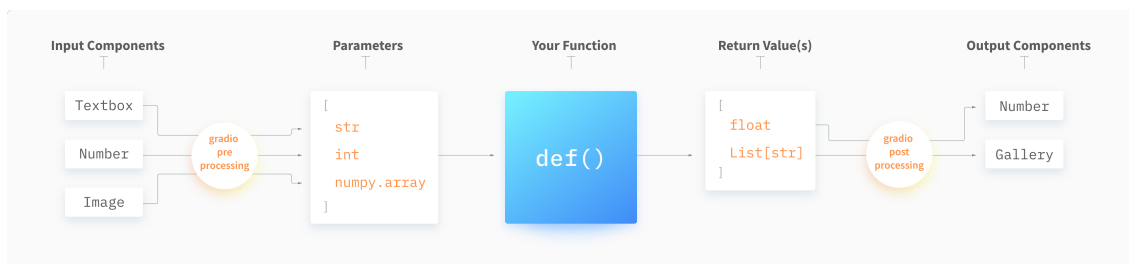


Figura 4.8: Desglose del funcionamiento de la librería Gradio.

CAPÍTULO 5

Evaluación y resultados

La evaluación de los sistemas de búsqueda de respuestas se puede realizar de manera conjunta o aislada. En la evaluación conjunta, cada componente recibe las predicciones del componente anterior como *input*. Esta evaluación muestra el rendimiento que los usuarios pueden esperar a la hora de realizar una búsqueda fuera del conjunto de entrenamiento. En cambio, la evaluación aislada muestra el rendimiento máximo que cada componente puede alcanzar si los datos de entrada son perfectos. El uso de modelos pre-entrenados implica que estos ya hayan sido evaluados con diferentes conjuntos de *test*. Este es el caso del *document reader*. Como se ha mencionado previamente en el capítulo *refdoc-reader*, *MarIA* y *BERTa* han sido entrenados y evaluados con adaptaciones del corpus *SQuAD*[17]. Estas evaluaciones del rendimiento no tienen en cuenta que la extracción de la respuesta se haga sobre un pasaje erróneo, ya que los datos de entrada son una pregunta y un contexto que contiene la respuesta. Por esta razón, es necesario realizar una evaluación conjunta del sistema para observar el rendimiento a la hora de predecir respuestas.

Se han diseñado dos datasets extrayendo pasajes de diversos artículos de la Wikipedia en español y catalán a partir de los cuáles se han realizado preguntas y extraído respuestas. El número total de preguntas en el dataset es de 20 para el español y catalán. Destacar que los contenidos de la Wikipedia en español y catalán difieren bastante el uno del otro, siendo el catalán muy pobre en comparación con el español, por lo que 10 de estas preguntas son comunes mientras que las otras 10 han sido seleccionadas de artículos diferentes.

5.1 Métricas utilizadas

Para la evaluación del *document reader* se hará uso de las métrica *F1-score* y *Exact Match* mientras que para el *document retriever* usaremos *Recall* y *Mean Reciprocal Rank*.

5.1.1. *F1-score*

F1-score es una métrica utilizada para evaluar la precisión. Está compuesta por la media armónica de las variables *Precision* y *Recall*. Dada una recuperación de documentos en base a una consulta, podemos distinguir cuatro grupos:

- Verdadero positivo (*TP*): documentos que han sido recuperados y son relevantes.
- Falso positivo (*FP*): documentos que han sido recuperados pero no son relevantes.
- Verdadero negativo (*TN*): documentos que no han sido recuperados y no son relevantes.

— Falso negativo (*FN*): documentos que no han sido recuperados pero son relevantes.

Estas cuatro categorías se pueden representar mediante un matriz de confusión, que compara las predicciones realizadas con los valores de las muestras reales.

		PREDICTED	
		POSITIVE	NEGATIVE
ACTUAL	POSITIVE	TRUE POSITIVES	FALSE NEGATIVES
	NEGATIVE	FALSE POSITIVES	TRUE NEGATIVES

Figura 5.1: Matriz de confusión.

URL: <https://towardsdatascience.com/precision-and-recall-88a3776c8007>

Precision

La *Precision* es el ratio de verdaderos positivos entre el número total de verdaderos positivos y falsos positivos.

$$Precision = \frac{TP}{TP + FP} \quad (5.1)$$

Podemos interpretar estos valores dentro de la predicción de una respuesta como el número de palabras que coinciden entre la predicción y la respuesta real dividido por el número de palabras de la predicción. Supongamos por ejemplo la predicción para la pregunta "¿Cuándo murió Mozart?" es "en 1756". La respuesta correcta para esta pregunta es "27 de enero de 1756", por lo que el número de *tokens* comunes entre respuesta y predicción sería 1756 y el valor de la *precision* se $1/2 = 0.5$.

Recall

El *Recall* se entiende como el cómputo de verdaderos positivos dividido por la suma de verdaderos positivos y falsos negativos.

$$Recall = \frac{TP}{TP + FN} \quad (5.2)$$

En este caso, el valor se calcula en función del número de palabras contenidas en la respuesta. Tomando el ejemplo anterior, el número de palabras comunes seguirá siendo el mismo pero esta vez será dividido entre el número de palabras contenido en la respuesta, cuyo valor sería $1/5 = 0.2$.

Finalmente, la métrica F1-score se calcula como:

$$F1\text{-score} = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \quad (5.3)$$

5.1.2. Mean Reciprocal Rank

Mean Reciprocal Rank es una medida estadística que mide el grado de precisión con el que un proceso genera una lista de posibles candidatos a una consulta. Dado un conjunto de consultas Q , el *reciprocal rank* de una respuesta para consulta contenida en Q se calcula como el inverso multiplicativo de la posición en la que se encuentra la primera respuesta correcta.

$$MRR = \frac{1}{|Q|} \sum_{i=1}^Q \frac{1}{rank_i} \quad (5.4)$$

Consulta	¿A qué distancia está Marte?	¿Cuántos kg es una tonelada?
Resultados propuestos	Sistema solar, Marte , Tierra	tonelada , kilogramo, masa
Posición	2	1
Reciprocal rank	2/3	1/3
Mean Reciprocal Rank	(2/3) + (1/3)/2 = 0.5	

Tabla 5.1: Ejemplo de *Mean Reciprocal Rank*.

5.2 Evaluación aislada

Como se ha mencionado anteriormente, la evaluación de los módulos que componen el sistema puede hacerse de manera aislada o conjunta. Los resultados obtenidos se han realizado a partir del análisis de las 20 preguntas creadas para español y catalán. Se ha establecido un número total de 20 documentos a recuperar de Wikipedia para el cálculo del *Recall* y el *Mean Reciprocal Rank* con motivo de evitar desbalance entre ambos modelos. Destacar que el valor del *Recall* es calculado tomando únicamente el primer pasaje recuperado. Consideraremos relevante un pasaje en caso de que proceda del mismo documento del que se haya extraído la pregunta. Los valores *F1-Score* y *Exact Match* se han calculado a partir de la pregunta y el pasaje del que ha sido recuperado. Aunque los modelos ya estén preentrenados, se ha optado por calcular su rendimiento con el dataset creado. A continuación se muestran los resultados:

5.2.1. Castellano

Resultados del modelo en español.

	Recall	Mean Reciprocal Rank
Document Retriever	0,6	0,787

Tabla 5.2: Resultados del *document retriever* para el español.

	F1-Score	Exact Match
Document Reader	0.6858	0.4454

Tabla 5.3: Resultados *document reader* para español.

5.2.2. Catalán

Resultados del modelo en catalán.

	Recall	Mean Reciprocal Rank
Document Retriever	0.633	0,725

Tabla 5.4: Resultados *document retriever* para catalán.

	F1-Score	Exact Match
Document Reader	0,6154	0,4

Tabla 5.5: Resultados *document reader* para catalán.

5.2.3. Comparativa recalls

Adicionalmente se ha realizado el cálculo del *Recall* para los 5, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100 pasajes recuperados con el objetivo analizar el número de pasajes óptimos que se deben devolver para extraer la respuesta. A continuación se muestran los resultados.

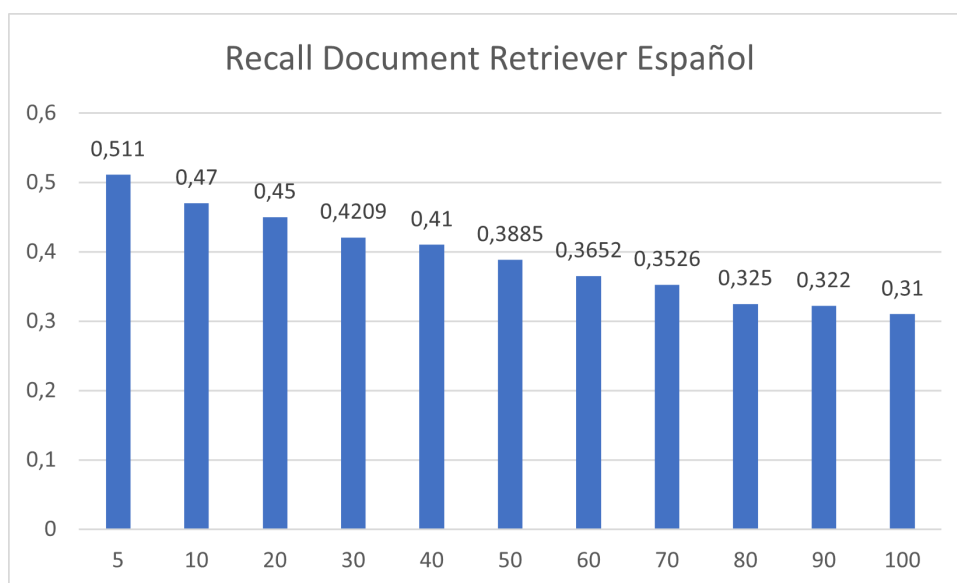


Figura 5.2: Recall obtenido del document retriever para el español.

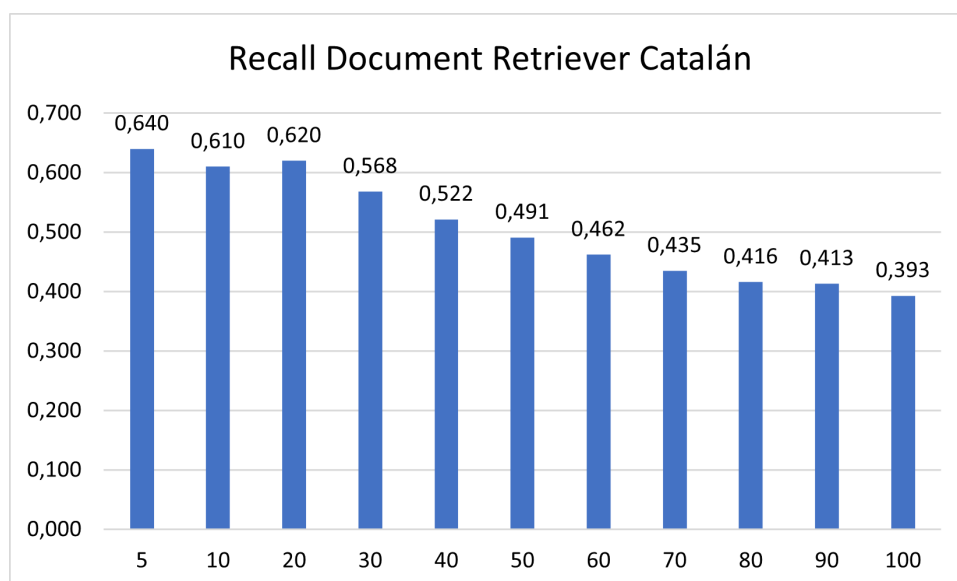


Figura 5.3: Recall obtenido del document retriever para el catalán.

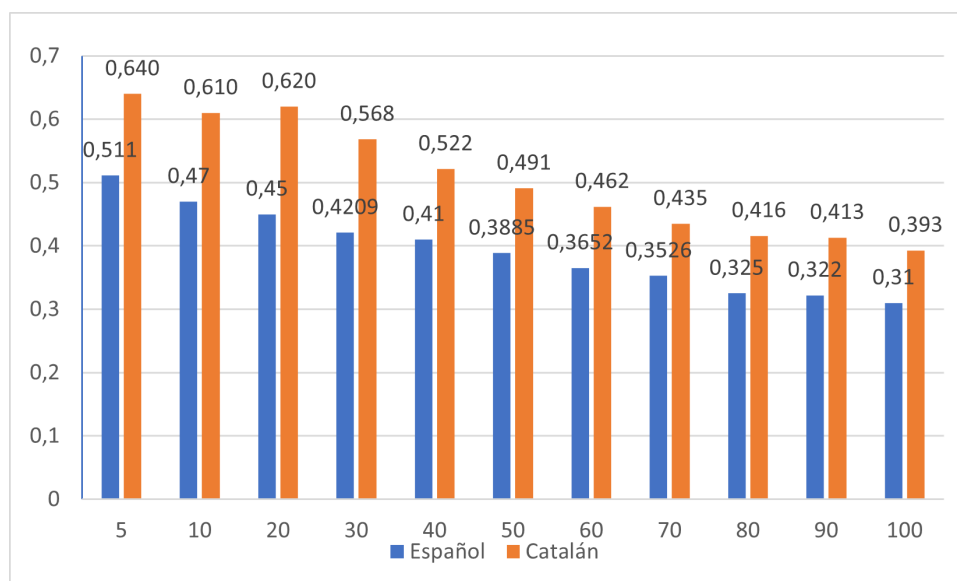


Figura 5.4: Comparativa de los recalls de ambos modelos.

En conclusión podemos observar que el document reader para el catalán obtiene mejores resultados a la hora de filtrar pasajes. Esto no quiere decir que el modelo se desenvuelva mejor que el modelo en castellano, ya que si observamos el valor del Mean Reciprocal Rank, el document retriever en castellano clasifica los pasajes de mejor manera, lo que significa que encuentra el pasaje relevante. La razón por la que el modelo en catalán obtiene un mejor resultado se debe al menor tamaño de la base de conocimiento de Wikipedia en catalán, lo que permite un mejor rendimiento a la hora de recuperar los pasajes.

5.3 Evaluación conjunta

Tomando los 5 pasajes más relevantes que el *document retriever* obtiene en base a la pregunta, el *document reader* predice para cada pasaje una respuesta que se compara con la respuesta real de la pregunta. De estas 5 predicciones se calcula el máximo *F1-Score* y el *Exact Match* obteniendo los siguientes resultados:

5.3.1. Castellano

	F1-Score	Exact Match
Document Reader	0,543	0,389

Tabla 5.6: Resultados del *document reader* para el español.

5.3.2. Catalán

	F1-Score	Exact Match
Document Reader	0,504	0,366

Tabla 5.7: Resultados *document reader* para catalán.

5.4 Tiempos

En esta sección se exponen los tiempos de ejecución del sistema de varias etapas. En cada una de los siguientes apartados de esta sección se detallará la forma de tomar las medidas.

Procesamiento de la pregunta y recuperación de documentos

Se han utilizado todas las preguntas del dataset de creación propia para recuperar los documentos más relevantes a la hora de hacer consultas Wikipedia, ajustando el número máximo de documentos a recuperar. Se ha ejecutado un total de 5 iteraciones por cada valor máximo de documentos para promediar los tiempos de ejecución.

Documentos	Media	Desviación típica
5	5,3	0,8
10	10,8	1,1
20	19,1	2,9
30	28,1	4,6
40	34,8	3,9
50	48,9	7,1
60	53,6	4,3
70	65,2	7,1
80	70,2	6,2
90	82,4	9,1
100	94	2,645

Tabla 5.8: Tiempo en segundos medio de la recuperación de documentos.



Figura 5.5: Tiempo medio de análisis y recuperación de documentos.

Se puede observar que el tiempo medio de recuperación de un documento de Wikipedia ronda el segundo, siguiendo un coste lineal. Destacar que estos tiempos son muy susceptibles al cambio ya que dependen totalmente de latencia.

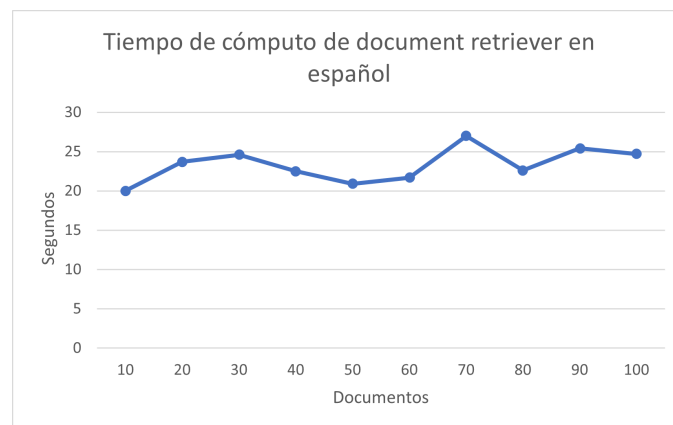
Document Retriever

Figura 5.6: Tiempo de inferencia del *document retriever* en castellano.

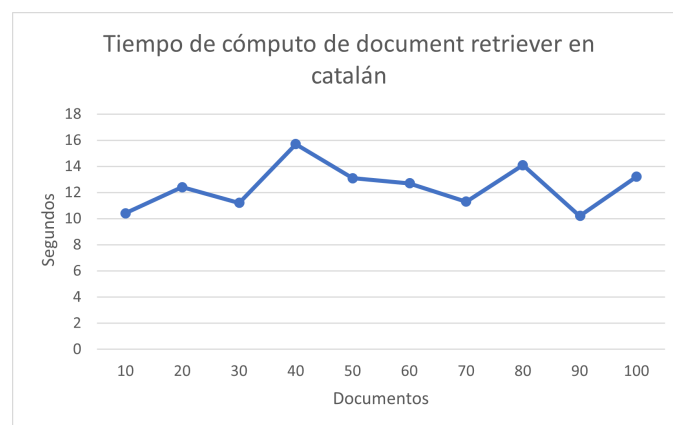


Figura 5.7: Tiempo de inferencia del *document retriever* en catalán.

Estos tiempos se corresponden con el cálculo de los vectores densos de cada pasaje y el uso de la función de similitud coseno. Podemos observar que el tiempo de cómputo es prácticamente constante. Esto se debe principalmente por la capacidad de inferencia de la GPU que ofrece el entorno de desarrollo Google Colab, la cuál es un *NVIDIA Tesla P100* de 16GB de memoria.

Conclusiones y trabajos futuros

En el presente proyecto se ha abordado el desarrollo de un sistema de búsqueda de respuestas de dominio abierto a preguntas factuales en los idiomas español y catalán mediante el uso de redes neuronales. Para cumplir dicho objetivo, se han estudiado y analizado las herramientas, técnicas y modelos disponibles en la actualidad, lo que ha sido de vital importancia para la resolución del problema. Además, se ha creado una interfaz gráfica web que integra los procesos de recuperación de pasajes y extracción de respuestas y permite un uso sencillo e intuitivo para el usuario final.

Para evaluar el correcto funcionamiento del sistema, se elaboró un conjunto de preguntas y respuestas de test (compuesto por preguntas factuales como *¿Cuál es la capital de Rusia?*). En base a los resultados obtenidos, podemos afirmar que el objetivo principal de este proyecto se ha cumplido de manera satisfactoria, siendo el sistema capaz de encontrar respuesta a gran parte de las preguntas efectuadas. A su vez, estos resultados han derivado varias conclusiones acerca de la efectividad del sistema en comparación con los modelos clásicos y variantes neuronales.

Por un lado, el uso de *Word Embeddings* permite recuperar documentos en función del contexto y la semántica, a diferencia de las aproximaciones clásicas de recuperación que se basan en ocurrencias exactas de términos. Esto conduce a recuperar documentos potencialmente mejores dada una pregunta en lenguaje natural que podría ser formulada de maneras muy diversas. Además, los modelos de *embeddings* pueden ser ajustados para dominios, lenguajes y tareas específicas con el fin de mejorar su rendimiento, por ejemplo, pueden ser optimizadas para reducir la distancia entre pregunta y documentos relevantes, como es el caso de los DPR empleados en este trabajo.

Por otro lado, durante el desarrollo del proyecto, la solución inicial propuesta a la recuperación y clasificación de documentos estaba basada en el uso de *Sentence-BERT* [18] debido a su reducido tiempo de cómputo y escalabilidad a la hora de obtener *Word Embeddings*. Sin embargo, los resultados obtenidos no fueron lo suficientemente satisfactorios, lo que supuso la búsqueda de nuevas soluciones dando así lugar al uso de los DPR. En este aspecto, podemos destacar que los modelos entrenados con conjuntos de datos similares a nuestra tarea, es decir, preguntas y pasajes, presentan un mejor rendimiento que aquellos que son entrenados con otro conjunto de datos, por ejemplo pares de sentencias en el caso de *Sentence-BERT*, aunque ello suponga un aumento en los tiempos de cómputo.

Respecto a las librerías de código abierto utilizadas, como Hugging Face o SpaCy, su uso ha sido de gran utilidad a la hora de desarrollar este proyecto. En concreto, la librería Hugging Face permite importar modelos preentrenados compartidos por la comunidad de manera gratuita, por lo que evita la necesidad de diseñar e implementar gran parte de los modelos neuronales utilizados para el desarrollo de los componentes *document retriever* y *document reader*. A pesar de que hay disponibles muchos modelos preentrenados

en su *Hub*, para algunos casos específicos la comunidad todavía no ha compartido los modelos. Este es el caso del *document retriever* basado en DPR para el catalán. Con motivo de satisfacer esta necesidad, hemos creado uno propio, siendo el primero para el idioma catalán liberado en Hugging Face para su libre uso.

En cuanto a futuros trabajos, la arquitectura de este sistema permite reemplazar con facilidad los modelos neuronales que componen los módulos del *document retriever* y *document reader* por modelos que presentarán un mejor rendimiento. En este aspecto, el sistema es capaz de adaptarse a los avances y cambios que están por venir en el campo del procesamiento natural del lenguaje, reflejándose así en un mejor resultado.

Un aspecto que no se ha tratado en este proyecto y que afecta directamente al correcto funcionamiento del sistema es la veracidad de los artículos y documentos recuperados. En la actualidad, la manipulación de la información es algo habitual en Internet y en los medios de comunicación. Como posible solución a este problema, existen modelos neuronales que son entrenados para detectar si una noticia o artículo es falsa. Los documentos recuperados serían filtrados por este modelo para evitar así sesgos no deseados.

Asimismo, cada una de las etapas de la búsqueda de respuestas presenta posibles mejoras o nuevos planteamientos que no se han tenido en cuenta a la hora de realizar este proyecto pero que son dignos de mencionar.

En primer lugar, en la fase de análisis de la pregunta, una posible mejora sería utilizar una expansión de la consulta usando la base de datos *WordNet* tras haber extraído las palabras clave con el objetivo de recuperar el máximo número de documentos. También cabe destacar la posibilidad de clasificar el tipo de pregunta factual al que nos enfrentamos (dónde, cuándo, quién...), algo común en los modelos clásicos de búsqueda de respuestas, reduciendo así el abanico de posibles respuestas ya que sabemos el tipo de entidad que estamos buscando.

En segundo lugar, en la etapa de recuperación de documentos, la construcción de una base de datos persistente como *ElasticSearch* donde guardar documentos asociados a una consulta realizada en Wikipedia, junto con el uso de FAISS [11] que permite la búsqueda de similitudes entre *Word Embeddings* de manera eficiente, reduciría considerablemente el tiempo de ejecución de esta etapa.

Por último, en la extracción de la respuesta, podemos plantear un enfoque totalmente distinto, el cual consiste en prescindir de todos los componentes del sistema para utilizar un modelo de lenguaje autorregresivo como GPT-3 [4] que es capaz de generar texto a partir de una secuencia de entrada. La respuesta sería reescrita en base a la pregunta para que el modelo complete la oración con el valor factual.

Bibliografía

- [1] Abubakar Abid y col. *Gradio: Hassle-Free Sharing and Testing of ML Models in the Wild*. 2019. DOI: [10.48550/ARXIV.1906.02569](https://doi.org/10.48550/ARXIV.1906.02569). URL: <https://arxiv.org/abs/1906.02569>.
- [2] Jordi Armengol-Estapé y col. “Are Multilingual Models the Best Choice for Moderately Under-resourced Languages? A Comprehensive Assessment for Catalan”. En: *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*. Online: Association for Computational Linguistics, ago. de 2021, págs. 4933-4946. DOI: [10.18653/v1/2021.findings-acl.437](https://doi.org/10.18653/v1/2021.findings-acl.437). URL: <https://aclanthology.org/2021.findings-acl.437>.
- [3] Yoshua Bengio, Paolo Frasconi y Patrice Y. Simard. “The problem of learning long-term dependencies in recurrent networks”. En: *IEEE International Conference on Neural Networks* (1993), 1183-1188 vol.3.
- [4] Tom B. Brown y col. *Language Models are Few-Shot Learners*. 2020. DOI: [10.48550/ARXIV.2005.14165](https://doi.org/10.48550/ARXIV.2005.14165). URL: <https://arxiv.org/abs/2005.14165>.
- [5] José Cañete y col. “Spanish Pre-Trained BERT Model and Evaluation Data”. En: *PML4DC at ICLR 2020*. 2020.
- [6] Carrino Casimiro Pio, Costa-jussa Marta R. y Fonollosa Jose A. R. “Automatic Spanish Translation of the SQuAD Dataset for Multilingual Question Answering”. En: *arXiv e-prints*, arXiv:1912.05200v1 (2019), arXiv:1912.05200v1. arXiv: [1912.05200v2](https://arxiv.org/abs/1912.05200v2).
- [7] Danqi Chen y col. *Reading Wikipedia to Answer Open-Domain Questions*. 2017. DOI: [10.48550/ARXIV.1704.00051](https://doi.org/10.48550/ARXIV.1704.00051). URL: <https://arxiv.org/abs/1704.00051>.
- [8] Jacob Devlin y col. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2018. DOI: [10.48550/ARXIV.1810.04805](https://doi.org/10.48550/ARXIV.1810.04805). URL: <https://arxiv.org/abs/1810.04805>.
- [9] Asier Gutiérrez-Fandiño y col. “MarIA: Spanish Language Models”. En: *CoRR* abs/2107.07253 (2021). arXiv: [2107.07253](https://arxiv.org/abs/2107.07253). URL: <https://arxiv.org/abs/2107.07253>.
- [10] Sepp Hochreiter y Jürgen Schmidhuber. “Long Short-Term Memory”. En: *Neural Comput.* 9.8 (nov. de 1997), págs. 1735-1780. ISSN: 0899-7667. DOI: [10.1162/neco.1997.9.8.1735](https://doi.org/10.1162/neco.1997.9.8.1735). URL: <https://doi.org/10.1162/neco.1997.9.8.1735>.
- [11] Jeff Johnson, Matthijs Douze y Hervé Jégou. *Billion-scale similarity search with GPUs*. 2017. DOI: [10.48550/ARXIV.1702.08734](https://doi.org/10.48550/ARXIV.1702.08734). URL: <https://arxiv.org/abs/1702.08734>.
- [12] K Jones, Walker SG y Stephen Robertson. “A probabilistic model of information retrieval: development and comparative experiments: Part 2”. En: *Information Processing Management* 36 (nov. de 2000), págs. 809-840. DOI: [10.1016/S0306-4573\(00\)00016-9](https://doi.org/10.1016/S0306-4573(00)00016-9).

- [13] Daniel Jurafsky. *Chapter 25: Question Answering*. 2019. URL: https://web.stanford.edu/~jurafsky/slp3/old_oct19/25.pdf.
- [14] Vladimir Karpukhin y col. *Dense Passage Retrieval for Open-Domain Question Answering*. 2020. DOI: [10.48550/ARXIV.2004.04906](https://doi.org/10.48550/ARXIV.2004.04906). URL: <https://arxiv.org/abs/2004.04906>.
- [15] Yinhan Liu y col. *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. 2019. DOI: [10.48550/ARXIV.1907.11692](https://doi.org/10.48550/ARXIV.1907.11692). URL: <https://arxiv.org/abs/1907.11692>.
- [16] Christopher Olah. *Understanding LSTM Networks*. 2015. URL: <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>.
- [17] Pranav Rajpurkar y col. *SQuAD: 100,000+ Questions for Machine Comprehension of Text*. 2016. DOI: [10.48550/ARXIV.1606.05250](https://doi.org/10.48550/ARXIV.1606.05250). URL: <https://arxiv.org/abs/1606.05250>.
- [18] Nils Reimers e Iryna Gurevych. *Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks*. 2019. DOI: [10.48550/ARXIV.1908.10084](https://doi.org/10.48550/ARXIV.1908.10084). URL: <https://arxiv.org/abs/1908.10084>.
- [19] Mike Schuster y Kuldip Paliwal. "Bidirectional recurrent neural networks". En: *Signal Processing, IEEE Transactions on* 45 (dic. de 1997), págs. 2673-2681. DOI: [10.1109/78.650093](https://doi.org/10.1109/78.650093).
- [20] Jörg Tiedemann. "Parallel Data, Tools and Interfaces in OPUS". En: *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*. Istanbul, Turkey: European Language Resources Association (ELRA), mayo de 2012, págs. 2214-2218. URL: http://www.lrec-conf.org/proceedings/lrec2012/pdf/463_Paper.pdf.
- [21] Ashish Vaswani y col. *Attention Is All You Need*. 2017. DOI: [10.48550/ARXIV.1706.03762](https://doi.org/10.48550/ARXIV.1706.03762). URL: <https://arxiv.org/abs/1706.03762>.
- [22] Ellen Voorhees. "Overview of the TREC 2001 Question Answering Track". En: (jun. de 2002).
- [23] Fengbin Zhu y col. *Retrieving and Reading: A Comprehensive Survey on Open-domain Question Answering*. 2021. DOI: [10.48550/ARXIV.2101.00774](https://doi.org/10.48550/ARXIV.2101.00774). URL: <https://arxiv.org/abs/2101.00774>.

APÉNDICE A

OBJETIVOS DE DESARROLLO SOSTENIBLE

Objetivos de desarrollo sostenible	Alto	Medio	Bajo	No procede
ODS1. Fin de la pobreza				X
ODS2. Hambre cero				X
ODS3. Salud y bienestar				X
ODS4. Educación de calidad				X
ODS5. Igualdad de género				X
ODS6. Agua limpia y saneamiento				X
ODS7. Energía asequible y no contaminante				X
ODS8. Trabajo decente y crecimiento económico				X
ODS9. Industria, innovación e infraestructura		X		
ODS10. Reducción de las desigualdades	X			
ODS11. Ciudades y comunidades sostenibles				X
ODS12. Producción y consumo responsable				X
ODS13. Acción por el clima	X			
ODS14. Vida submarina				X
ODS15. Vida de ecosistema terrestres				X
ODS16. Paz, justicia e instituciones sólidas				X
ODS17. Alianzas para lograr objetivos				X

Con motivo de cumplir los objetivos de desarrollo sostenible en relación al trabajo realizado se considera al objetivo **ODS13. Acción por el clima** el más próximo al resultado conseguido. El uso de modelos preentrenados disponibles en Internet ha descartado la necesidad de invertir recursos computacionales para obtener los mismos resultados, lo que reduce el impacto de la huella de carbono.

Otros dos objetivos que también se asimilan con el resultado final de este proyecto son el objetivo **ODS9. Industria, innovación e infraestructura** y **ODS10. Reducción de las desigualdades**. El desarrollo del sistema de búsqueda de respuestas permite que lenguas minoritarias como el catalán cuenten con los mismos avances tecnológicos que otras lenguas como el español, abriendo paso a la innovación.