EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

Tecnológico Nacional de México

Centro Nacional de Investigación y Desarrollo Tecnológico

Tesis de Doctorado

Redes Neuronales Convolucionales y Recurrentes para la Extracción de Respuestas en Español

presentada por

MC. Alberto Iturbe Herrera

como requisito para la obtención del grado de

Doctor en Ciencias de la Computación

Director de tesis

Dr. Noé Alejandro Castro Sánchez

Codirector de tesis

Dr. Dante Mújica Vargas

Cuernavaca, Morelos, México. Noviembre de 2023.

cenidet
Centro Nacional de Investigación y Desarrollo Tecnológico

EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

Centro Nacional de Investigación y Desarrollo Tecnológico
Departamento de Ciencias Computacionales

Tecnológico Nacional de México

Centro Nacional de Investigación y Desarrollo Tecnológico
Departamento de Ciencias Computacionales

REDES NEURONALES CONVOLUCIONALES Y RECURRENTES PARA LA EXTRACCIÓN DE RESPUESTAS EN ESPAÑOL

Presentada por
M.C. Alberto Iturbe Herrera

Como requisito para la obtención del grado:
Doctorado en Ciencias de la Computación

Director:	**Dr. Noé Alejandro Castro Sánchez**
Co-Director:	**Dr. Dante Mújica Vargas**
Revisores internos:	Dr. Juan Gabriel González Serna Dr. Nimrod González Franco Dr. Máximo López Sánchez
Revisor externo:	Dra. Helena Montserrat Gómez Adorno

Mayo, 2023
Cuernavaca, Morelos

Centro Nacional de Investigación y Desarrollo Tecnológico
Departamento de Ciencias Computacionales

Cuernavaca, Morelos, 04/enero/2023

ASUNTO: OFICIO DE LIBERACIÓN DE PUBLICACIÓN DE ARTÍCULO

M.C. ALBERTO ITURBE HERRERA
ESTUDIANTE DEL PROGRAMA DE DOCTORADO
EN CIENCIAS DE LA COMPUTACIÓN.
PRESENTE

Después de haber evaluado los resultados derivados de su Examen Predoctoral, en el cual presentó el artículo de investigación abajo listado, como producto de su tema de tesis doctoral **"REDES NEURONALES CONVOLUCIONALES Y RECURRENTES PARA LA GENERACIÓN AUTOMÁTICA DE RESPUESTAS EN ESPAÑOL"** tenemos el honor de expedirle la presente Liberación de Publicación de Artículos para obtener el grado de Doctor en Ciencias de la Computación.

RULE-BASED SPANISH MULTIPLE QUESTION REFORMULATION AND THEIR CLASSIFICATION USING A CONVOLUTIONAL NEURAL NETWORK

Reciba un cordial saludo.

A T E N T A M E N T E
"Excelencia en Educación Tecnológica"
"Educación Tecnológica al Servicio de México"

DR. NOÉ ALEJANDRO CASTRO SÁNCHEZ
CENIDET

DR. DANTE MÚJICA VARGAS
CENIDET

DR. JUAN GABRIEL GONZÁLEZ SERNA
CENIDET

DR. NIMROD GONZÁLEZ FRANCO
CENIDET

DR. MÁXIMO LÓPEZ SÁNCHEZ
CENIDET

DRA. HELENA GÓMEZ ADORNO
UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

C.c.p.: MTI. María Elena Gómez Torres / Jefa del Depto. de Servicios Escolares
Dr. Carlos Manuel Astorga Zaragoza / Subdirector Académico
Expediente

EDUCACIÓN | TECNOLÓGICO NACIONAL DE MÉXICO
13 NOV 2023
CENTRO NACIONAL DE INVESTIGACIÓN Y DESARROLLO TECNOLÓGICO
SERVICIOS ESCOLARES
RECIBIDO

EDUCACIÓN

TECNOLOGICO NACIONAL DE MÉXICO

SEP TecNM CENTRO NACIONAL DE INVESTIGACIÓN Y DESARROLLO TECNOLÓGICO
RECIBIDO
16 NOV 2023
SUBDIRECCIÓN ACADÉMICA

Centro Nacional de Investigación y Desarrollo Tecnológico
Departamento de Ciencias Computacionales

ESC\FORDOC09

Cuernavaca, Morelos, 13/noviembre/2023

ASUNTO: ACEPTACIÓN DEL TRABAJO DE TESIS DOCTORAL

MARÍA YASMÍN HERNÁNDEZ PÉREZ
JEFE DEL DEPARTAMENTO DE CIENCIAS COMPUTACIONALES
PRESENTE

Los abajo firmantes, miembros del Comité Tutorial de la Tesis Doctoral del alumno **ALBERTO ITURBE HERRERA** manifiestan que después de haber revisado su trabajo de tesis doctoral titulado **"REDES NEURONALES CONVOLUCIONALES Y RECURRENTES PARA LA GENERACIÓN AUTOMÁTICA DE RESPUESTAS EN ESPAÑOL"**, realizado bajo la dirección de **Noé Alejandro Castro Sánchez,** el trabajo se ACEPTA para proceder a su impresión.

ATENTAMENTE
"Excelencia en Educación Tecnológica®
"Educación Tecnológica al Servicio de México"

NOÉ ALEJANDRO CASTRO SÁNCHEZ
CENIDET

DANTE MÚJICA VARGAS
CENIDET

JUAN GABRIEL GONZÁLEZ SERNA
CENIDET

NIMROD GONZÁLEZ FRANCO
CENIDET

MÁXIMO LÓPEZ SÁNCHEZ
CENIDET

HELENA GÓMEZ ADORNO
UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

C.c.p.: María Elena Gómez Torres / Jefa del Depto. de Servicios Escolares
Carlos Manuel Astorga Zaragoza / Subdirector Académico
Expediente

EDUCACIÓN
16 NOV 2023

cenidet Centro Nacional de Investigación y Desarrollo Tecnológico	SOLICITUD DE REVISIÓN DE DOCUMENTO DE TESIS DOCTORAL	Código: CENIDET-AC-006-D18
		Revisión: O
	Referencia a la Norma ISO 9001:2008 7.1, 7.2.1, 7.5.1, 7.6, 8.1, 8.2.4	Página 1 de 1

Cuernavaca, Mor., a 04 de enero de 2023

DR. JUAN GABRIEL GONZÁLEZ SERNA
JEFE DEL DEPARTAMENTO DE CIENCIAS COMPUTACIONALES
PRESENTE

AT´N: DR. JUAN CARLOS ROJAS PÉREZ
PRESIDENTE DEL CLAUSTRO DOCTORAL DE CIENCIAS DE LA COMPUTACIÓN

Por este medio le solicito atentamente la revisión de mi documento de tesis de doctorado cuyo título es:

"REDES NEURONALES CONVOLUCIONALES Y RECURRENTES PARA LA GENERACIÓN AUTOMÁTICA DE RESPUESTAS EN ESPAÑOL"

Adjunto a la presente le entrego cinco ejemplares del documento de tesis, avalada por mi(s) director(es).

Atentamente

ALBERTO ITURBE HERRERA
No. de control D15CE083

Vo.Bo.

DR. NOÉ ALEJANDRO CASTRO SÁNCHEZ
Director de tesis

DR. DANTE MÚJICA VARGAS
Codirector de tesis

C.p. Departamento de Servicios Escolares
Expediente

EDUCACIÓN | TECNOLÓGICO NACIONAL DE MÉXICO
13 NOV 2023
CENTRO NACIONAL DE INVESTIGACIÓN Y DESARROLLO TECNOLÓGICO
SERVICIOS ESCOLARES
RECIBIDO

RESUMEN

Actualmente, los mecanismos de búsqueda permiten al usuario identificar documentos o enlaces que contengan información relacionada a partir de una consulta determinada. Sin embargo, este proceso implica un gran consumo de tiempo cuando se trata de preguntas como: *¿Quién ganó la mayor cantidad de medallas individuales en los Juegos Olímpicos 2012?* A la cual se esperaría obtener la siguiente respuesta: **Michael Phelps**.

Buscando satisfacer la necesidad de obtener respuestas concisas surgen los sistemas pregunta-respuesta (del inglés *Question-Answering*), que permiten al usuario realizar preguntas y obtener la información en palabras relevantes y concisas. Ésta ha sido un área de interés desde 1961, cuando se desarrolla el primer sistema de pregunta respuesta hasta la actualidad implementando arquitecturas de aprendizaje profundo. Sin embargo, estos sistemas requieren una basta cantidad de información en las diferentes etapas necesarias para obtener una respuesta final. La diversidad en esta área de investigación va desde el idioma hasta el dominio de conocimiento de que éstos abarcan. Incluso sobre en el tipo de información sobre el cual se pretende extraer la respuesta, desde texto plano hasta información tabulada.

En esta investigación nos enfocamos en la resolución de esta tarea sobre el dominio de Ciencias Computacionales o áreas afines en el idioma español utilizando dos arquitectura de redes neuronales artificiales: Redes Neuronales Recurrentes *Bi-LSTM* y Redes Neuronales Convolucionales. El primer reto de este investigación fue la construcción de la base de conocimiento sobre la cual se buscan las respuestas, este primer módulo se logró construir con la ayuda una herramienta denominada WikiExtractor, que permite extraer toda la base de conocimiento de la enciclopedia en línea Wikipedia. Sin embargo, esta herramienta no permite introducir algún criterio de búsqueda para seleccionar un subconjunto de información en específico, por consecuente se desarrollaron algoritmos para clasificar los artículos extraídos utilizando palabras clave relacionadas al área de conocimiento anteriormente mencionada. Con la ayuda de este algoritmo cada artículo obtuvo una puntuación que determina si pertenece o no al dominio de interés en función de un umbral definido.

El siguiente reto fue la construcción de los modelos de Redes Neuronales Artificiales. Tomando en cuenta que estas arquitecturas utilizan datos numéricos como entrada fue necesario crear vectores de valor real denominados *Word Embeddings* utilizando la base de conocimiento anteriormente mencionada. La primera arquitectura desarrollada fue la Red Neuronal Convolucional. Esta arquitectura, comúnmente utilizada para la clasificación de imágenes ha logrado resultados prometedores en el procesamiento de lenguaje natural, lo que permitió crear un modelo capaz de clasificar las preguntas en cinco clases según el foco de éstas. Previo a esto se desarrolló la red *Bi-LSTM* enfocada en codificar las preguntas y pasajes candidatos así como la recuperación de las respuestas candidatas en función del análisis bi-direccional de los pasajes codificados.

Finalmente, con el objetivo de evaluar los modelos desarrollados y de acuerdo con la literatura se seleccionó el corpus SQuAD 1.1. Este corpus fue necesario traducirlo al idioma español con la ayuda de herramientas en línea. Esto permitió realizar una colaboración con el IMAAS de la UNAM para realizar un procesamiento manual de la traducciones con errores presentes en éstas, que mejoraron significativamente los errores generados con anterioridad. Utilizando un fragmento del corpus traducido en el dominio correspondiente se lograron observar resultados bastante interesantes y alentadores para el desarrollo de modelos de

aprendizaje profundo en el idioma español. De igual forma, el área de crecimiento y mejora sobre el actual sistema nos permitió identificar el gran impacto de los modelos de aprendizaje profundo en el Procesamiento de Lenguaje Natural.

Palabras clave: Procesamiento de Lenguaje Natural, Redes Neuronales Recurrentes, Redes Neuronales Convolucionales.

ABSTRACT

Currently, search mechanisms allow the user to identify documents or links that contain information related to a specific query. However, this process is very time consuming when it comes to questions like: *Who won the most individual medals at the 2012 Olympic Games?* To which you would expect to get the following answer: **Michael Phelps**.

Seeking to satisfy the need to obtain concise answers, question-answering systems arise, which allow the user to ask questions and obtain information in relevant and concise words. This has been an area of interest since 1961, when the first question-and-answer system was developed, to the present day by implementing deep learning architectures. However, these systems require a vast amount of information in the different stages necessary to obtain a final answer. The diversity in this area of research goes from the language to the domain of knowledge that they cover. Even on the type of information from which the answer is intended to be extracted, from plain text to tabulated information.

In this research we focus on the resolution of this task on the domain of Computer Science or related areas in the Spanish language using two architectures of artificial neural networks: recurrent neural networks *Bi-LSTM* and convolutional neural networks. The first challenge of this system was the construction of the knowledge base on which the answers are sought, this first module was built with the help of a tool called WikiExtractor, which allows extracting the entire knowledge base of the Wikipedia online encyclopedia . However, this tool does not allow the introduction of any search criteria to select a specific subset of information, therefore, algorithms were developed to classify the extracted articles using keywords related to the aforementioned area of knowledge. With the help of this algorithm, each article obtained a score that determines whether or not it belongs to the domain of interest based on a defined threshold.

The next challenge was the development of the artificial neural network models. Taking into account that these architectures use numerical data as input, it was necessary to create real value vectors called *Word Embeddings* using the aforementioned knowledge base. The first architecture developed was the convolutional neural network. This architecture, commonly used for image classification, has achieved promising results in natural language processing, which allowed the creation of a model capable of classifying questions into five classes according to their focus. Prior to this, the *Bi-LSTM* network was developed focused on encoding candidate questions and passages as well as the recovery of candidate answers based on the bi-directional analysis of the encoded passages.

Finally, in order to evaluate the developed models and in accordance with the literature, the SQuAD 1.1 corpus was selected. It was necessary to translate this corpus into Spanish with the help of online tools. However, to improve the translations carried out, we had the support of undergraduate students from the IMAAS of UNAM, who significantly improved the errors generated previously. Using a fragment of the corpus translated in the corresponding domain, quite interesting and encouraging results were observed for the development of deep learning models in the Spanish language. Similarly, the area of growth and improvement over the current system allowed us to identify the great impact of deep learning models on natural language processing.

Keywords: Natural Language Processing, Recurrente Neural Networks, Convolutional Neural Networks.

A mis padres, familiares, novia y compañeros que formaron parte de este proceso, por el apoyo emocional, académico y personal...

Gracias

Agradecimientos

A Conahcyt por el apoyo económico proporcionado durante estos años y a Cenidet por haberme brindado la oportunidad de realizar el posgrado en sus instalaciones.
Al Dr. Noé Alejandro Castro Sánchez y al Dr. Dante Mújica Vargas por haber dirigido esta investigación, así como al resto de Doctores que participaron en este trabajo con su tutela y consejos.
A mis padres, quienes siempre han estadoa mi lado brindando su apoyo moral y económicamente de forma incondicional en todo momento, que gracias a ellos soy la persona que soy hoy en día, por los valores que desde la infancia han inculcado en mí, porque gracias a ellos he tenido una vida plena; por y para ustedes es este logro más.
A mi novia, Vianey Castañeda por el apoyo constante e incondicional en esta etapa que concluye.
A mi hermano Daniel, quien es, además de mis padres, mi pilar, mi luz y mi fuerza. Gracias por todo.
A mi tía Marina y a mi abuela Mercedes, dos de las mujeres más importantes en mi vida junto a mi madre. Gracias a ustedes por formar parte de cada momento y cada logro en mi vida.
A mis amigos, en especial a Fernando Patiño por haber hecho de estos cuatro años un tiempo increíble y ameno. Por su compañía y apoyo.
A Dios, por permitirme alcanzar esta meta, por darme la fuerza necesaria en momentos difíciles, por permitirme compartir este logro en compañía de mis seres queridos.

Tabla de Contenido

Índice de Tablas

Índice de Figuras

Capítulo 1

Introducción

El Procesamiento del Lenguaje Natural (PLN), o bien por sus siglas en inglés NLP, es un subconjunto de un área denominada procesamiento del habla y del lenguaje que tuvo origen en la década de 1940 de la mano con el desarrollo de la informática lo que permitió la construcción de la teoría del lenguaje formal que modela el lenguaje en estructuras y reglas cada vez más complejas. Por ejemplo, el alfabeto es la estructura más simple, ya que es una colección de letras que pueden formar cadenas llamadas palabras.

De acuerdo con Beysolow [1], un lenguaje natural es aquel que tiene una gramática regular, libre de contexto y formal. Esta es la razón principal por la cual se comparten similitudes con la lingüística computacional centrada en modelar el lenguaje utilizando modelos basados en reglas. Además del desarrollo de las ciencias de la computación, los avances de la inteligencia artificial también desempeñaron un papel la comprensión continua del PLN [1]. Básicamente, el PLN es un campo que aborda varias formas en que las computadoras pueden comprender el lenguaje humano [2]. El PLN comprende una amplia cantidad de tareas, entre las cuales destacan pero no se limitan a:

- Recuperación de información
- Reconocimiento de entidad nombrada
- Clasificación de documentos
- Extracción de palabras clave
- Traducción automática
- Etiquetado de partes del discurso
- Etiquetado de roles semánticos
- Desambiguación de sentido de palabra
- Corrección de errores gramaticales
- Similitud textual semántica
- Resumen de texto/Resumen de reunión
- Comprensión de lectura
- Preguntas y respuestas
- Generación de preguntas
- Subtítulos de imágenes
- Detección de noticias falsas/Detección de discurso de odio
- Generación de texto
- Análisis de sentimientos/emociones
- Conversión de voz a texto

Los primeros enfoques del PLN se basaban en reglas y plantillas codificadas en los sistemas, es decir, se creaban los patrones y reglas sobre cómo una palabra o frase en un idioma deben traducirse a otra en el idioma objetivo, o bien plantillas para extraer información de los textos. Estos enfoques tienen una clara ventaja: se basan en el conocimiento experto confiable que

se les aplica. Y, en algunos casos, funcionan bien, tal es el ejemplo de ELIZA [3], el primer chatbot desarrollado.

Sin embargo, el lenguaje humano es diverso, ambiguo y creativo, atributos que vuelven imposible que los enfoques basados en reglas tengan todas las plantillas, posibilidad y excepciones del lenguaje. Ejemplo de esto fue la traducción automática en la década de 1950. Sin embargo, en la década de 1980 surgen los primeros modelos estadísticos, lo que permitió analizar los datos del lenguaje con las estadísticas derivadas del texto en conjunto con el aprendizaje automático. Los avances logrados se vieron impulsados de forma progresiva en las distintas tareas del PLN principalmente por el desarrollo de corpus paralelos, la llegada de internet y por último pero no menos importante el crecimiento exponencial del hardware que permitió que el aprendizaje automático se llevara al siguiente nivel de arquitecturas conocidas como *Deep Learning* tal y como se muestra en la Figura 1.1. Esto ha permitido que los enfoques iniciales como los nuevos logren complementarse mutuamente de ser necesario o bien lograr buenos resultados independientemente.

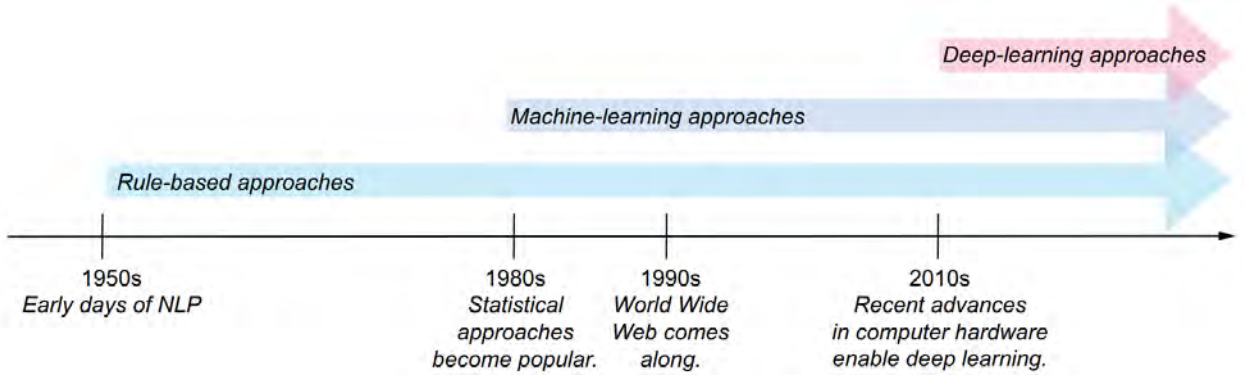


Figura 1.1: Línea de tiempo de los enfoques para el PLN [2]

Tomando en cuenta que ambos enfoques forman parte de la actualidad en la resolución de problemas del PLN es importante destacar las diferencias entre éstos. Los enfoques basados en reglas definen un conjunto de reglas muy precisas pero rígidas, que naturalmente son dificiles de adaptar. Por otra parte, los enfoques estadísticos se centran en determinar qué es lo correcto y qué es lo incorrecto lo que permite tener cierta flexibilidad acerca de las predicciones. Sin embargo, estos tienen la desventaja de la basta cantidad de información requerida y la calidad de la misma para obtener resultados aceptables.

En esta tesis de doctorado se introduce un sistema de pregunta-respuesta basado en el enfoque del aprendizaje profundo capaz de procesar preguntas múltiples para su reformulación y obtención de respuestas en el dominio de las Ciencias Computacionales o áreas afines en español.

1.1. Planteamiento del problema

Actualmente, los mecanismos de búsqueda permiten al usuario identificar documentos o enlaces que contengan información relacionada a partir de una consulta determinada. Sin embargo, este proceso implica un gran consumo de tiempo cuando se trata de preguntas como: *¿Quién ganó la mayor cantidad de medallas individuales en los Juegos Olímpicos 2012?* A la cual se esperaría obtener la siguiente respuesta: **Michael Phelps**. Tomando en cuenta esto, los sistemas pregunta-respuesta permiten al usuario realizar preguntas y obtener

la información en palabras relevantes y concisas. Sin embargo, la mayoría de estos sistemas se enfocan en el idioma inglés. Un estudio realizado por [4] en 2016 expone la diversidad lingüística durante dicho año, en el cual, del 69 % de los trabajos extensos de ACL solo evalúan el idioma inglés, llegando en ciertos casos a darse por hecho que el inglés es una especie de *default* en el PLN.

A esto se suman aspectos como las múltiples formas en las que un ser humano puedo generar una consulta, por ejemplo, al referirse una fecha se pueden tener las siguientes variantes: ¿cuándo se inventó el primer balón de fútbol?, ¿en qué fecha se inventó el primer balón de fútbol?, fecha en la que se inventó el primer balón de fútbol, entre otras. Algunas formas suelen ser más comunes que otras, sin embargo, tales variaciones son posibles, así como aquellas oraciones donde se carece de una palabra interrogativa y de signos de puntuación. De esta forma se podrían continuar mencionando los fenómenos del lenguaje en la generación de consultas como el uso de sinónimos.

A lo largo de la literatura se ha logrado observar que los modelos de inteligencia artificial han obtenido resultados que han igualado e incluso superado a las técnicas convencionales de Procesamiento de Lenguaje Natural. Sin embargo, los modelos como las Redes Neuronales Artificiales han sido utilizadas únicamente para resolver distintas tareas del área sólo en el idioma inglés.

Sin embargo, una de las tareas más complejas en dichos sistemas en la interpretación correcta de las preguntas en lenguaje natural y por consecuente obtener la información deseada, generando respuestas candidatas con información irrelevante, además de prolongados tiempos de espera en el proceso de evaluación y obtención de resultados. Como todo sistema de información, ninguna herramienta o técnica es perfecta, lo que puede ocasionar pérdida de información o descarte de datos que son relevantes para el objetivo. A esto se suman los fenómenos del lenguaje natural, como la ambigüedad en los distintos niveles: léxico, sintáctico, pragmático e incluso referencial.

Lo que implica que muchas de las lenguas de mundo no dispongan de herramientas de preprocesamiento o bien, no con la misma eficiencia que en el idioma inglés; así como carecer también de corpus/datasets. Lo que conlleva a que los métodos estado del arte no necesariamente funcionen bien en los escenarios con menores recursos, indistintamente si éstos utilizan técnicas de PLN o de Aprendizaje Profundo. Basado en lo anterior y previo a un periodo de experimentación, los modelos originalmente creados para el idioma inglés muestran un decremento en su eficiencia y en algunos casos a no ejecutarse únicamente realizando el cambio de archivos (*word embeddings* y conjuntos de entrenamiento y prueba). Esto demuestra que no existe una arquitectura de Redes Neuronales independiente del idioma.

Por lo tanto, se desarrollarán los modelos de Red Neuronal Convolucional y Red Neuronal Recurrente capaces de interpretar preguntas realizadas en lenguaje natural en español y obtener la mejor respuesta.

1.2. Objetivos

En esta sección se describen los objetivos generales y específicos de este trabajo de investigación.

1.2.1. Objetivo general

El objetivo de esta tesis doctoral es diseñar y desarrollar un sistema de preguntas y respuestas en español utilizando técnicas de procesamiento de lenguaje natural y aprendizaje profundo. Específicamente, se investigará la utilización de redes neuronales convolucionales (CNN) y redes neuronales recurrentes (RNN) para mejorar la precisión y efectividad del sistema. Los modelos propuestos serán capaces de comprender preguntas en lenguaje natural en español, procesarlas y generar respuestas relevantes y precisas en función de los conjuntos de datos utilizados. Además, se explorará la posibilidad de utilizar técnicas pre-entrenamiento de modelos de lenguaje para mejorar el rendimiento. La evaluación de los modelos se llevará a cabo mediante la utilización de conjuntos de datos relevantes adaptados al español. Se espera que los resultados de esta investigación contribuyan al avance en el campo de procesamiento del lenguaje natural y permitan el desarrollo de modelos más efectivos y precisos para la búsqueda y recuperación de información en español.

1.2.2. Objetivos específicos

- Investigar y analizar los desafíos específicos asociados con la comprensión y generación de respuestas en español en sistemas de preguntas y respuestas basados en inteligencia artificial.
- Implementar una arquitectura de red neuronal convolucional (CNN) y recurrente (RNN) adaptada para el procesamiento de preguntas y respuestas en español.
- Investigar técnicas avanzadas de procesamiento de lenguaje natural para el preprocesamiento de datos en español, incluyendo tokenización, lematización, etiquetado gramatical, entre otros.
- Realizar experimentos utilizando conjuntos adaptados al español para entrenar y evaluar el rendimiento del sistema de preguntas y respuestas propuesto.
- Explorar la utilización de técnicas de pre-entrenamiento de modelos de lenguaje en el contexto de sistemas de preguntas y respuestas en español, con el objetivo de mejorar la precisión y la capacidad de respuesta.
- Evaluar el rendimiento de los modelos desarrollados utilizando métricas estándar de evaluación, como la precisión, la cobertura y la coherencia de las respuestas generadas, entre otras.
- Contribuir al avance del campo de procesamiento del lenguaje natural en español, al proporcionar un modelo de preguntas y respuestas eficaz que pueda utilizarse para la búsqueda y recuperación de información en lenguaje natural.

1.3. Alcances y limitaciones

1.3.1. Alcances

- El sistema procesará preguntas expresadas en lenguaje natural en español.
- Podrán realizarse consultas fuera de línea al disponer de las bases de conocimiento almacenadas localmente.

- Se creará una base de conocimiento en español.
- Se utilizará Wikipedia como base de conocimiento.
- Se adaptarán los datasets de entrenamiento al idioma español.

1.3.2. Limitaciones

- No se realizará reconocimiento del habla.
- No se utilizarán sintetizadores de voz.
- El sistema responderá únicamente preguntas del dominio de computación o áreas afines.
- El sistema no procesará imágenes o videos.
- No se recurrirá a la traducción.

1.4. Justificación

Hoy en día, existe una amplia variedad de sistemas pregunta-respuesta como: aquellos de dominio específico, acotado, enfocados a ciertos tipos de preguntas o tipos de respuestas, así como las distintas técnicas para realizar la extracción de la respuesta. Tal es el área de oportunidad en estos sistemas, que, algunos autores como Kondra y Meçe [5] consideran que éstos pueden llegar a ser el futuro de la búsqueda a través de internet.

Sin embargo, a pesar de la amplia variedad de sistemas pregunta-respuesta, ninguno está exento a los problemas mencionados anteriormente, además de que estos no se enfocan en el idioma español, siendo este uno de los más hablados a nivel mundial. Por lo tanto, esta investigación se enfocará en desarrollar/mejorar los algoritmos y técnicas de recuperación de información para generar respuestas a partir de una pregunta en idioma español, utilizando técnicas de Procesamiento de Lenguaje Natural e inteligencia artificial con una arquitectura de hardware intermedia para el desarrollo.

De acuerdo con información proporcionada por [6] el español es hablado por millones de personas en todo el mundo y tiene una presencia significativa en Internet. Es el idioma principal en varios países y muchas comunidades de habla hispana contribuyen al contenido en línea. Se estima que aproximadamente 460 millones de personas hablan español como lengua materna en todo el mundo, siendo este el segundo idioma más hablado en el mundo después del chino mandarín.

Además de utilizar Wikipedia como base de conocimiento en el dominio de computación, informática o afín al área, dada la cantidad de información disponible en el idioma en cuestión y el constante crecimiento e interés en relación a las tecnologías de la información. Además de la posibilidad y facilidad de extraer el contenido mediante técnicas web scrapping tomando en cuenta la homogeneidad de los sitios; similar al análisis realizado por Alarcón et al. [7] enfocado en los infoboxes, o bien, utilizando software como XOWA o WikiExtractor. Esto permite crear bases de conocimiento para distintas tareas de lenguaje natural, teniendo en cuenta que, si bien Wikipedia proporciona información valiosa, puede tener imprecisiones o sesgos ocasionales debido a su modelo de edición abierto. Por lo cual, la tarea de pregunta-respuesta se centrará en la obtención de la respuesta correcta a partir del conjunto de información proporcionado.

1.5. Organización de la tesis

El presente documento está compuesto por cinco capítulos y sección de referencias. El capítulo 2 contiene el marco conceptual en el cual se definen los conceptos fundamentales para la familiarización del lector con el tema de investigación, por ejemplo, Procesamiento de Lenguaje Natural, Redes Neuronales Artificiales, Redes Neuronales Recurrentes, entre otros. De igual forma se presenta el estado del arte elaborado que describe aquellos trabajos que fueron tomados como base para esta investigación. El capítulo 3 describe el método de solución implementado en esta investigación, detallando las distintas aportaciones logradas. Posteriormente, en el capítulo 4 se desglosan los experimentos realizados, así como los resultados obtenidos. Finalmente, en el capítulo 5 se muestran las conclusiones generales de la investigación, así como las propuestas para trabajos futuros.

Capítulo 2

Marco Conceptual

A continuación se describen algunos conceptos que serán de utilidad para la familiarización con el tema de investigación.

2.1. Procesamiento de Lenguaje Natural

El Procesamiento del Lenguaje Natural (PLN), es un subconjunto de un área denominada procesamiento del habla y del lenguaje que tuvo origen en la década de 1940 de la mano con el desarrollo de la informática lo que permitió la construcción de la teoría del lenguaje formal que modela el lenguaje en estructuras y reglas cada vez más complejas. Por ejemplo, el alfabeto es la estructura más simple, ya que es una colección de letras que pueden formar cadenas llamadas palabras.

Un lenguaje natural es aquel que tiene una gramática regular, libre de contexto y formal [1]. Esta es la razón principal por la cual se comparten similitudes entre disciplinas como la lingüística computacional centrada en modelar el lenguaje utilizando modelos basados en reglas. Además del desarrollo de las ciencias de la computación, los avances de la inteligencia artificial también desempeñaron un papel la comprensión continua del PLN [1]. Básicamente, el PLN es un campo que aborda varias formas en que las computadoras pueden comprender el lenguaje humano [2]. El PLN comprende una amplia cantidad de tareas, entre las cuales destacan pero no se limitan a:

- Recuperación de información
- Reconocimiento de entidad nombrada
- Clasificación de documentos
- Extracción de palabras clave
- Traducción automática
- Etiquetado de partes del discurso
- Etiquetado de roles semánticos
- Desambiguación de sentido de palabra
- Corrección de errores gramaticales
- Similitud textual semántica
- Resumen de texto/Resumen de reunión
- Comprensión de lectura
- Preguntas y respuestas
- Generación de preguntas
- Subtítulos de imágenes
- Detección de noticias falsas/Detección de discurso de odio
- Generación de texto
- Análisis de sentimientos/emociones
- Conversión de voz a texto

2.2. Redes Neuronales Artificiales

Según Fausett [8] una Red Neuronal Artificial (ANN, por sus siglas en inglés *Artificial Neural Network*)se define como "un sistema de procesamiento de información que posee ciertas características de rendimiento en común con las redes neuronales biológicas".

Una red neuronal consiste en un amplio número de elementos simples de procesamiento llamados neuronas, unidades, células o nodos. Cada neurona está conectada a otras neuronas por medio de enlaces de comunicación directos, cada uno de éstos con un peso asociado. Los pesos representan la información que será utilizada por la red para resolver un problema determinado [8].

Las redes neuronales pueden ser aplicadas a una gran variedad de problemas como: almacenar y recuperar datos o patrones, clasificar patrones, realizar mapeos generales desde patrones de entrada a patrones de salida, agrupación, o búsqueda de soluciones a problemas de optimización restringidos. Son modelos estadísticos directamente inspirados y parcialmente modelados en redes neuronales biológicas, capaces de modelar y procesar relaciones no lineales entre entradas y salidas en paralelo.

De acuerdo con Goodfellow et al. [9] las redes neuronales de mayor popularidad son las siguientes:

- Redes Neuronales Convolucionales: comúnmente utilizada para el procesamiento de imágenes, segmentación, extracción de características, detección y clasificación de objetos. Sin embargo, recientemente, estas han sido utilizadas en trabajos de clasificación de lenguaje natural.

- Redes Neuronales Recurrentes: utilizadas comúnmente para procesamiento y aprendizaje de datos secuenciales.

2.3. Redes Neuronales Convolucionales

Las redes neuronales convolucionales (CNN, Convolutional Neural Networks) también conocidas como ConvNets son un tipo de redes neuronales feed-forward (FNN) en las cuales, el patrón de conectividad entre las neuronas está inspirada en la corteza visual de un animal. Recientemente, las CNN han demostrado tener un rendimiento sobresaliente en los servicios de búsqueda de imágenes, automóviles conducidos de forma autónoma, clasificación automática de videos, reconocimiento de voz y procesamiento de lenguaje natural [10].

Las CNN comparten ciertas similitudes con las redes neuronales regulares, están formadas por neuronas con pesos que se pueden aprender de los datos. Cada neurona recibe un número de entradas y realiza un producto de puntos. Cuentan con una función de pérdida en la última capa completamente conectada, además utilizan una función no lineal [10]. Una red neuronal regular recibe datos de entrada como un único vector y pasa a través de una serie de capas ocultas. Cada capa oculta consiste en un conjunto de neuronas, donde cada neurona está completamente conectada a todas las otras neuronas en la capa anterior. Dentro de una sola capa, cada neurona es completamente independiente y no comparten ninguna conexión [10].

La última capa totalmente conectada, también llamada capa de salida, contiene puntuaciones de clase. Una CNN simple está compuesta por tres capas, las cuales son: capa de convolución, capa de agrupamiento y la capa completamente conectada [10].

En la Figura 2.1 se muestra la arquitectura de una CNN para la clasificación de oraciones propuesta por Kim [11]. En ésta se muestran tres tamaños de filtro de región: 2, 3 y 4. Donde cada uno de éstos contiene dos filtros adicionales. Cada filtro realiza convolución en la matriz de oraciones y genera mapas de características (de longitud variable). Posteriormente, se realiza 1-max pooling en cada mapa. Con base en esto, se genera un vector de características univariadas a partir de los seis mapas, y las seis características son concatenadas para formar un vector de características para la penúltima capa. La capa final de softmax recibe este vector de características como entrada y lo usa para clasificar la oración.

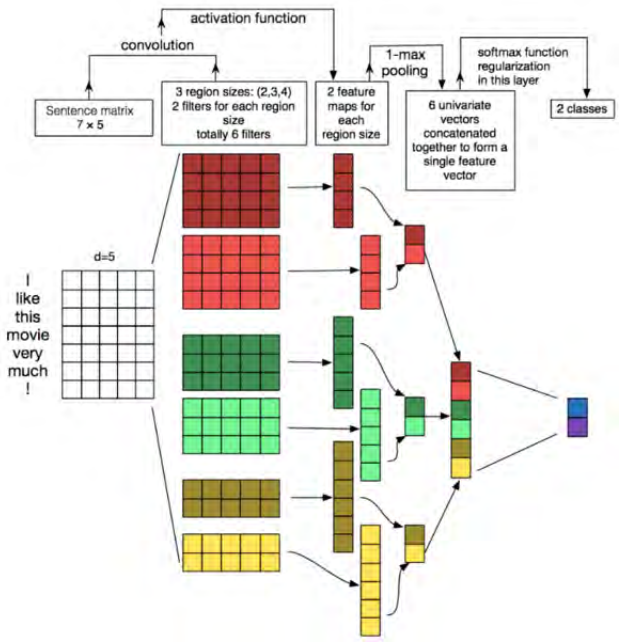


Figura 2.1: Red Neuronal Convolucional para clasificación de oraciones [11]

Generalmente, en las tareas de PLN se utilizan filtros que se desplazan sobre filas completas de una matriz basada en las palabras. Donde, el "ancho" de cada filtro suele ser el mismo que el ancho de la matriz de entrada. La altura o el tamaño de la región pueden variar, pero el deslizamiento de ventanas de dos a cinco palabras por vez.

2.4. Redes Neuronales Recurrentes

Las Redes Neuronales Recurrentes (RNN por sus siglas en inglés, *Recurrent Neural Networks*) son modelos de Redes Neuronales Artificiales inicialmente estudiados por [12–14] para el modelado de series de tiempo. La estructura de la red es similar a la del *Multilayer Perceptron*, con la diferencia en que se permiten conexiones entre unidades ocultas asociadas con un retraso de tiempo.

En esta sección se describen las arquitecturas de RNN más comunes para resolver distintas tareas. Aquí se describen las redes tipo *Long-Short Term Memory* (LSTM) y las *Gated Recurrent Unit* (GRU). Sin embargo, el predecesor de estas es la llamada *Vanilla RNN* (Figura 2.2), la cual tiene como premisa básica analizar cada elemento de una serie de entrada, uno tras otro, y realizar una actualización constante del vector denominado "estado oculto" en cada paso de la secuencia. Este estado oculto se considera el codificador y el vector de estado oculto final se denomina decodificador.

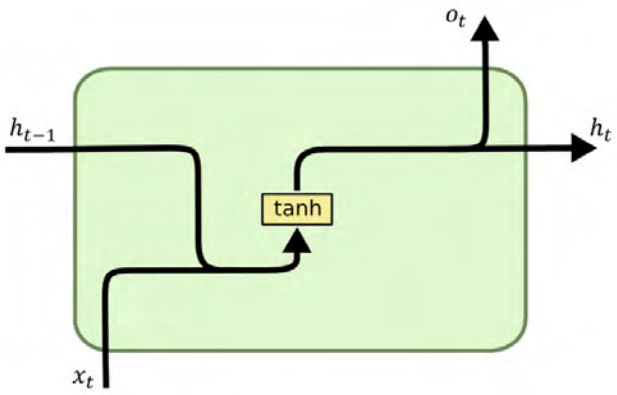


Figura 2.2: Arquitectura de la RNN Vanilla

Sin embargo, este mecanismo secuencial implica ciertas desventajas en el Procesamiento de Lenguaje Natural debido a que existen idiomas como el polaco y húngaro donde el orden de las palabras no importa estrictamente. O bien, los idiomas inglés y español en los cuales es posible cambiar el orden de las palabras según el énfasis que se desea realizar.

Es por eso que las LSTM y GRU han ayudado en gran medida al proporcionar una forma de llevar solo la información relevante de un paso al siguiente a través de varias innovaciones a nivel celular. Las RNN bidireccionales proporcionaron un mecanismo para observar no solo las entradas anteriores, sino también las posteriores antes de generar una salida en un paso de tiempo.

2.4.1. *Long-Short Term Memory* (LSTM)

Este tipo de unidad recurrente fue originalmente propuesta por Hochreiter y Schmidhuber [15] en 1997. También conocidas como LSTM, son capaces de aprender dependencias a largo plazo.

El proceso de aprendizaje comienza con determinar qué información será descartada del estado de la celda. Esta acción se lleva a cabo con una capa sigmoide (denotado como σ) denominada *compuerta de olvido*. Tomando como entradas a h_{t-1} y x_t, se genera una salida con un valor 0 o 1 para cada número en el estado de la celda C_{t-1} (ver ecuación 2.1). Este valor representa: descartar o conservar la totalidad de la información, 0 o 1 respectivamente.

Por ejemplo, para las tareas de Procesamiento de Lenguaje Natural, más concretamente en la generación de lenguaje, el estado de la celda debe incluir el género del sujeto, de modo que se puedan usar los pronombres correctos.

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right) \tag{2.1}$$

Posteriormente, se determina qué información nueva será almacenada en el estado de la celda. Para esto, se aplica una capa sigmoide denominada *capa de puerta de entrada* i_t (ver ecuación 2.2) que se encarga de seleccionar que valores serán actualizados. Inmediatamente, una capa de tangente hiperbólica genera un vector de valores candidatos nuevos, $\tilde{C}_t$ (ver ecuación 2.3), que pueden ser agregados al estado. La salida de ambas capas es combinada para generar una actualización al estado.

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right) \tag{2.2}$$

$$\widetilde{C}_t = \tanh\left(W_C \cdot [h_{t-1}, x_t] + b_C\right) \tag{2.3}$$

Lo siguiente es actualizar el estado de la celda C_{t-1}, al nuevo estado C_t. Esto se realiza mediante la multiplicación del estado anterior por f_t y se suma el producto de i_t y $\tilde{C}_t$ (ver ecuación 2.4). Estos son los nuevos valores candidatos, escalados según cuánto se decida actualizar de cada valor de estado.

$$C_t = f_t * C_{t-1} + i_t * \widetilde{C}_t \tag{2.4}$$

Finalmente, es necesario determinar qué es lo que se mostrará como salida. Esta salida se basará en una versión filtrada del estado de la celda. Para esto se aplica una capa sigmoide (ver ecuación 2.5), la cual, decide qué partes del estado de la celda serán mostradas como salida. Posteriormente, se aplica una tangente hiperbólica al estado en cuestión y se multiplica por la salida de la capa sigmoide (ver ecuación 2.6).

$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right) \tag{2.5}$$

$$h_t = o_t * \tanh\left(C_t\right) \tag{2.6}$$

2.4.1.1. Mecanismos de atención

Si bien las RNN mencionadas anteriormente han logrado erradicar el problema del procesamiento estrictamente secuencial y la selección de información relevante del pasado, estas no logran resolver el problema de la longitud de la secuencia, es decir, mientras más larga sea la entrada, más complicado será para el vector oculto capturar el contexto. Cuantas más actualizaciones se realicen al mismo vector, mayores serán las probabilidades de que se pierdan las entradas y actualizaciones anteriores.

Esto se puede resolver utilizando una arquitectura que utilice todos los estados ocultos a diferencia de solo utilizar el último de estos. Esto evita que el contexto se pierda según pasa el tiempo y se realizan actualizaciones. A esto se le denomina ***mecanismo de atención***

En el modelo de atención propuesto por Bahdanau et al. [16] cada palabra de salida generada no es únicamente una función del estado oculto final, sino una función de todos los estados ocultos. A esto se le define como una operación de atención que consiste en producir un vector distinto que representa todos los estados ocultos del codificador para cada paso de salida del decodificador 2.3. Este otorga diferentes pesos a los distintos estados.

$$p\left(y_i | y_1, \ldots, y_{i-1}, \mathbf{x}\right) = g\left(y_{i-1}, s_i, c_i\right) \tag{2.11}$$

Debe destacarse que, contrario al enfoque *encoder-decoder*, aquí la probabilidad es condicionada a un vector de contexto distinto C_i para cada palabra objetivo y_i (ver ecuación 2.11).

El vector de contexto distinto C_i para un paso de salida es un producto suma de los pesos de atención y de todos los estados ocultos (ver ecuación 2.12). Los pesos de atención para

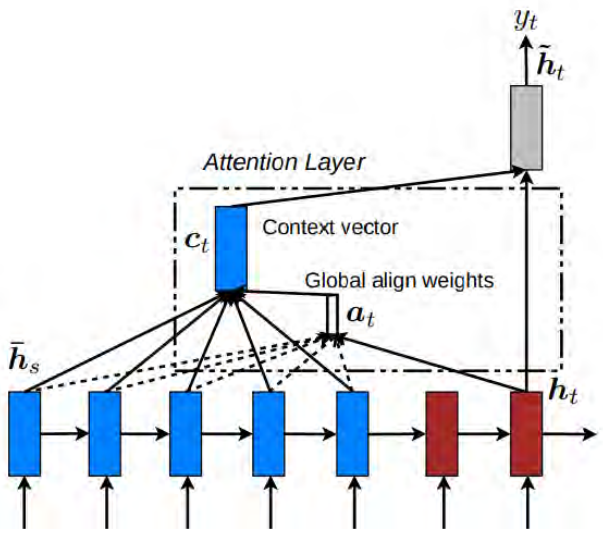


Figura 2.3: Atención Global

cada salida serán diferentes y por lo tanto la suma de los vectores ocultos ponderados es distinta en cada paso de salida. Este vector depende de una secuencia de anotaciones a la cual se le asigna una oración de entrada por el codificador.

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j \tag{2.12}$$

El peso de atención se calcula como un Softmax de ponderaciones alineadas (ver ecuación 2.13). Por otra parte, el puntaje de alineación es una función del estado oculto anterior del decodificador y del codificador (ver ecuación 2.14) [16]. Esta última utiliza una función de alineamiento "a" que es una red *feed forward* entrenada con el modelo.

$$\alpha_{ij} = \frac{\exp\left(e_{ij}\right)}{\sum_{k=1}^{T_x} \exp\left(e_{ik}\right)} \tag{2.13}$$

$$e_{ij} = a\left(s_{i-1}, h_j\right) \tag{2.14}$$

La atención es un mecanismo que proporciona una codificación más completa de la secuencia de entrada a partir de la cual se construye un vector de contexto que luego es utilizado por el decodificador. Este mecanismo permite que el modelo aprenda qué palabras codificadas en la entrada deben ser el foco de atención y qué grado durante la predicción de cada palabra en la secuencia objetivo. Cabe resaltar que a este tipo de atención se le denomina **atención global**, que tiene como inconveniente prestar atención a todas las palabras en el origen para cada palabra objetivo, lo cual, es costoso computacionalmente tornándose complejo en secuencias largas como párrafos o documentos completos. Sin embargo, existen otras variantes de atención.

2.4.1.2. Atención Local

Tomando en cuenta el problema al que se enfrenta la atención global, Luong et. al [17] proponen la atención local, mecanismo inspirado en la compensación entre los modelos de atención suaves y duros propuestos por Xu et al [18] (ver secciones 2.4.1.3 y 2.4.1.4).

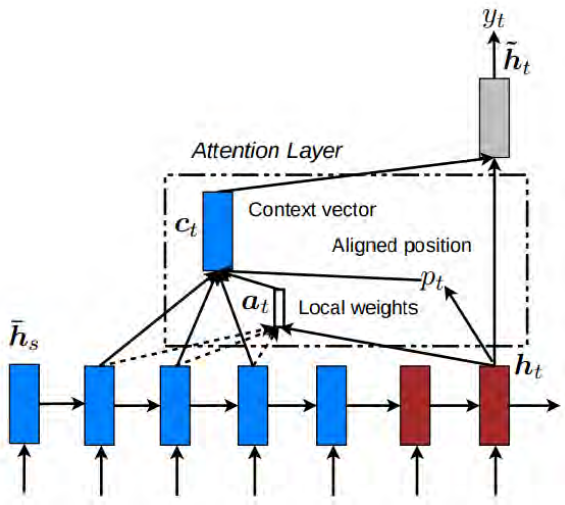


Figura 2.4: Atención Local

En este modelo se predice primero una sola posición alineada p_t para la palabra objetivo actual 2.4. Posteriormente, se utiliza una ventana centrada alrededor de la posición de origen para calcular el vector de contexto $\boldsymbol{c_t}$, el cual, es un promedio ponderado de los estados ocultos en la ventana fijada. Los pesos $\boldsymbol{a_t}$ son inferidos del estado objetivo actual $\boldsymbol{h_t}$ y de los estados origen $\overline{\boldsymbol{h}}_s$ en la ventana.

Este enfoque tiene la ventaja de evitar los altos consumos de recursos computacionales en la atención suave y, al mismo tiempo, es más fácil de entrenar que el enfoque de atención dura. En detalles concretos, el modelo genera primero una posición alineada p_t para cada palabra objetivo en el tiempo t. El vector de contexto c_t se deriva como un promedio ponderado sobre el conjunto de estados ocultos de origen dentro de la ventana $[p_t - D, p_t + D]$, donde D es empíricamente seleccionado. A diferencia de la atención global, el vector de alineación local $\boldsymbol{a}_t$ es de dimensión fija, $\in \mathbb{R}^{2D+1}$.

Además, se implementa un alineamiento predictivo (**local-p**), el cual, en lugar de asumir alineaciones monótonas, predice una posición alineada con la ecuación 2.15.

$$p_t = S \cdot \text{sigmoid}\left(\boldsymbol{v}_p^\top \tanh\left(\boldsymbol{W}_p \boldsymbol{h}_t\right)\right) \tag{2.15}$$

Donde:

- $\boldsymbol{W_p}$ y $\boldsymbol{v}_p$: son los parámetros del modelos que serán aprendidos para predecir las posiciones.
- S: representa la longitud de la cadena de origen. Como resultado del sigmoide, $p_t \in [0, S]$.

Para favorecer los puntos de alineación cercanos a p_t, se colocó una distribución gaussiana a su al rededor (ver ecuación 2.16) y la función de alineación similar a la atención global (ver ecuación 2.17) con una desviación estándar $\sigma = \frac{D}{2}$.

$$\boldsymbol{a}_t(s) = \text{align}\left(\boldsymbol{h}_t, \overline{\boldsymbol{h}}_s\right) \exp\left(-\frac{(s-p_t)^2}{2\sigma^2}\right) \tag{2.16}$$

$$\begin{aligned} \boldsymbol{a}_t(s) &= \text{align}\left(\boldsymbol{h}_t, \overline{\boldsymbol{h}}_s\right) \\ &= \frac{\exp\left(\text{score}\left(\boldsymbol{h}_t, \overline{\boldsymbol{h}}_s\right)\right)}{\sum_{s'} \exp\left(\text{score}\left(\boldsymbol{h}_t, \overline{\boldsymbol{h}}_{s'}\right)\right)} \end{aligned} \tag{2.17}$$

2.4.1.3. Atención dura

Este trabajo propuesto por Xu et al. [18] se enfoca en generar descripciones de imágenes utilizando una RNN tipo LSTM y una CNN, se proponen dos tipos de atención.

La primera de estas denominada atención dura, también conocida como **atención estocástica**, significa aprender a maximizar el vector de contexto $\hat{z}$ a partir de una combinación de una variable codificada *one-hot* $s_{t,i}$ y las características extraídas a_i. Es considerado "atención dura", debido a que se realiza una elección "difícil" en cada característica. Sin embargo, es considerado estocástico ya que $s_{t,i}$ se elige de una distribución mutlinoulli. En este enfoque, la variable de ubicación s_t se presenta como la ubicación donde el modelo decide enfocar la atención al generar la palabra t^{th}.

La puntuación de atención es usada como la probabilidad de que la i-ésima posición sea seleccionada. Aquí podría aplicarse una función argmax, sin embargo, esto no sería diferenciable y por lo tanto se utilizan técnicas más complejas (ver ecuaciones 2.18, 2.19, 2.20).

$$p\left(s_{t,i} = 1 | s_{j<t}, \mathbf{a}\right) = \alpha_{t,i} \tag{2.18}$$

$$\hat{\mathbf{z}}_t = \sum_i s_{t,i} \mathbf{a}_i \tag{2.19}$$

$$\tilde{s_t} \sim \text{ Multinoulli }_L\left(\{\alpha_i\}\right) \tag{2.20}$$

Aprender la atención estocástica requiere muestrear la ubicación de la atención cada vez, en su lugar se puede tomar la expectativa del vector de contexto z^t directamente y formular un modelo de atención determinista calculando un vector de anotación ponderado de atención suave.

2.4.1.4. Atención suave

También descrita por Xu et al. [18] como **atención determinista**, significa aprender maximizando la expectativa del vector de contexto. Es determinista, ya que $s_{t,i}$ no se selecciona de una distribución y es flexible ya que las opciones individuales no están optimizadas, sino toda la distribución.

La puntuación de atención se utiliza como peso en el cálculo del vector de contexto promedio ponderado. Esta es una función diferenciable (ver ecuación 2.21).

$$\mathbb{E}_{p(s_t|a)}\left[\hat{\mathbf{z}}_t\right] = \sum_{i=1}^{L} \alpha_{t,i} \mathbf{a}_i \tag{2.21}$$

2.5. *Word Embeddings*

Los sistemas de procesamiento de imágenes o de audio trabajan con conjuntos de datos multidimensionales muy ricos en información que se codifican como vectores. Estos vectores tendrán las intensidades de los pixeles en los distintos colores, o coeficientes espectrales para audio. Sin embargo, cuando se trata de Procesamiento de Lenguaje Natural, las palabras se tratan como símbolos individuales y discretos. Por ejemplo, podríamos representar "sol" con id343 y "luna" con id432. Las codificaciones son arbitrarias y no proporcionan información sobre las relaciones que pueden existir entre las distintas entidades. Peor aún, la representación como id discretos tiene como consecuencia una dispersión de los datos importantes lo que implica que los entrenamientos necesiten más datos para ofrecer buenos modelos [19].

Los *Word Embeddings*, vectores de palabras o vectores de valor real tienen como objectivo extraer el verdadero significado de una palabra en una oración, donde el significado es la idea o la representación que transmite una palabra. Dado que el objetivo principal del PLN es lograr un desempeño similar al del ser humano en tareas linguísticas surge la necesidad de explorar formas en cómo representar palabras de forma comprensible para los algoritmos. Con estos modelos, palabras similares como *rana* y *sapo* obtienen valores numéricos similares debido a la similitud de sus contextos y/o relaciones, en comparación a palabras como *gato* y *volcán* [19].

En esta sección se describen los tres modelos más explorados en la literatura para la generación de *Word Embeddings*:

2.5.1. Word2Vec

Un modelo particularmente eficiente desde el punto de vista computacional basado en la literatura es **Word2Vec** [20]. Este modelo se encuentra disponible de dos formas: *Continuous Bag-of-Words* (CBOW) o el modelo *Skip-Gram*. Este último suele funcionar mejor que CBOW.

El modelo *Skip-Gram* puede definirse como: dado un conjunto de frases (corpus) el modelo analiza las palabras en cada oración y trata de usar cada una de estas para predecir que palabras serán vecinas. Por ejemplo, la palabra "Caperucita" le seguirá "Roja" con más probabilidad que cualquier otra.

Este modelo utiliza una Red Neuronal Artificial con una única capa oculta. Dado el vocabulario generado a partir del corpus, se entrena esta red neuronal con cada una de las oraciones, para que cada palabra obtenga la probabilidad de ser vecina de todos los elementos disponibles en el corpus.

Previo al entrenamiento de la red, es necesario conservar el modelo generado debido a la importancia de los pesos obtenidos en la capa oculta, ya que estos son los ***Word Vectors*** que se buscan aprender, de ahí el nombre *Word 2 Vector*, es decir, la conversión de palabras a vectores.

Uno de los parámetros de entrenamiento de mayor importancia en este modelo es ***Window Size***, el cual, determina el número de palabras de cada contexto. Este es utilizado para delimitar la selección de los pares de palabras en función del tamaño asignado. A continuación, la Figura 2.5 ejemplifica la forma en cómo se construyen tales pares con tamaño de ventana igual a dos. La entrada de la red neuronal será una palabra representada como un ***one-hot vector***, es decir, un vector con tantas posiciones como el tamaño del vocabulario. Por ejemplo, si queremos representar la palabra "sol" de un vocabulario con 10,000 palabras, se utiliza un vector de esta dimensión con cero en todas las posiciones menos la correspondiente

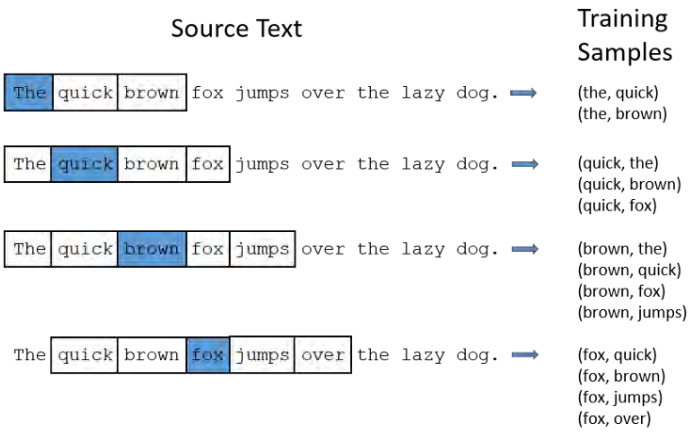


Figura 2.5: Ejemplo de construcción de pares con window size = 2

a la palabra "sol" que tendrá un uno [20].

La salida de la red neuronal será otro vector con 10,000 posiciones que representará las probabilidades de que cada una de las palabras sean vecinas de la palabra representada en la entrada. Dentro de las características de mayor relevancia la capa oculta no tiene función de activación, y en cambio la capa de salida aplica una función Softmax. Cabe resaltar que, aunque en el entrenamiento tanto la entrada como el valor esperado son ***one-hot vectors***, la salida es una distribución de probabilidades [20]. En el diagrama anterior, se está entrenado vectores de palabras con 300 características. La matriz de representación tendrá por tanto 10,000 filas (una por palabra) y 300 columnas (una por neurona). El objetivo de todo el ejercicio es aprender los pesos de esta capa oculta, ya que:

- La representación *one-hot* es un vector de 1x10,000.
- Si se multiplica el vector mencionado por la matriz (10,000x300) se obtiene una representación de 1x300 que efectivamente consiste en la selección de la fila correspondiente al único uno presente en la representación one-hot.

Para que la capa de salida cumpla con lo ya mencionado, se utiliza un clasificador Softmax (Figura 2.6), que consigue que todos los valores estén entre 0 y 1 y que la suma de todos los valores sea 1, es decir, que sea una distribución de probabilidades.

2.5.2. FastText

FastText [21] es una biblioteca escrita en C++ para el aprendizaje eficiente de representaciones de palabras y clasificación de oraciones. FastText permite entrenar representaciones supervisadas y no supervisadas de palabras y oraciones. Estas pueden ser usadas para distintas aplicaciones de compresión de datos como características en modelos adicionales, para selección de candidatos o como inicializadores para el aprendizaje por transferencia.

Utiliza el entrenamiento continuo de palabras (CBOW, por sus siglas en inglés) o los modelos *Skip-gram* usando muestreo negativo, softmax o funciones de pérdida jerárquica Softmax. Esta herramienta logra un bueno rendimiento para las tareas ya mencionadas, pero también en el caso de palabras raras haciendo uso de información a nivel de carácter.

Cada palabra se representa como una bolsa de n-gramas de caracteres además de la palabra en cuestión. A cada n-grama se agregan <, > como símbolos de límite con el objetivo de

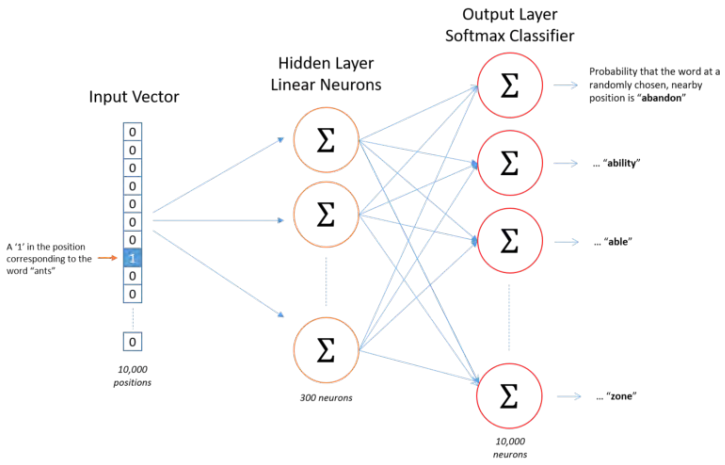


Figura 2.6: Ejemplo de construcción de pares con window size = 2

distinguirlos de otras palabras. Esto permite preservar el significado de palabras más cortas que pueden aparecer, además de capturar el significado de los sufijos y prefijos de estas.

FastText inicializa un par de vectores para realizar un seguimiento de la información de entrada, *word2int_* y *words_*. El primero de estos se indexa en el hash de la palabra y almacena el índice entero secuencial en el segundo arreglo.

El vector *word2int_* es de tamaño MAX_VOCAB_SIZE = 30000000; Este número es comúnmente utilizado en la literatura. Sin embargo, este puede ser modificado según las dimensiones del corpus a utilizar mejorando el rendimiento. El índice para la matriz es el valor de una cadena para el hash de enteros, y es un número único entre 0 y la variable anteriormente mencionada. En caso de coincidir, este se incrementa hasta encontrar un ID único para asignarlo a la palabra.

Para llevar a cabo el entrenamiento, se inicializan los vectores de entrada y los ocultos, así como varios subprocesos de entrenamiento. El número de estos se controla mediante el argumento *-thread*. Los detalles acerca de los distintos argumentos en FastText son descritos en [21] o directamente en el sitio de GitHub. Todos los hilos de entrenamiento tienen un puntero compartido a las matrices para los vectores ya mencionados. Todos los subprocesos se leen desde el archivo de entrada, actualizando el modelo con cada línea de entrada al momento que esta es leída, es decir, el descenso del gradiente estocástico con un tamaño de lote 1.

Una línea de entrada se trunca si se encuentra el carácter de nueva línea o si el conteo de palabras que han sido leídas alcanza el tamaño máximo. Esto se establece a través de MAX_LINE_SIZE, con un valor por defecto de 1024.

Tanto la bolsa de palabras continua como el modelo *Skip-gram* actualizan las ponderaciones para un contexto de tamaño entre una distribución aleatoria uniforme entre 1 y el valor determinado por $-ws$, es decir, el tamaño de la ventana es estocástico.

El vector objetivo para la función de pérdida se calcula a través de una suma normalizada de todos los vectores de entrada. Estos son la representación vectorial de la palabra original y todos los n-gramas de esa palabra. A su vez, se calcula la pérdida que establece los pesos para el paso hacia adelante (*Feed-Forward*), los cuales se propagan en su camino hasta los vectores para la capa de entrada en el paso de propagación hacia atrás (*Back-Forward*). Este ajuste de pesos del vector de entrada es el que permite aprender representaciones, las cuales, maximizan la similitud de ocurrencia. La tasa de aprendizaje $-lr$ afecta a cada instancia en

particular a los pesos.

Los pesos de entrada del modelo, de la capa oculta junto con los argumentos pasados se guardan en el formato .bin y el indicador -saveOutput controla si también se genera un archivo .vec que contiene vectores para la capa oculta en el formato de archivo word2vec.

Capítulo 3

Estado del arte

A lo largo de esta investigación se han identificado numerosos trabajos en el area del *Question-Answering*. Tomando en cuenta que esta tarea ha sido ampliamente investigada desde 1961, se realizó un análisis de las distintas investigaciones lo que permitió clasificar estos trabajos basado en el tipo de técnicas utilizadas. Por ejemplo, trabajos como [22–82] llevan a cabo la tarea de pregunta-respuesta implementando técnicas de Procesamiento de Lenguaje Natural, análisis probabilísticos, reconocimiento de patrones, modelos estadísticos, entre otros.

Por otra parte, investigaciones como [83–122] utilizan técnicas de inteligencia artificial, de forma más específica, distintos tipos o variaciones de Redes Neuronales como: *Multi-Column Convolutional Neural Networks, Attention-based LSTM, Question Dependent Recurrent Entity Network, BiLSTM, Convolutional Neural Networks, Dynamic Memory Networks, Recurrent Neural Networks HR-BiLSTM, Deep Convolutional Neural Networks, MatchLSTM, Tensor Product Recurrent Network, Dynamic Coattention Networks, Attentive Recurrent Neural Network, Bidirectional Attention Flow, Graph Neural Networks, Graph Convolutional Networks*, entre otras, para dar solución a la tarea en cuestión.

3.1. Procesamiento de Lenguaje Natural

El primer sistema pregunta-respuesta fue publicado en 1961 por Green *et al.* [22] enfocado en el contexto de datos de juegos de baseball como: mes, día, lugar, equipos y marcadores por año. Por lo tanto, este contexto es limitado, es suficiente la implementación de un vocabulario reducido y el tema es familiar. Esta investigación se compone de dos partes: la parte lingüística y el procesador; la primera de estas, se encarga de leer la pregunta desde una tarjeta perforada, se analiza sintácticamente, y posteriormente determina que información se está proporcionando acerca de los datos que se están solicitando. El procesador busca atreves de los datos la información apropiada, se procesan los resultados de la búsqueda y finalmente, se imprime la respuesta obtenida. Este sistema está escrito en IPL-V, un lenguaje de procesamiento de información, el cual, utiliza listas y listas de jerarquías, llamadas listas de estructuras para representación de información. Tanto los datos como diccionario son listas de estructuras, en las cuales, los elementos de información se expresan como pares atributo-valor, por ejemplo, equipo = Red Sox.

En 1994 se propone MASQUE/SQL [23], una interfaz portable de lenguaje natural para bases de datos Prolog. Este sistema se encarga de responder preguntas en inglés, generando a partir de éstas la respectiva consulta en Prolog, la cual, se evalúa a partir de una base

de datos del mismo lenguaje de programación. MASQUE se ejecuta mediante el DBMS (Database Management System) relacional. MASQUE/SQL asume que el mundo al que se refieren las pregunta en inglés consta de entidades y ciertos tipos de relaciones entre éstas. Los posibles tipos de entidades del mundo son organizados en arboles de dependencias, donde cada ancestro asume sus tipos descendientes.

FALCON [24] es un mecanismo de respuesta que integra diferentes formas de conocimiento sintáctico, semántico y pragmático con el objetivo de lograr un mejor rendimiento. Para encontrar una respuesta a partir de una pregunta de dominio abierto expresada en lenguaje natural en una amplia colección de textos, es necesario identificar de forma eficiente todos aquellos pasajes de texto en los cuales pueda encontrarse. La búsqueda de las respuestas se basa en el hecho de que una eventual respuesta será encontrada en un párrafo que contenga los conceptos de la pregunta más representativos e incluya un concepto textual de la misma categoría de la respuesta esperada. Este trabajo se divide en una recuperación booleana y un mecanismo de filtrado.

AnswerBus [25], un sistema pregunta-respuesta de dominio abierto basado en recuperación de información a nivel de oraciones. Este sistema acepta preguntas expresadas en lenguaje natural en idioma inglés nativamente y mediante un módulo de traducción permite realizar preguntas en alemán, francés, español, italiano y portugués. AnswerBus, toma la pregunta del usuario expresada en lenguaje natural para posteriormente someterla a un módulo de detección de idioma, el cual, al no identificar el idioma inglés éste envía la cadena de texto original con el idioma origen a la herramienta Altavista BabelFish, que genera como salida la pregunta traducida al inglés.

Brill *et al.* [26] presentan AskMSR, un sistema pregunta-respuesta con una arquitectura dividida en cuatro etapas principales: reformulación de consulta, minería de n-gramas, filtrado y tiling de n-gramas.

Más a detalle, la etapa de reformulación de consulta consistió en, dada una pregunta el sistema genera un conjunto de cadenas de texto reescritas, la cuales son descritas como sub-cadenas de texto de respuestas declarativas a la pregunta. Un ejemplo propuesto por los autores es el siguiente: *When was the paper clip invented?* que puede ser reescrito como: *The paper clip was invented*. Tomando en cuenta la posibilidad de que estas cadenas de texto nuevas no obtengan resultados al realizar la búsqueda, se producen variaciones menos precisas que brindan la oportunidad de encontrar coincidencias.

Parthasarathy *et al.* [28] utilizaron una técnica de mapeo de plantillas para la detección del tipo de pregunta, con base en la consulta apropiada para un motor de búsqueda específico, esto con el objetivo de dar respuesta a preguntas factoide. Posteriormente, bloques de contenido fueron extraídos de páginas relevantes, siendo éstos considerados como respuestas. Tales bloques son detectados utilizando una técnica de proyección y se agrupan según su similitud. Se selecciona el cluster de respuestas utilizando la propiedad de redundancia de las respuestas web. Por último, los bloques de contenido son colocados en una misma página para realizar una comparación.

Unger *et al.* [29] describen un sistema pregunta-respuesta basado en ontologías, Pythia, el cual utiliza representaciones lingüísticas de principios con la finalidad de construir de forma compositiva representaciones de significado general, que pueden ser subsecuentemente traducidos a consultas formales. Esto permite al sistema construir consultas formales incluso para preguntas complejas de lenguaje natural, es decir, aquellas que involucran cuantificación y superlativos. Por otra parte, se basa en una especificación de una interfaz léxico-ontológica que explica la posible realización lingüística de los conceptos de la ontología. Esto permite

construir representaciones de significado que utilizan un vocabulario alineado al vocabulario de una ontología dada, garantizando un mapeo preciso y correcto de términos de lenguaje natural a los términos correspondientes en la ontología.

Dong *et al.* [30] presentan LogAnswer, un sistema pregunta-respuesta para el idioma alemán que utiliza como base de conocimiento la información disponible en línea. Este sistema recibe como entrada una pregunta expresada en lenguaje natural y posteriormente se presenta al usuario una lista de respuestas resaltadas en los fragmentos de textos relevantes para proporcionar al usuario un contexto acerca de la misma.

Las respuestas se derivan de una amplia base de conocimiento, la cual, fue generada mediante la traducción de la Wikipedia alemana a una representación de red semántica en el formalismo MultiNet (Multilayered Extended Semantic Networks). 29.1 millones de oraciones fueron analizadas en conjunto con 12,000 reglas lógicas que conforman el conocimiento general de fondo.

Shizhu *et al.* [31] proponen un algoritmo basado en un framework de aprendizaje, red lógica de Markov (MLN, por sus siglas en inglés, *Markov Logic Network*), para aprender un modelo conjunto con la finalidad de construir consultas estructuradas a partir de enunciados del lenguaje natural.

Kim et al. [33] consiste en una arquitectura para sistemas pregunta-respuesta capaz de utilizar las oraciones a lo largo de un documento como fuente para la pregunta y/o respuesta. Este sistema proporciona al usuario un conjunto de preguntas candidatas de acuerdo con la información que el usuario necesite, y las preguntas candidatas son generadas automáticamente a partir de oraciones significativas que se espera que contengan hechos o eventos relevantes. Además, se construyó una base de datos de pares (preguntas - respuestas) después de analizar una colección completa de documentos. Para esto, se realizaron los siguientes procedimientos: división de oraciones, reconocimiento de entidades nombradas, generación de preguntas, filtrado de preguntas, indexación de preguntas y respuestas.

Zhang et al. [93] proponen un enfoque para el análisis de la pregunta y extracción de respuestas para el idioma chino. Este sistema utiliza el análisis general de preguntas, el cual, involucra la extracción de palabras clave y la clasificación de preguntas; Éste último implementa *Support Vector Machines* (SVM) utilizando cuatro tipos de características: palabras, *part of speech*, entidades nombradas y semántica.

ACQA_onto, propuesto por Hu et al. [34] se enfoca en el uso de ontologías restringidas, el modelo de espacio vectorial y técnicas de procesamiento de lenguaje natural para obtener la respuesta de preguntas en el dominio de cultivo agrícola. Este sistema se compone de cuatro módulos: pre-procesamiento de la pregunta, análisis del tipo de problema, extracción de respuestas y clasificación de respuestas. Además, se empleó un tesauro de agricultura, un léxico de sinónimos y plantillas para los tipos de pregunta.

Liu et al. [42] presenta un sistema pregunta-respuesta en idioma chino con las ventajas y propiedades de los sistemas tradicionales, sino también, capaz de utilizar la información disponible en línea. Éste se compone de tres módulos: análisis de la pregunta, recuperación de información y extracción de respuestas. Para llevar a cabo estas tareas se implementaron distintas tecnologías para el procesamiento de lenguaje natural como: clasificación de preguntas, análisis de dependencia sintáctica, detección de entidades nombradas, etc.

Xie et al. [49] presentan un sistema pregunta-respuesta basado en ontologías, este extrae palabras clave tras haber analizado la pregunta del usuario, la cual, es transformada en una consulta de elementos básicos en la ontología. Al final, el sistema proporciona conocimiento relacionado para ayudar a estudiantes en la adquisición de conocimiento sistemáticamente. La

ontología de este sistema se centra en temas de procesamiento de lenguaje natural únicamente.

Los trabajos descritos en [57] y [58] se enfocan en el uso de ontologías para resolver la tarea de pregunta-respuesta cada uno con sus respectivas distinciones, por ejemplo, Devi et al. [57] presentan DANS, un sistema pregunta-respuesta en el dominio de la agricultura. Este sistema utiliza una combinación de tecnologías de procesamiento de lenguaje natural y web semántica para formular una consulta SPARQL a partir de las cadenas de texto de entrada. Por otra parte, Jayalakshmi et al. [58] presentan un sistema pregunta-respuesta con un enfoque basado en la medida de similitud en la pregunta de entrada con el objetivo de encontrar el significado apropiado entre las palabras. Este sistema se compone de tres fases: determinación de la consulta, expansión de la misma y el método de vinculación de entidades.

Garg et al. [60] desarrollaron un sistema pregunta-respuesta para el lenguaje de programación Java, en el cual, la base de conocimiento utilizó una ontología y se basó en las relaciones derivadas del dominio para obtener la respuesta final. Se combinaron técnicas de procesamiento de lenguaje natural y web semántica.

3.2. Aprendizaje Automático, Aprendizaje Profundo

Juárez-González *et al.* [84] presentan un sistema pregunta-respuesta centrado en una arquitectura full data-driven que requiere conocimiento mínimo acerca del léxico y de la sintaxis de un lenguaje determinado. Este sistema implementó aprendizaje automático y técnicas de minería de texto para identificar las respuestas más probables a dos tipos de preguntas: factoides y de definición. Para brindar respuesta a las preguntas factoide se consideraron tres módulos: recuperación de pasajes, donde los pasajes con mayor probabilidad de contener la respuesta son recuperados de la colección de documentos; clasificación de la pregunta, que tiene como objetivo determinar el tipo de respuesta; y extracción de respuesta, donde las respuestas candidatas son seleccionadas utilizando un enfoque de aprendizaje automático, generando como salida la respuesta al usuario.

Ferrucci *et al.* [85] presentan Watson, un sistema DeepQA que nace del reto de competir en el nivel del campeón humano en el programa americano de televisión, Jeopardy. Watson logró desempeñarse en niveles de expertos humanos en términos de precisión, confianza y velocidad de respuesta en la competición en cuestión. Este sistema es una arquitectura basada en evidencia probabilística paralela. Se implementaron más de 100 técnicas diferentes para análisis de lenguaje natural, identificación de fuentes, búsqueda y generación de hipótesis, búsqueda y puntuación de evidencia, y unión y puntuación de hipótesis. La arquitectura de una aplicación DeepQA inicia con la etapa de adquisición de contenido, es decir, la identificación y recolección de la información que será utilizada como fuente de respuesta y evidencia. Inicialmente, se realiza un análisis de las preguntas de ejemplo del espacio del problema para producir una descripción de los tipos de preguntas que deben responderse y una caracterización del dominio de la aplicación. Dado el tipo de preguntas y el amplio domino del desafío Jeopardy, las fuentes de Watson incluyen una amplia gama de enciclopedias, diccionarios, tesauros, artículos de noticias, obras literarias, entre otros. Además del contenido para las fuentes de respuestas y evidencia, DeepQA aprovecha otros tipos de contenido estructurado y semiestructurado. Otro paso en el proceso de adquisición de contenido fue identificar y recopilar estos recursos, que incluyen bases de datos, taxonomías y ontologías, como dbPedia, WordNet y Yago ontology.

Cao *et al.* [86] presentan AskHERMES, un sistema pregunta-respuesta enfocado en preguntas clínicas implementando análisis semántico profundo, obteniendo como salida resúmenes

extractivos como respuestas. Este sistema inicia con un proceso de análisis de pregunta, en el cual, las preguntas son clasificadas en 12 temas generales para facilitar la recuperación de información, éstos incluyen: dispositivo, diagnóstico, epidemiología, etiología, historia, manejo, farmacología, hallazgo físico, procedimiento, pronóstico, prueba y tratamiento, y prevención. Además, estos temas fueron usados para anotar 4654 preguntas clínicas. Algunas de las técnicas implementadas en la etapa de análisis de la pregunta son: enfoques de aprendizaje automático supervisado y support vector machines (SVM) para realizar la clasificación de la pregunta. Para la identificación de palabras clave, se formuló como un problema de etiquetado de secuencias utilizando el modelo de campos aleatorios condicionales (Conditional Random Fields, CRF). Adicionalmente, se utilizaron características léxicas básicas, características sintácticas y la incorporación de la herramienta léxica MMTx, una implementación de MetaMap, para asignar texto a los conceptos UML y tipos semánticos como características de aprendizaje para ambas tareas. Adicionalmente, para que las redes semánticas fueran accesibles por gestores de teoremas modernos, la base de conocimiento MultiNet fue traducida en lógica de primer orden. Tomando en cuenta que la base de conocimiento es demasiado grande para ser manejada por una demostración de teoremas, es necesario realizar una serie de procesos que involucran a la pregunta para identificar todos los fragmentos relevantes en la base de conocimiento. Posteriormente, la pregunta es analizada utilizando técnicas de procesamiento de lenguaje natural y traducida a la representación correspondiente para MultiNet y en lógicas de primer orden.

Madotto et al. [87] proponen un modelo basado en una Recurrent Entity Network (REN) y a su vez una variante de la misma, llamada *Question Dependent Recurrent Entity Network* (QDREN). Este modelo consta de tres componentes: codificador de entrada, memoria dinámica y módulo de salida. El codificador de entrada convierte el conjunto de palabras de la oración y de la pregunta a sola representación vectorial mediante el uso de una máscara multiplicativa. Durante este proceso, una matriz de inclusión es utilizada para convertir las palabras a vectores. Por otra parte, el componente de memoria dinámica tiene como objetivo almacenar información de las entidades presentes en el conjunto de oraciones. Cabe mencionar que este módulo es similar a una *Gated Recurrent Unit* (GRU) con un estado oculto dividido en bloques, los cuales, representan una entidad y almacenan hechos relevantes de los mismos. Enseguida, el módulo de salida crea una distribución de probabilidad sobre los estados ocultos de las memorias usando la pregunta dada. Por lo tanto, los estados ocultos se suman, usando la probabilidad como peso, para obtener un estado único que representa toda la entrada. Finalmente, la salida de la red se obtiene combinando el estado final con la pregunta. Es necesario mencionar que el modelo fue entrenado utilizando pérdida de entropía cruzada en conjunto con el término de regularización L2. La Figura 4 muestra el esquema conceptual del modelo propuesto. Este modelo fue evaluado con dos datasets: bAbI 1k y CNN News Articles, obteniendo una precisión de 62.8 %.

Wang et al. [88] proponen un modelo generativo de comprensión de máquina que aprende de forma conjunta a preguntar y responder preguntas basadas en documentos. El modelo propuesto utiliza un framework secuencia a secuencia que codifica el documento y genera una pregunta dada una respuesta y viceversa en conjunto con un mecanismo de atención y un decodificador pointer-softmax. El modelo toma un documento de entrada, es decir, una secuencia de palabras y una secuencia de condición para obtener como salida una secuencia de destino. La condición corresponde a la secuencia de palabras de la pregunta en el modo de generación de respuestas (a-gen) y la secuencia de palabras de la respuesta en el modo de generación de preguntas (q-gen). Posteriormente, en el codificador, una palabra en una

secuencia de entrada es incrustada en un vector utilizando una capa de incrustación. La información a nivel de carácter es capturada con los estados finales de un modelo LSTM en las secuencias de caracteres. La representación final para un token de palabra concatena las incrustaciones de nivel de palabra y carácter. Éstos son subsecuentemente codificados con otro BiLSTM en vectores de anotación para el documento y la secuencia respectivamente. En el módulo decodificador se utiliza un decodificador basado en redes neuronales recurrentes que implementa un mecanismo pointer-softmax. En cada paso de generación el decodificador decide adaptativamente si genera a partir de un vocabulario de decodificador o señalar una palabra en la secuencia de origen. La recurrencia del decodificador apuntador es implementada con dos celdas LSTM, donde cada una de éstas son los estados recurrentes. Finalmente, este modelo es evaluado con el corpus SQuAD, que es un machine comprension dataset que consta de más de 100 mil pares de preguntas y respuestas de fuentes múltiples en 536 artículos de Wikipedia, obteniendo los siguientes resultados: para la generación de respuestas se obtuvo un F1-Score de 54.5 % y un Exact Match de 41 %.

Yingqi et al. [89] proponen una red neuronal recurrente atenta con una matriz de similitud basada en un modelo de red neuronal convolucional (AR-SMCNN), el cual, es capaz de capturar información jerárquica completa utilizando las ventajas de las RNN y de las CNN. Las RNN son utilizadas para capturar las correlaciones de nivel semántico por su naturaleza de modelado secuencial, y se utiliza un mecanismo de atención para realizar un seguimiento de las entidades y relaciones simultáneamente. Por otra parte, las CNN basadas en la matriz de similitud son utilizadas en conjunto con agrupación en dos direcciones para extraer la coincidencia de interacción de palabras de nivel literal explotando la fuerza de CNN de la correlación espacial de modelado entre los datos. Además, se utiliza un nuevo método de extensión heurística para la detección de entidades, el cual, disminuye el efecto de ruido en el procesamiento de la pregunta. Este sistema se basa en Freebase, una base de conocimiento donde las tripletas se forman con la estructura subject-relation-object. Básicamente, el sistema realiza dos procesos fundamentales: (1) extracción del tema de la pregunta, que corresponde al subject en las tripletas, y (2) predicción de relaciones que mejor describan el subject y la respuesta. Para la detección de entidades, dada una pregunta, se entrena una red bidireccional LSTM como una tarea de etiqueta secuencial, que puede considerarse como un problema de clasificación binaria que predice si cada palabra en una oración pertenece a la mención de entidad.

Caiming Xiong et al. [90] presentan una Dynamic Coattention Network (DCN), una red neuronal de extremo a extremo para responder preguntas. El modelo consiste en un módulo de codificación de documentos y preguntas, un módulo coattention encoder y un módulo dynamic pointing decoder. El módulo de codificación de documentos y preguntas crea una secuencia de vectores de palabras correspondientes a la pregunta, mismo proceso se aplica para las palabras en un documento. Posteriormente, se utiliza LSTM para crear una matriz de codificación del mismo. Además, se añade un vector centinela. Las incrustaciones de preguntas se calculan con el mismo LSTM para compartir el poder de representación. Adicionalmente, se define una representación intermedia de la pregunta. Para permitir la variación entre el espacio de codificación de preguntas y el espacio de codificación de documentos, se utilizó una capa de proyección no lineal sobre la codificación de pregunta. El módulo coattention encoder se implementa un mecanismo coattention que procesa la pregunta y el documento de forma simultánea, para finalmente fusionar ambos contextos de atención. Subsecuentemente, se calcula una matriz de afinidad, la cual, contiene puntajes de afinidad correspondientes a todos los pares de palabras del documento y de la pregunta

Dong *et al.* [83] presenta un sistema pregunta-respuesta que implementa *Multi-Column Convolutional Neural Networks* (MCCNNs) para analizar preguntas de múltiples aspectos. Las MCCNNs utilizan diferentes redes de columnas para extraer el tipo de respuesta, relaciones, e información del contexto de las preguntas de entrada. Las entidades y relaciones en la base de conocimiento son representadas como lower-dimensional vectors. Entonces, una capa de puntuación es empleada para clasificar las respuestas candidatas de acuerdo a la representación de las preguntas y las respuestas candidatas. Adicionalmente, utilizando Freebase se generó una paráfrasis de la pregunta, la cual, se utilizó para entrenar redes y generalizar para todas aquellas palabras que no hayan sido vistas o utilizadas inicialmente en una manera de aprendizaje de múltiples tareas. A continuación, la Figura 3 muestra del lado izquierdo la arquitectura de red para el entendimiento de la pregunta y del lado derecho las respuestas candidatas a la pregunta: *when did Avatar release in UK?*. Este sistema fue evaluado utilizando el dataset WebQuestions con el cual se obtuvo un 40.8 % en F1-Score y 45.1 % para P@1 Score.

Htut et al. [91] proponen dos clasificadores basados en redes neuronales que asignan diferentes valores a diferentes pasajes basándose en la probabilidad de que contengan las respuestas a una pregunta dada. Como base de conocimiento se utilizan todos los documentos disponibles en línea. La arquitectura propuesta se compone por un mecanismo de búsqueda, ranker y el lector. Los dos clasificadores desarrollados en este trabajo son: InferSent ranker el cual evalúa el rendimiento de la similitud semántica en ranking para el QA, y Relation-Networks ranker que es usado para evaluar el rendimiento de coincidencia de relevancia en el ranking para QA. Además, el lector de documentos DrQA fue utilizado en esta arquitectura. Dada una pregunta y un párrafo, el modelo de rangos actúa como una función de puntuación que calcula la similitud entre ellos. Se utilizaron dos modelos de redes neuronales como funciones de puntuación, el primero de éstos, una red neuronal feed-forward que utiliza InferSent, el cual proporciona representaciones distribuidas para oraciones. Éste es entrenado con Stanford Natural Language Inference Dataset y MultiGenre NLI Corpus empleando aprendizaje supervisado. Estas representaciones son utilizadas para implementar un clasificador que ordena los elementos de acuerdo a la similitud semántica. Esta red neuronal es implementada como función de puntuación, la cual consiste en una capa lineal seguida de una función de activación ReLu en conjunto con otra capa lineal de valor escalar que proporciona la puntuación de similitud entre una pregunta y un párrafo. El segundo modelo utiliza Relation-Networks como el modelo de clasificación que se enfoca en medir la relevancia entre las palabras de la pregunta y aquellas que se encuentran en el párrafo. Las Relation-Networks tiene como objetivo inferir la relación entre pares de objetos, en este caso corresponde a los pares de las palabras de la pregunta y el párrafo. Por otra parte, el lector implementa DrQA, el cual es una red neuronal recurrente multi-capa diseñada para extraer un lapso de respuesta a una pregunta de documento o conjunto de párrafos. Posteriormente, para la selección de los párrafos se utiliza el resultado de los clasificadores y se seleccionan los primeros cinco párrafos, de los cuales se calcula la probabilidad de mayor relevancia entre éstos. Finalmente, se utilizó el dataset QUASAR-T para evaluar el trabajo en cuestión, obteniendo los siguientes resultados: Exact Match obtuvo un total de 31.2 % y un F1-Score de 37.6 % utilizando el clasificador InferSent + DrQA. El modelo de rangos que se centra en la coincidencia de relevancia (Relation-Networks ranker) logra una recuperación de recuperación significativamente más alta, pero el modelo de rangos que se centra en la similitud semántica (InferSent ranker) tiene un mejor rendimiento general de QA.

Dodiya et al. [96] presentan un método de clasificación de preguntas para sistemas pregunta-

respuesta basado en reglas. El módulo de procesamiento de preguntas ayuda a asignar una la categoría adecuada a una pregunta e identifica las palabras clave en esta. Este sistema utilizó la taxonomía de dos capas propuesta Li y Roth [123]. Además, se analizó la estructura sintáctica de la pregunta para generar patrones sintácticos para cada categoría.

La clasificación de preguntas es el paso inicial para la recuperación de información relevante en este sistema. Esta toma como entrada la consulta del usuario y la clasifica. Además, se identifica las palabras clave, se eliminan los stop words y se realiza el stemming de cada palabra. La taxonomía utilizada [123] compuesta por seis categorías generales y 50 categorías específicas.

Chen et al. [98] presentan un modelo basado en una red neuronal atenta, se utilizó una Bi-LSTM para una mejor representación de las preguntas con el agrupamiento de atención. Además, se aplicó TransH a la arquitectura con el objetivo de evitar los inconvenientes de TransE para una mejor integración del conocimiento. De forma muy general, el sistema recibe una pregunta expresada en lenguaje natural y devuelve un conjunto de entidades como respuestas. Inicialmente, se identifica la entidad de tema y se generan respuestas candidatas. Posteriormente, se emplea una red neuronal atenta para representar la pregunta bajo la influencia de los aspectos de las respuestas candidatas. Finalmente, la puntuación de similitud entre la pregunta y cada respuesta candidata es calculada, y aquellas con mayor puntuación son seleccionadas como la respuesta final. Se utilizaron incrustaciones de palabras de word2vec, las cuales, fueron aprendidas utilizando el modelo skipgram al predecir palabras de contexto lineal que rodean las palabras de destino. Estos vectores de palabras fueron entrenados en aproximadamente 100 billones de palabras del corpus Google News utilizando el método bag-of-words. La entrada del modelo son N oraciones con longitudes variables. Cada una de estas está constituida por palabras representadas en vectores.

Para la representación de las preguntas se utilizó una Bi-LSTM, la cual consiste en dos LSTM que se ejecutan en paralelo: una en la secuencia de entrada y la otra al reverso de esta. En cada paso de tiempo, el estado oculto de la LSTM bidireccional es la concatenación de los estados ocultos hacia delante y hacia atrás. Esta configuración permite la actualización de los estados ocultos para capturar información tanto pasada como futura. En la sección de representación de respuestas con incrustación de conocimiento se adoptó el modelo TransH para representar la base de conocimiento, el cual, es una mejora de TransE, además se integraron las representaciones en el proceso de entrenamiento del sistema. Por otra parte, el mecanismo de atención, la atención de la respuesta a la pregunta puede medirse por la relación entre cada representación de la palabra y la incrustación del aspecto de la respuesta.

Minaee et al. [99] proponen un modelo basado en aprendizaje profundo para la generación de respuestas de forma automática. En primer lugar, las preguntas y respuestas se integran mediante modelos neurales probabilísticos. Luego se entrena una red neuronal de similitud profunda para encontrar la puntuación de similitud de un par de respuestas y preguntas. Posteriormente, para cada pregunta se determina la mejor respuesta con base en la puntuación de similitud. Primero se entrenó este modelo en una base de datos pública de preguntas y respuestas de gran escala, y luego se adjuntó para transferirlo a los datos de chat de atención al cliente.

En esta investigación, Ma et al. [100] proponen un modelo híbrido de selección de respuesta combinando una red neuronal convolucional (CNN) y los métodos de extracción abstracta. En el modelo, el resumen de respuestas se extrae del texto con múltiples características y se envía a la CNN junto con la pregunta para obtener una representación semántica concisa y eficiente. A diferencia de los modelos profundos anteriores, se elimina la información irrelevante y se

generan mejores representaciones para las preguntas y respuestas, lo cual es necesario para responder preguntas que no son de tipo factoide.

Hong et al. [92] proponen, comparan e implementan dos modelos Match-LSTM y Bidirection Attention Flow (BiDAF). Para los modelos propuestos, se utilizan los vectores preentrenados GloVe 100-dimensional, el optimizador Adam y la pérdida de entropía cruzada de softmax estándar entre las probabilidades pronosticadas de inicio y final y las posiciones verdaderas. Tomando en cuenta factores como: la distribución de preguntas y longitudes de contexto en los datos de entrenamiento, y considerando la compensación entre la pérdida de información y la eficiencia de esta, se limitó la pregunta y la longitud del contexto a 30 y 300 caracteres, respectivamente. Para el modelo de línea de base, se utilizó una forma simple de cálculo de atención: para cada par de codificaciones de palabra de contexto y pregunta, se calculó el producto de punto como el puntaje de atención. Seguido de esto, el vector de atención es concatenado para cada posición de contexto con las codificaciones de contexto para formar la representación contextual atendida. Posteriormente, esta representación es enviada a otra capa LSTM para predecir el puntaje de cada posición, punto de inicio y punto final. Para Match-LSTM, se utiliza exactamente el codificador utilizado en el documento original. En el decodificador, se implementa una red de puntero ligeramente modificada para decodificar la salida de la capa Match-LSTM. La red de punteros es similar a un decodificador normal con atención aplicada a los estados ocultos del codificador, excepto que su salida es el puntaje de atención. La función de puntuación de atención tanh-softmax en el documento original es reemplazada con similitud bilineal.

El modelo BiDAF, con base en la evidencia del entrenamiento empírico, se demostró que el costo adicional de entrenamiento de parámetros adicionales y ajuste de hiperparámetros supera el beneficio potencial.

Joty et al. [102] presentan una solución a las preguntas y respuestas en foros comunitarios utilizando tres tareas: encontrar preguntas existentes relacionadas, encontrar respuestas relevantes a una pregunta nueva y encontrar buenas respuestas en un hilo de preguntas y comentarios relacionados. Además, se utilizaron redes neuronales profundas (DNN del inglés Deep Neural Networks) para aprender incorporaciones significativas específicas de tareas, las cuales son incorporadas en un modelo CRF para la configuración de tareas múltiples, realizando un aprendizaje conjunto sobre una estructura de gráfico compleja.

De Cao et al. [105] presentan un enfoque para la comprensión de lectura de múltiples saltos que se basa en la propagación de información a través del paso de mensajes entre entidades distribuidas en múltiples documentos. Este método utiliza tripletas conformadas por tuplas con elementos tales como: consulta, conjunto de documentos de apoyo, conjunto de respuestas candidatas y la respuesta que correcta, la cual, pertenece a este último. En el grafo de entidades de forma fuera de línea se organiza el contenido de cada instancia de entrenamiento en un gráfico que conecta las menciones de las respuestas candidatas entre los documentos de respaldo. Posteriormente, dada una consulta se identifican las menciones en el conjunto de documentos de respaldo de las entidades en las respuestas candidatas, que finalmente generan un nodo por mención. Para identificar esto se utilizó una heurística simple de coincidencia exacta. Este modelo aborda el razonamiento de varios pasos mediante la transformación de las representaciones de los nodos con un algoritmo de paso de mensajes diferenciable que propaga información a través del gráfico de entidades. El algoritmo está parametrizado por una red neuronal convolucional gráfica, GCN relacionales. Por otra parte, se codificaron las palabras en los documentos de respaldo y en la consulta utilizando un modelo de representaciones de palabras contextualizadas denominado ELMo, el cual, combina caracteres en una CNN

de bajo nivel con representaciones de capas intermedias LSTM que dan como resultado representaciones de palabras profundamente contextualizadas. Adicionalmente, el modelo utiliza una versión cerrada de la regla de propagación RGCN original.

Yue et al. [106] proponen redes de memoria dinámica (DMN del inglés, Dynamic Memory Networks) mejoradas para realizar la tarea de pregunta-respuesta mediante el procesamiento de entradas para extraer simultáneamente características salientes globales y jerárquicas. Estas redes también fueron utilizadas para construir múltiples conjuntos de características en cada paso de razonamiento. El módulo de entrada se desempeña como lectores de oraciones para procesar hechos mediante el uso de diferentes métodos de codificación como: la memoria a largo plazo (LSTM), las unidades recurrentes sincronizadas (GRU) y la codificación de posición (PE). Este módulo se divide en dos partes: el PE que se utiliza para producir representaciones originales de los hechos; y, la segunda parte, una red neuronal recurrente cerrada bidireccional, que se utiliza para producir representaciones finales de los hechos. El módulo de memoria episódica es la parte central de una DMN, donde el módulo de entrada interactúa con el módulo de pregunta. La DMN utiliza una estructura de atención recurrente para lograr la extracción o el razonamiento progresivo de la información en el módulo en cuestión. Además, se utilizan dos mecanismos: el mecanismo de atención decide cómo extraer información de los hechos en cada salto, implementado mediante ponderaciones de atención; y, el mecanismo de memoria se utiliza para generar una nueva memoria episódica basada en la anterior, el vector contextual actual y la representación de una pregunta. El módulo de respuesta recibe la salida del módulo anterior para inferir una respuesta mediante softmax. Luego, el modelo se entrenó minimizando el error de crossentropy. Además, sin importar que una respuesta se componga de una o varias palabras, se considera como un todo y no se utilizó otra RNN para producir la respuesta.

Phong-Khac et al. [110] presentan un estudio sobre el empleo de Ranking SVM y CNN para dos tareas: recuperación de información legal y respuesta a preguntas en el Concurso de extracción / incorporación de información legal.

Este sistema pregunta-respuesta se centra en el uso de CNN con características adicionales, esta recibe las características de entrada construidas, es decir, los vectores de incrustaciones de palabras de la pregunta y de las oraciones más relevantes en los artículos relacionados. Una oración representada por un conjunto de palabras se convierte en estos vectores utilizando el modelo de bolsa de palabras (BOW del inglés, Bag Of Words). Este genera una representación vectorial para una oración al hacer un resumen sobre la inclusión de palabras en la oración y el vector se normaliza luego por la longitud de la oración. Adicionalmente, se formó un modelo de incrustación de palabras mediante el uso de Word2Vec en los datos del cuerpo legal japonés. Este corpus contiene todos los artículos de derecho civil de la constitución de Japón con 13.5 millones de palabras de 642 artículos limpiados y tokenizados. Cada mapa de características se alimentó a una capa de agrupación para generar características potenciales mediante el uso del mecanismo promedio. Estas características se concatenaron en un solo vector para su clasificación mediante el Multi-Layer Perceptron con activación sigmoide. Durante el proceso de entrenamiento, se aprenden los parámetros de filtros y perceptrones para optimizar la función objetivo. En este modelo, se aplicaron 10 filtros de convolución a dos nodos de entrada adyacentes dado que estos nodos son del mismo tipo de entidad. Luego, se utilizó una capa de agrupamiento promedio para sintetizar características importantes. Para mejorar el rendimiento de CNN se utilizaron dos características estadísticas adicionales: TF-IDF y LSI, las cuales, se concatenaron con el resultado de la capa de agrupación, y posteriormente se introdujeron en un modelo Perceptron de 2 capas para predecir la respuesta.

Ma et al. [114] presentan una red de memoria a largo plazo (LTMN), que incorpora un módulo de memoria externa y un módulo de memoria a corto plazo largo (LSTM) para comprender los datos de entrada y generar respuestas multi-palabras. Este modelo LTMN puede entrenarse de extremo a extremo utilizando back-propagation y requiere supervisión mínima. La LTMN se divide en cuatro módulos: entrada, pregunta, memoria y respuesta. El módulo de entrada codifica datos de texto sin procesar (oraciones) en representaciones vectoriales. Del mismo modo, el módulo de preguntas codifica las preguntas las respectivas representaciones. Los módulos de entrada y pregunta pueden utilizar los mismos o diferentes métodos de codificación. Para llevar a cabo esto, se utilizó una técnica de aprendizaje de representaciones distribuida denominada, modelo de vectores de párrafo (paragraph2vec). Posteriormente, dadas las representaciones de las oraciones de entrada, el módulo de memoria calcula las probabilidades coincidentes entre la representación de la pregunta y de la oración, obteniendo como salida un vector. Inmediatamente, se genera la suma ponderada de las representaciones de la oración y las probabilidades coincidentes. Usando este vector de suma ponderada y la representación de la pregunta, el módulo de respuesta utiliza LSTM para generar la oración correspondiente, esta recibe la incrustación del token de inicio de respuesta y su estado correspondiente. Además, se utiliza la entrada del módulo de memoria, la representación de la pregunta, la matriz de pesos y el bias para generar la incrustación respectiva. Este modelo se entrenó de extremo a extremo con la pérdida definida por la entropía cruzada entre la respuesta verdadera y la salida pronosticada en peso, representada mediante la codificación instantánea. La respuesta pronosticada se genera al concatenar todas las palabras generadas por el modelo.

3.3. Discusión

Dichas técnicas han sido implementadas en distintos dominios como: baseball, restaurantes, trabajos y geografía, clínico, religión, pedagógico, *e-learning*, agricultura, Corán, debates políticos, biología, medicina tradicional china, horarios de eventos relacionados a los servicios básicos del hogar en Vietnam, médico, turismo en Rajasthan, lenguaje de programación Java, viajes, consejos turísticos, datos de atención al cliente, biomédico, patológico, hechos en una línea de tiempo, legal, datos tabulares, salud, influenza, Linux, científico, comunidades de preguntas y respuestas (*stack overflow*), problemas aritméticos y aquellos de dominio abierto.

Del mismo modo, existen trabajos basados en el tipo de la pregunta sin importar el dominio, es decir, preguntas factoide, definición, opinión, fill in the blank y de relación simple. Además, aquellas investigaciones enfocadas en la información disponible según la base de conocimiento como: Wikipedia, según la ontología, Freebase, Linked Data, preguntas del concurso Jeopardy!, TRECQA y en tablas.

La Figura 3.1 representa visualmente la distribución de los sistemas de preguntas y respuestas publicados a lo largo del tiempo, mostrando los avances en inteligencia artificial y procesamiento de lenguaje natural que han revolucionado el campo e impulsado el desarrollo de sistemas más sofisticados y precisos.

Por otra parte, la Figura 3.2 muestra la distribución de cada conjunto de datos utilizado en cada uno de los trabajo citados en esta investigación.

Es necesario mencionar que la mayoría de estos trabajos se enfocan en el idioma inglés. Sin embargo, algunas investigaciones fueron desarrolladas para los idiomas chino [34, 37, 41, 42, 45, 93, 120, 121], tibetano [43], persa [94], árabe [47, 48, 77], vietnamita [46], malayalam

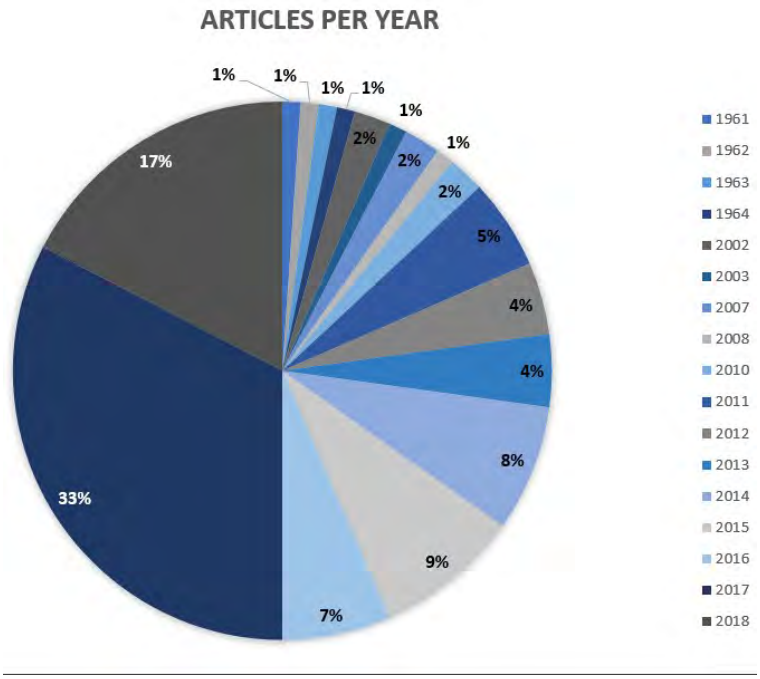


Figura 3.1: QA en el tiempo

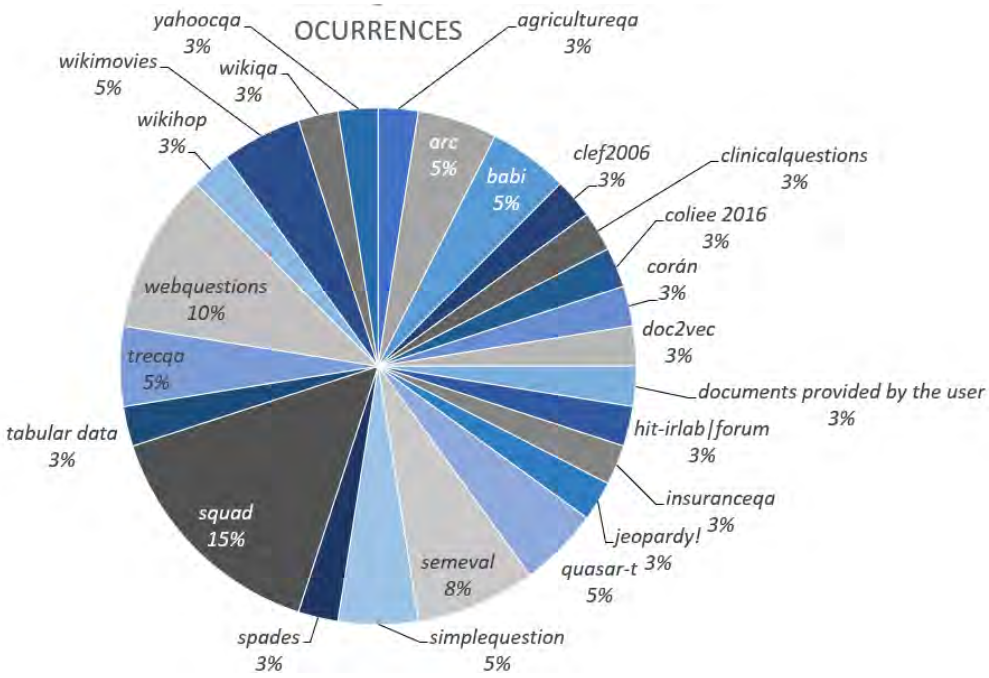


Figura 3.2: Datasets utilizados en la tarea de QA

[75], rayastani [59], turco [64] e indonesio [36, 79], los cuales, utilizan únicamente técnicas convencionales de Procesamiento de Lenguaje Natural, lo cual, implica que los modelos desarrollados inicialmente no pueden ser adaptados a otros idiomas, es decir, que estos modelos pueden haberse diseñado hacia un idioma específico o con un lenguaje controlado.

Sin embargo, a pesar de que estas investigaciones se enfocan en la misma tarea, utilizaron distintos datasets como: TREC en distintas ediciones, WebQuestions, CLEF, cor-

pus diseñados de acuerdo al dominio, QALD, COPARQ, Wikipedia, FREE917, bAbI 1k, CNN News Articles, InsuranceQA, AgricultureQA, BioASQ, Freebase, ClueWeb, SimpleQuestions, WebQSP, WikiMovies-FL, WikiMovies-WE, Quasar-T, SQUAD_OPEN, CuratedTREC, YahooCQA, WikiQA, SemEvalCQA, FB2M, FB5M y TRIVIA-QA. Así como distintas métricas de evaluación entre las que destacan: *precision*, *recall*, *accuracy*, *f-measure*, *mean reciprocal rate*, entre otras tal y como se muestra en la Figura 3.3 en cada uno de los trabajos citados.

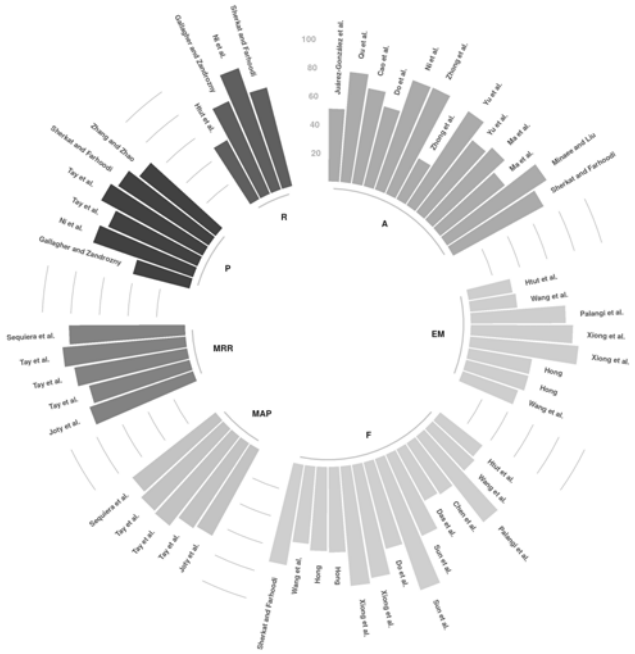


Figura 3.3: Investigaciones y métricas de evaluación utilizadas

De forma general, la tarea de pregunta-respuesta se enfrenta a distintos problemas como: la ambigüedad de la consulta, la complejidad del idioma, la disponibilidad y recuperación de la información, problemas relacionados a aspectos lingüísticos, situacionales, contextuales o visuales, polisemia, hiperonimia, la generación de modelos con fácil adaptabilidad a distintos temas y a diferentes idiomas, los cuales, son características que vuelven al lenguaje humano ambiguo a diferencia de los lenguajes de programación u otros lenguajes formales. Sin embargo, algunos autores consideran que el potencial de esta área de investigación es tal, que estos pueden llegar a ser los modelos que revolucionen a los motores de búsqueda actuales a pesar de lo avanzado que son hoy en día.

Capítulo 4

Método de solución

En este capítulo se describen los módulos que componen el método de solución. En la Figura 4.1 se muestra la arquitectura general del modelo. Los detalles de cada uno de estos módulos son descritos a detalle en las secciones correspondientes. En esta se visualiza el flujo al cual es sometida una pregunta. En la parte superior se muestra el *Wikipedia Corpus Generation*, el cual se ejecuta una única vez previo a la ejecución del modelo. Aquí se crea un corpus de Wikipedia en dos formatos: json y xml, así como una base de datos para posteriormente generar un archivo de puntuaciones TF-IDF de todos los documentos de Wikipedia entre sí.

La primera etapa, denominada *Question Reformulation* (Sección 4.3) se encarga de identificar si la pregunta del usuario es simple o múltiple; y en caso de detectar una pregunta múltiple, ésta es sometida al proceso de reformulación. Este módulo a su vez utiliza tareas de pre-procesamiento (Sección 4.1). La salida de este módulo es la entrada de los módulos *Document Retrieval & Answer Generation* (4.5) y *Question Classification* (Sección 4.4). Ambos módulos utilizan el Word Embedding de *Spanish Billion Word Corpus* para la conversión de texto a valores numéricos (Sección 4.2.3). Finalmente, las salidas de los módulos mencionados son evaluados de forma idependiente: reformulación de preguntas, clasificación de preguntas y recuperación y generación de respuestas.

El resto de este capítulo se estructura de la siguinte forma: en la Sección 4.6 se describe el software utilizado para la interfaz web, así como las instrucciones y algunas capturas de pantalla para mostrar su funcionamiento. En la Sección 4.1 muestran algunas de las herramientas de PLN más utilizadas en la literatura y la selección de una herramienta para PLN en español. La Sección 4.2.1 describe brevemente los datasets para QA más utilizados en la literatura. En la Sección 4.2.2 se describe la herramienta utilizada para la generación del corpus y la información sobre la fuente de información. Por otra parte, en la Sección 4.2.3 se detalla la creación del *Word Embedding* para el idioma español. Finalmente, en la Secciones 4.3, 4.4 y 4.5 se detalla el módulo de reformulación, la Red Neuronal Convolucional y la Red Neuronal Recurrente, respectivamente.

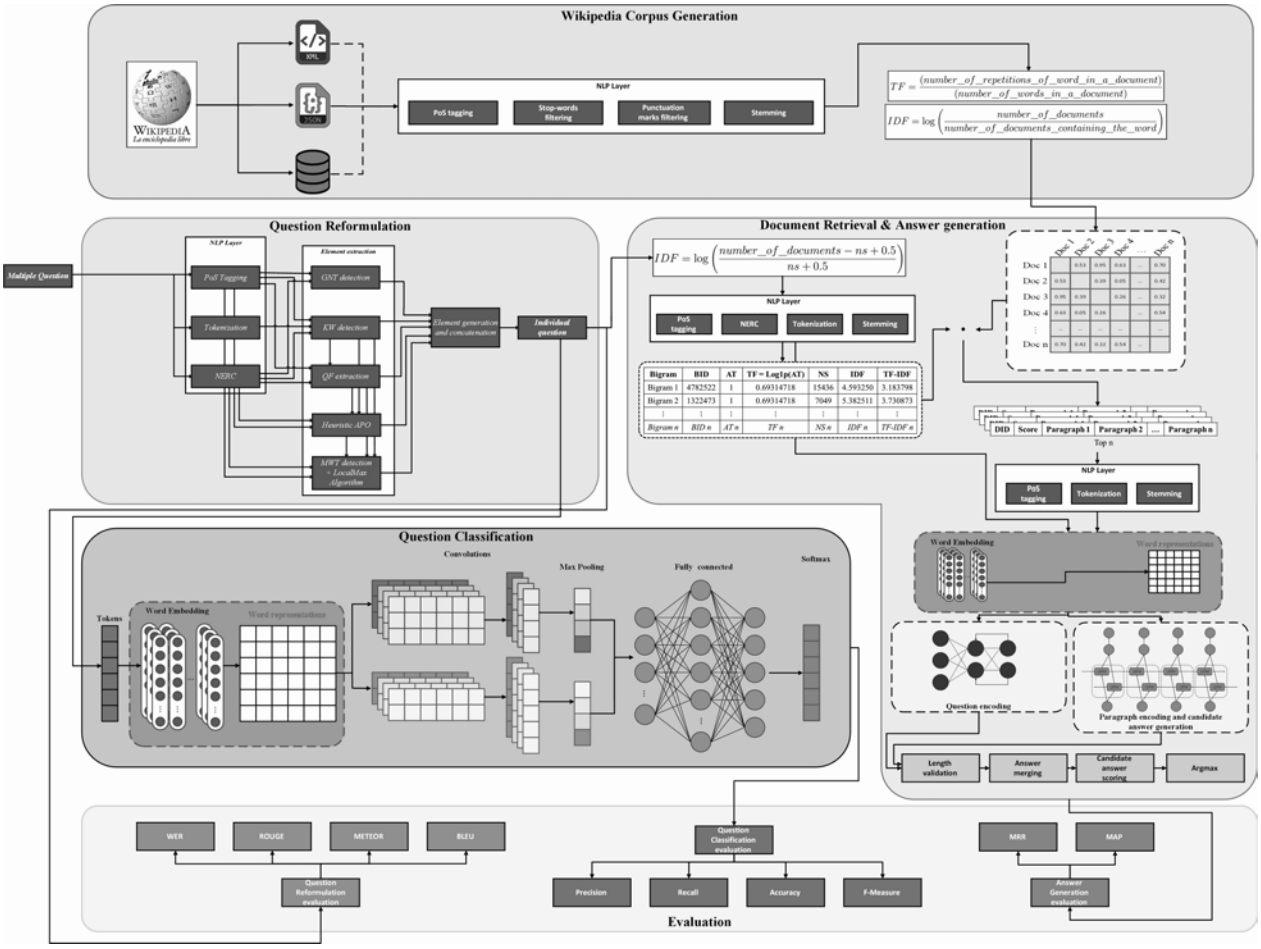


Figura 4.1: Arquitectura general del modelo

4.1. Análisis y selección de herramientas para el módulo de PLN

En esta actividad se realizó el análisis de cuatro herramientas para el Procesamiento de Lenguaje Natural para el idioma español. La Tabla 4.1 realiza la comparación entre las herramientas más comunmente utilizadas en la literatura: NLKT, Scikit Learn, Stanford Core NLP, Freeling y Spacy.

Si bien existen más herramientas para el procesamiento de lenguaje natural, las mencionadas anteriormente fueron seleccionadas por una caracteristica adicional, que es la integración de éstas con el lenguaje de programación Python.

Cada una de las herramientas mencionadas han mostrado un gran desempeño en el PLN en distintos idiomas. Sin embargo, para esta investigación se seleccionó la herramienta **Freeling**. Esta herramienta es desarrollada y mantenida en el Centro de Investigación TALP, en la Universitat Politècnica de Catalunya. El TALP forma parte del RTTH, una red temática en tecnologías del habla. Actualmente compuesta por 17 grupos de investigación españoles. Sus objetivos son fomentar la investigación en Tecnologías del Habla en distintos idiomas principalmente el español y el catalán.

Tabla 4.1: Herramientas para PLN

Herramienta	Descripción	Idiomas
NLTK [124]	Considerada por algunos autores como la herramienta de PLN más utilizada. Variedad de extensiones para terceros.	Árabe, danés, holandés, inglés, finlandés, francés, alemán, húngaro, italiano, noruego, portugués, rumano, ruso, **español** y sueco.
Scikit Learn[125]	Cuenta con funciones que ayudan a utilizar tokenización y bolsas de palabras para crear características en problemas de clasificación de textos.	Árabe, alemán, inglés, **español**, francés, italiano, japonés, ruso, holandés, polaco y portugués.
Stanford Core NLP[126]	Creada y mantenida por Stanford. Esta biblioteca está optimizada para la velocidad y tiene funciones como etiquetado PoS, análisis de aprendizaje de patrones, reconocimiento de entidades nombradas, entre otros.	Árabe, chino, inglés, francés, alemán y **español**.
Freeling[127]	Biblioteca en C ++ que proporciona funcionalidades de análisis de lenguaje (análisis morfológico, detección de entidades nombradas, etiquetado PoS, análisis, desambiguación del sentido de las palabras, etiquetado de roles semánticos, entre otros) para distintos idiomas.	Inglés, **español**, portugués, italiano, francés, alemán, ruso, catalán, gallego, croata, esloveno, entre otros
SpaCy[128]	Biblioteca gratuita de código abierto para el Procesamiento de Lenguaje Natural en Python capaz de realizar tokenización, Etiquetado PoS, Análisis de dependencias, Lematización, Detección de límites de oración, Reconocimiento de entidad nombrada, Vinculación de entidades, Semejanza, Clasificación de texto, Coincidencia basada en reglas, Entrenamiento de modelos, entre otras	chino, danés, holandés, inglés, francés, alemán, griego, italiano, japonés, lituano, noruego Bokmål, polaco, portugués, rumano y **español**

4.2. Generación de corpus de Wikipedia

En esta sección se describen los distintos módulos utilizados para la creación del corpus de Wikipedia.

4.2.1. Análisis y selección de Datasets para QA

En esta sección se describen algunos de los datasets para QA más destacados en la literatura. Si bien no son los únicos datasets utilizados para la tarea en cuestión, únicamente fueron considerados aquellos de dominio abierto con el objetivo de no sesgar el vocabulario e incluso la sintaxis de las preguntas.

WebQuestions [129] fue creado utilizando la API de sugerencias de Google. La API recibe tres entradas: la pregunta sin la frase antes de la entidad, la pregunta sin la entidad y la pregunta sin la frase después de la entidad. Cada una de estas consultas genera 5 preguntas candidatas. Debido a la naturaleza del enfoque, las preguntas generadas comienzan con una palabra clave y giran en torno a una sola entidad. Este dataset tiene un total de 5,810 preguntas con la estructura que se muestra en la Figura 4.2.


```
{"qId": "wqr001696", "answers": ["Federal republic"], "qText": "what is the politicals system of nigeria?"}
{"qId": "wqr001095", "answers": ["Italia Conti Academy of Theatre Arts"], "qText": "where did pixie lott go to school?"}
{"qId": "wqr003126", "answers": ["Mwinilunga"], "qText": "where does the zambezi river begin?"}
{"qId": "wqr001075", "answers": ["Dwayne Carter III"], "qText": "what is lil wayne real name?"}
```


Figura 4.2: Formato del dataset WebQuestions

SimpleQuestions [130] fue generado a partir de la recopilación de hechos en Freebase, utilizandolos como tripletas *sujeto-predicado-objeto*. Este dataset consta de 108,442 preguntas como las que se muestran en la Figura 4.3.


```
www.freebase.com/m/0fll_lg  www.freebase.com/music/release/track    www.freebase.com/m/0dwkrfl  What's a song off of knuffelrock 6
www.freebase.com/m/09nzdd   www.freebase.com/film/director/film www.freebase.com/m/0tm5mj7  which film was directed by joseph morgan (actor)
www.freebase.com/m/02hlgmw  www.freebase.com/cvg/game_version/publisher www.freebase.com/m/020wds   Who published heart of the maelstrom
www.freebase.com/m/03nq3v   www.freebase.com/people/deceased_person/place_of_death  www.freebase.com/m/09jp3    where did juan antonio lavalleja's life end?
```


Figura 4.3: Formato del dataset SimpleQuestions

Para el dataset 30M Factoid Question-Answer Corpus [131] se utilizó una arquitectura de *encoder-decoder* para generar preguntas a partir de hechos de Freebase. Este dataset consta de 30 millones de preguntas como las que se muestran en 4.4


```
<http://rdf.freebase.com/ns/m.04whkz5>    www.freebase.com/book/written_work/subjects    <http://rdf.freebase.com/ns/m.01cj3p>    what is the book e about ?
<http://rdf.freebase.com/ns/m.04j0t75>    www.freebase.com/film/film/country    <http://rdf.freebase.com/ns/m.07ssc>    what country is the debt from ?
<http://rdf.freebase.com/ns/m.0ftqr>    www.freebase.com/music/producer/tracks_produced    <http://rdf.freebase.com/ns/m.0p600l>    what songs have nobuo uematsu produced ?
<http://rdf.freebase.com/ns/m.036p007>    www.freebase.com/music/release/producers    <http://rdf.freebase.com/ns/m.0677ng>    who produced eve-olution ?
```


Figura 4.4: Formato del dataset 30M Factoid Question-Answer Corpus

WikiMovies [132] es un dataset que incluye solo la parte correspondiente a la tarea de QA del conjunto de datos de Movie Dialog [133], pero usando tres configuraciones diferentes de conocimiento: usando una base de conocimiento tradicional, usando Wikipedia como fuente de conocimiento o usando IE sobre Wikipedia. Este dataset cuenta con cerca de 100,000 preguntas. Algunos ejemplos se muestran en la Figura 4.5.


```
{"question": "what movies are about ginger rogers?", "answer": ["Top Hat", "Kitty Foyle", "The Barkleys of Broadway"]}
{"question": "which movies can be described by moore?", "answer": ["Fahrenheit 9/11", "Far from Heaven"]}
{"question": "what films can be described by occupation?", "answer": ["Red Dawn", "The Teahouse of the August Moon"]}
{"question": "which films are about jacques tati?", "answer": ["Mon Oncle", "Playtime", "Trafic"]}
```


Figura 4.5: Formato del dataset WikiMovies

Del mismo modo GraphQuestions [134] fue creado utilizando Freebase y un enfoque semi-automático para generar preguntas que exhiben un conjunto de características determinadas (complejidad de la estructura, función, similitud, cardinalidad de la respuesta y paráfrasis). Esto permite generar plantillas de consulta de gráficos sobre una base de conocimientos dada una cantidad deseada de bordes. El resultado final fue un corpus con un total de 5,166 preguntas. La Figura 4.6 muestra algunos ejemplos de este dataset.

Para TriviaQA [135] se recopilaron pares de preguntas y respuestas de 14 sitios web de trivias. Además, las respuestas fueron validadas mediante la recopilación y comparación de

```
"question": "which venues can seat at least 1417 people?", "answer": ["Reno Events Center", "United Spirit Arena", ...
"question": "which legal cases are handled by justice scalia?", "answer": ["Dolan v. United States Postal Service"], ...
"question": "find rockets made by chrysler llc that support low earth orbit.", "answer": ["Saturn I", "Saturn IB"], ...
"question": "the ws07 runner-up was managed by who?", "answer": ["Don Baylor", "Clint Hurdle", "Jim Leyland"], ...
```


Figura 4.6: Formato del dataset GraphQuestions

los resultados de la búsqueda web y artículos de Wikipedia. Esto corpus se compone de cerca de 95,000 preguntas como las que se muestran en la Figura 4.7.


```
"Question": "Bill Berry retired through ill health as a drummer in which band?", "Answer": "REM", ...
"Question": "Which James Bond film features a song by Louis Armstrong?", "On Her Majesty's Secret Service", ...
"Question": "Which country became the first in the world to issue the dreaded parking ticket?", "Answer": "France", ...
"Question": "Who had a 70s No 1 hit with Billy, Don't Be A Hero?", "Answer": "Bo Donaldson and the Heywoods", ...
```


Figura 4.7: Formato del dataset TriviaQA

Stanford Question Answering Dataset (SQuAD) [136, 137] es un conjunto de datos diseñado para tareas de comprensión de lectura. Para este se contrataron a trabajadores colectivos para hacer preguntas sobre un conjunto de artículos de Wikipedia. Luego se les pidió que anotaron las preguntas con el segmento de texto del artículo que forma la respuesta. La primera versión de SQuAD cuenta con más de 100,000 preguntas y en la versión 2 se agregaron más de 50,000 preguntas sin respuesta (Figura 4.8). Estas preguntas fueron generadas de la misma forma que la versión inicial.


```
"question": "How much did Dell spend on acquiring different divisions?", "answers": [{"text": "$13 billion", ...
"question": "What country found bank accounts and real estate owned by the Sassou regime?", "answers": [{"text": "France", ...
"question": "How many hours did it take to force the Russians away during the attack?", "answers": [{"text": "three hours", ...
"question": "The older version of Wayback Machine did not have much new data past what year?", "answers": [{"text": "2008", ...
```


Figura 4.8: Formato del dataset SQuAD

Tomando en cuenta los distintos datasets para QA se seleccionaron TREC, SimpleQuestions, WebQuestions, WikiMovies y una variante de TREC, denominada Curated TREC. La razón principal de haber seleccionado los datasets mencionados radica en el tipo de pregunta que se utiliza en esta investigación. Estas preguntas son tipo factoide, es decir, preguntas que pueden ser respondidas con hechos simples.

Sin embargo, se han encontrado otros datasets como: FreebaseQA [138], ComQA [139], LC-QuAD [140], ComplexWebQuestions [141], MS MARCO [142], Natural Questions [143] y Compositional Freebase Questions (CFQ) [144] que aún no se han explorado y que podrían ser de gran utilidad para esta investigación.

4.2.2. Creación de base de conocimiento

Durante esta investigación los corpus estudiados fueron aquellos comúnmente mencionados en la literatura como: TREC [145, 146], SQuAD [136, 137], entre otros. Sin embargo, dada la necesidadde utilizar *Wikipedia* como base de conocimiento se comenzó con el desarrollo de un corpus utilizando dicha plataforma.

Para ésto, se utilizó la herramienta ***WikiExtractor*** [147], ésta extrae y limpia la información desde la base de datos de dicha fuente. *WikiExtractor* realiza la expansión de plantillas pre-procesando todo el volcado y extrayendo las definiciones en cada una de éstas. Es importante destacar que la tarea de QA será evaluada en función de la respuesta correcta disponible en la base de conocimiento seleccionada.

El *script* se invoca con un archivo de *Wikipedia bump* como argumento. La salida se almacena en varios archivos de tamaño similar en un directorio definido por el usuario. Cada uno de éstos contendrá varios documentos en el formato que se muestra en la Figura 4.9.

```
<doc id="2" url="https://es.wikipedia.org/wiki/Inteligencia_artificial">
Inteligencia artificial
La inteligencia artificial (IA), es la inteligencia llevada a cabo por máquinas. En ciencias de
  ↪ la computación, una máquina "inteligente"
...
</doc>

```

Figura 4.9: Formato de documentos generados por *WikiExtractor*

Adicionalmente, se realizaron modificaciones al formato de almacenamiento del corpus. El formato original consistió en documentos con un formato tipo XML. Sin embargo, se generó el mismo corpus en formato JSON (Figura 4.10) para realizar la inserción de la información a una base de datos tipo sqlite.

```
{"id": "doc1", "text": "text of doc1"}
...
{"id": "docN", "text": "text of docN"}
```

Figura 4.10: Corpus en formato JSON

La información fue almacenada en una única tabla con dos columnas: id y contenido. Esta base de datos almacena 1,540,794 registros (Figura 4.11) los que permitirá acceder y almacenar la información de forma eficiente.

Posteriormente, fue necesario realizar la transformación de texto a valores numéricos. Para este proceso es necesario tokenizar, normalizar y filtrar las *stop-words* y los signos de puntuación en cada documento en el corpus. Es importante hacer hincapié en que las *stop-words* y los signos de puntuación **no fueron eliminados**, únicamente fueron filtrados previo al calculo de TF-IDF, dado que para este índice dicha información no aporta un valor significativo. El resto de los detalles del cálculo de TF-IDF son descritos en la Sección 4.5.1.

Esto permitirá manipular la información de cada tema de forma más sencilla tanto para la generación de preguntas como para la generación de respuestas en idioma español. Es necesario mencionar que este archivo contiene información que no corresponde al dominio previamente definido, por lo tanto, únicamente se trabajará con aquella información que cumpla con tal criterio.

4.2.3. Creación de Word Embedding

Tomando en cuenta la necesidad de utilizar vectores de palabras en idioma español se decidió utilizar Spanish Billion Word Corpus [148] que contiene alrededor de 1.5 billones de palabras, recopiladas de distintas fuentes de información como: SenSem, Ancora Corpus, Tibidabo Treebank y IULA Spanish Treebank; el proyecto OPUS que comprende: los libros

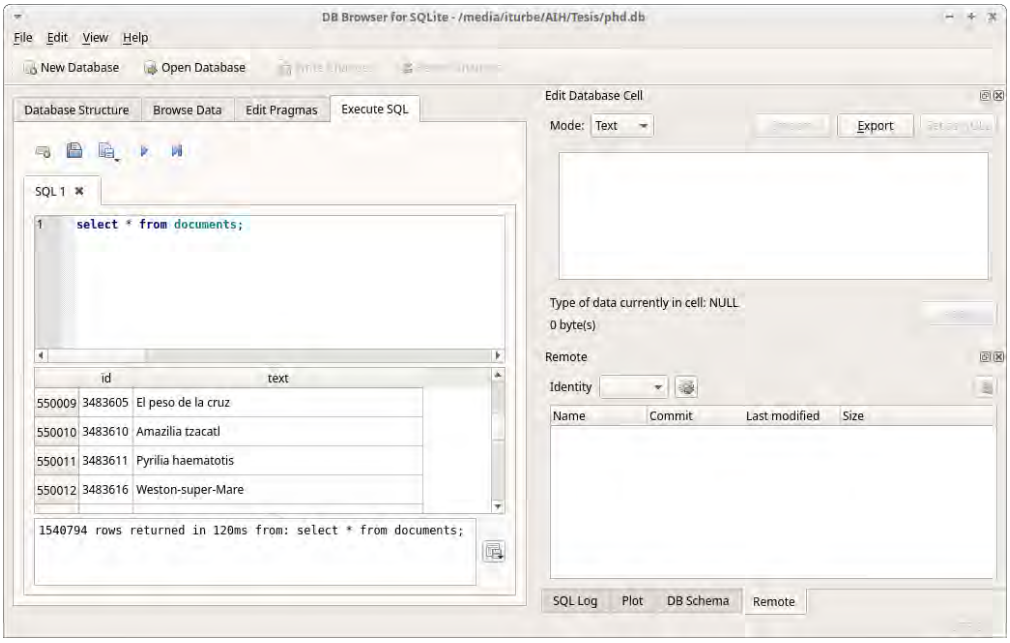


Figura 4.11: Base de datos

alineados por Andras Farkas, la recopilación de textos legislativos de la Unión Europea por el CCI, el corpus News Commentary y los documentos de las Naciones Unidas compilador por Alexandre Rafalovitch y Robert Dale; la porción en español del Parlamento Europeo por Philipp Koehn; y finalmente, fragmentos de Wikipedia, Wikisource y Wikibooks hasta 2015, utilizando Wikipedia Extractor.

Tomando en cuenta lo anterior se replicó el proceso para la creación del mismo utilizando FastText [21], el cual es una biblioteca escrita en C++ para el aprendizaje eficiente de representaciones de palabras y clasificación de oraciones. FastText permite entrenar representaciones supervisadas y no supervisadas de palabras y oraciones. Estas pueden ser usadas para distintas aplicaciones de compresión de datos como características en modelos adicionales, para selección de candidatos o como inicializadores para el aprendizaje por transferencia.

Para esta investigación se utilizó el modelo *Skip-gram* usando muestreo negativo y softmax. Esta herramienta logra un buen rendimiento para las tareas ya mencionadas, pero también en el caso de palabras raras haciendo uso de información a nivel de carácter.

4.3. Reformulación de preguntas múltiples

El objetivo principal consiste en mitigar el problema relacionado a la longitud y la complejidad de las oraciones, dado que estos factores en el PLN repercuten en la obtención de buenos resultados, tal y como lo demuestran [149, 150]. De forma muy similar, sin importar la tecnología aplicada en los mecanismos de búsqueda, los factores ya mencionados también repercuten en la recuperación de información.

Esta actividad fue retomada con la finalidad de mejorar los resultados previamente obtenidos y evaluarlos utilizando las métricas más utilizadas en la literatura, obteniendo como producto final la redacción de un artículo.

Estas fallas en la recuperación de información se observaron al realizar consultas a estos mecanismos con preguntas con dos o más interrogantes como: ***¿Quiénes fueron, cuándo y dónde nacieron los fundadores de Apple?***, los motores de búsqueda tienden a no extraer las respuestas correctas para todas las preguntas en esa cadena. Además, a mayor complejidad, menor es la probabilidad de que se extraiga la respuesta correcta. Por ejemplo, a diferencia de la pregunta 1 en la Tabla 4.2, las preguntas 5 y 6 son más complejas debido al uso de términos multi-palabra, las diferentes entidades nombradas, así como los distintos focos de pregunta que pueden existir.

Tabla 4.2: Ejemplos de cadenas de texto con dos o más preguntas

#	**Pregunta**
1	¿Quiénes fueron, cuándo y dónde nacieron los fundadores de Apple?
2	¿Quién fue conocido como el padre de la computación y cuándo murió?
3	¿Qué es Java y cuándo fue creado?
4	¿Quién fue, cuándo y dónde se creó el primer lenguaje de programación?
5	¿Qué es un bucle y cuál es su sintaxis en Java?
6	¿Cuáles son los lenguajes de programación más utilizados para minería de datos y por qué?

Partiendo de lo anterior fue necesario desarrollar un módulo capaz de identificar los elementos en estas cadenas y generar las respectivas preguntas (Figura 4.1). El objetivo de realizar este tratamiento se debe a la complejidad que una pregunta individual involucra por si misma y la importancia de la clasificación según la intención de ésta para la tarea de QA.

El método propuesto se divide en tres partes: la primera consiste en la generación de un corpus de preguntas utilizando Wikipedia, la segunda parte realiza el análisis y tratamiento de las preguntas para finalmente en la tercera etapa generar las preguntas individuales.

4.3.1. Generación de corpus de preguntas

Wikipedia en español reúne un total de 1,834,308 artículos, anexos y categorías posicionándolo en el noveno lugar de Wikipedias en función del total de páginas de contenido. La Figura 4.12 muestra la cantidad de artículos de la Wikipedia en español a lo largo de los años.

Esto convierte a la Wikipedia en español en una gran fuente de información lo que permitió realizar la construcción de un corpus utilizando esta información, acotándola al dominio de ciencias computacionales y área afines.

Se utilizó ***WikiExtractor***[1] para extraer y limpiar la información desde la base de datos de dicha fuente. *WikiExtractor* convierte artículos de Wikipedia en archivos de texto plano o en tipo json, lo que permite disponer de un conjunto de archivos libres de etiquetas HTML. Esta herramienta se invoca utilizando un archivo de *Wikipedia dump*[2]. La salida es almacenada en distintos archivos, los cuales contendrán varios artículos en formato XML y JSON.

Adicionalmente, se identificó que algunos artículos de Wikipedia contienen una sección denominada *infobox*[3], ésta resume el contenido del sitio o bien, permite al lector encontrar

[1] https://github.com/attardi/wikiextractor

[2] Estos archivos pueden ser descargados desde el siguiente enlace: https://dumps.wikimedia.org/

[3] Ejemplo de artículo de Wikipedia con *Infobox*: https://es.wikipedia.org/wiki/Python

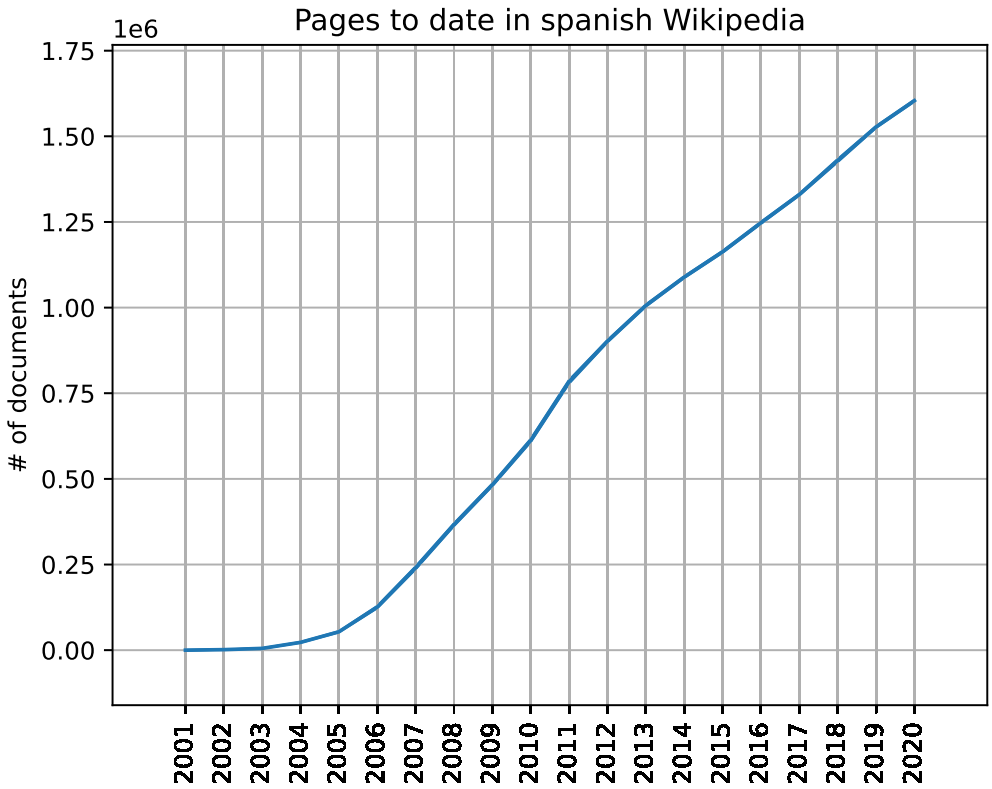


Figura 4.12: Estadísticas de Wikipedia en español

fragmentos de información con mayor facilidad. Tomando en cuenta la naturaleza tabulada de esta sección se generaron preguntas a partir de los elementos que contiene. De modo que el tema principal se identifica por ser el encabezado de la tabla; los elementos de la columna izquierda son focos de pregunta; y aquellos en la columna derecha son las respuestas inmediatal. El proceso de generación implementa el análisis de textos descrito en 4.3.2 y el algoritmo descrito en 4.3.3.

Con esta información se generó un corpus de 500 preguntas que se clasificaron de acuerdo con las clases de preguntas del dataset de clasificación TREC. En la Tabla 4.3 se muestra la distribución de las preguntas en dichas clases.

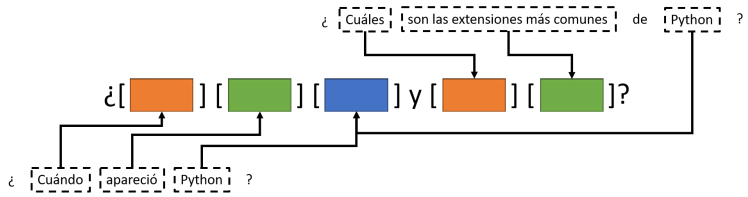
Tabla 4.3: Total de preguntas por clase

ENT	**NUM**	**UBI**	**DESC**	**PER**
131	70	84	76	139

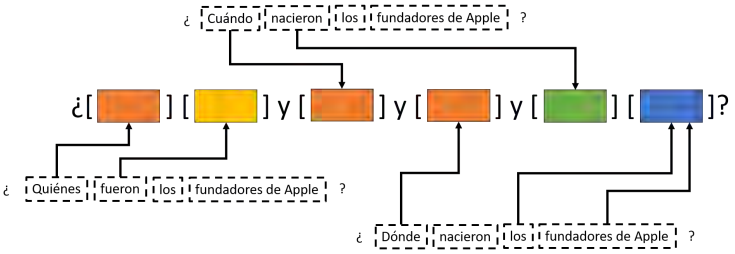
Es importante destacar que, las 500 preguntas generadas son preguntas simples, es decir, tienen un único término interrogativo. Las preguntas múltiples como: *¿Cuándo apareció Python y cuáles son las extensiones comunes?*, *¿Cuál es la última versión estable de Python y por quién fue influido?* o *¿Cuándo apareció Python y cuáles son las extensiones comunes?*, fueron generadas manualmente combinando dos o más de los elementos del corpus.

Por lo tanto, para crear una pregunta múltiple se utiliza como punto de partida la selección de entre dos y tres preguntas del mismo tema, de las dos preguntas mostradas en la

Figura 4.13 se identifican las palabras interrogativas y se posicionan de tal forma que todas las preguntas mantengan coherencias entre ellas mismas; siendo separadas por comas y la conjunción *y*. Seguido de cada palabra interrogativa se incrusta el foco de la pregunta en caso de que los focos de las preguntas sean diferentes (4.13*(a)*); Por el contrario, si el foco de la pregunta concuerda con los demás focos de preguntas, este es agregado después de la última palabra interrogativa (4.13*(b)*). Finalmente, el tema o entidad principal sobre el cual se están formulando las preguntas puede agregarse inmediatamente después del primer foco de pregunta (4.13*(a)*) o bien, al final de la oración (4.13*(b)*).



(a) Dos preguntas



(b) Tres preguntas

Figura 4.13: Construcción de preguntas múltiples

Tomando en cuenta esto, se obtuvieron dos corpus resultantes: el primero de éstos se compone de preguntas simples, el cual, fue utilizado para evaluar la clasificación de preguntas en español utilizando una Red Neuronal Convolucional; y el segundo corpus consta de con 1140 preguntas múltiples, el cual, fue utilizado para evaluar el módulo de reformulación de preguntas complejas/múltiples.

4.3.2. Análisis y tratamiento de preguntas

En la etapa de análisis y tratamiento de preguntas fue necesario extraer la categoría gramatical de cada palabra utilizando la herramienta *Freeling* [127]. Este módulo se ejecutan las siguientes tareas en paralelo: tokenización, etiquetado (*Part-of-Speech*) PoS y detección de Entidades Nombradas.

Disponer de esta información es fundamental para que el módulo pueda identificar las distintas etiquetas PoS y de esta forma poder generar las nuevas preguntas. La Tabla 4.4 describe las distintas posiciones que conforman las etiquetas de la categoría verbo.

Del mismo modo, se utilizó *Freeling* para realizar el Reconocimiento y Clasificación de Entidades Nombradas.Esta actividad tuvo como objetivo principal mejorar el desempeño del modelo desarrollado. Si bien la herramienta tiene un porcentaje elevado en esta tarea, fue necesario desarrollar una heurística de análisis por omisión de elementos (APO), la cual también fue utilizada para la generación de las nuevas preguntas.

Tabla 4.4: Elementos de la categoría verbo

Verb		
Position	***Atribute***	***Values***
0	*Category*	***V**:verb*
1	*Type*	***M**:main; **A**:auxiliary; **S**:semiauxiliary*
2	*Mood*	***I**:indicative; **S**:subjunctive; **M**:imperative; **P**:participle; **G**:gerund; **N**:infinitive*
3	*Tense*	***P**:present; **I**:imperfect; **F**:future; **S**:past; **C**:conditional*
4	*Person*	**1**:1; **2**:2; **3**:3
5	*Num*	***S**:singular; **P**:plural*
6	*Gen*	***F**:femenine; **M**:masculine; **C**:common*

4.3.3. Identificación de elementos y generación de nuevas preguntas

Para la reformulación de preguntas se desarrolló un algoritmo capaz de determinar los elementos gramaticales necesarios como: adjetivos, determinantes, pronombres o adverbios para crear una oración coherente. Para esto se realiza la detección del género, número y tiempo de las etiquetas PoS previamente obtenidas.

A su vez, el algoritmo realiza una búsqueda de palabras clave dentro de la oración, estas palabras denotan los términos interrogativos presentes en la cadena de texto. Dado que una pregunta puede utilizar distintas palabras clave para un determinado tipo de pregunta se crearon repositorios de palabras clave según el tipo de pregunta, por ejemplo: fecha[*cuándo, fecha, año, mes, día*], ubicación[*dónde, lugar, país, sede*], entre otros.

Una vez identificada la información anterior el algoritmo localiza el Foco de la Pregunta, el cual, es una declaración o frase que hace fluir la pregunta. Debe estar relacionada con el contenido o a los resultados que se desean obtener. Para esto se utilizó el algoritmo *LocalMax*[151] sobre el corpus construido.

LocalMax algorithm asume que los TMP tienen un índice alto de adherencia entre ellos. Los autores proponen una métrica de asociación denominada *Symmetrical Conditional Probability* (SCP) para medir la correlación entre dos palabras utilizando la Ecuación 4.1:

$$SCP = p(x \mid y) \cdot p(y \mid x) = \frac{p(x,y)^2}{p(x) \cdot p(y)} \tag{4.1}$$

Donde $p(x,y)$; $p(x)$; y $p(y)$ son las probabilidades de que el bigrama (x,y), el unigrama $p(x)$ y el unigrama $p(y)$ aparezcan en el corpus respectivamente. Para generalizar esta medida de n-gramas, los autores introducen la Normalización de Dispersión Justa (*Fair Dispersion Normalization*), que divide un n-grama $w_1, w_2, ..., w_n$ en diferentes puntos de dispersión y lo considera como combinaciones de las dos partes. FDS utiliza el promedio de los productos para normalizar la medida de asociación para un n-grama dado (Ecuación 4.2). Para medir la cohesión entre palabras en un n-grama, se calcula el promedio de los productos para las dos partes en diferentes puntos de dispersión del n-grama (Ecuación 4.3).

$$SCP\_f\left(w_1 w_2 \dots w_n\right) = \frac{p\left(w_1 w_2 \dots w_n\right)^2}{Avp} \tag{4.2}$$

$$Avp = \frac{1}{n-1} \sum_{i=1}^{n-1} p\left(w_1 \dots w_i\right) \cdot p\left(w_{i+1} \dots w_n\right) \tag{4.3}$$

Donde n es la longitud del n-grama y $p(w_1, \dots, w_n)$ es la probabilidad de la secuencia de palabras w_i. Basado en FDS, el algoritmo LocalMax trata de encontrar un n-grama con el SCP_f más fuerte que cualquier (n-1)-grama en éste y cualquier (n+1)-grama que lo contenga, permitiendo identificar términos multi-palabra y con ellos evitar en mayor medida la pérdida de información. A este proceso se suma una heurística basada en el análisis por omisión de elementos (APO) utilizando *stop words*, etiquetas PoS y NERC.

Previo a la obtención de los elementos mencionados, se desarrolló un conjunto de plantillas mediante la adición y concatenación de elementos para brindar concordancia a las oraciones, esto permite generar distintos tipos de preguntas a partir de la consulta original. La Tabla 4.5 muestra algunos de los patrones mencionados.

Tabla 4.5: Ejemplos de patrones para generación de preguntas

Patrones de preguntas
['FIA', 'PT', 'VS', 'DA', 'AO','FP, TMP, EN', 'FIT']
['FIA', 'PT', 'VS', 'DA', 'NP','FP, TMP, EN', 'FIT']
['FIA', 'PT', 'VS', 'DI', 'FP, TMP, EN', 'FIT']
['FIA', 'PT', 'VM', 'FP, TMP, EN', 'FIT']
['FIA', 'PT', 'VM', 'DA','FP, TMP, EN', 'FIT']
['FIA', 'PT', 'P0', 'FP, TMP, EN', 'FIT']
['FIA', 'PT', 'P0', 'VM','FP, TMP, EN', 'FIT']
['FIA', 'PT', 'P0', 'VM', 'DA','FP, TMP, EN', 'FIT']
['FIA', 'SP', 'PT', 'VS','FP, TMP, EN', 'FIT']
['FIA', 'SP', 'PT', 'VS', 'VM','FP, TMP, EN', 'FIT']
['FIA', 'SP', 'PT', 'VM','FP, TMP, EN', 'FIT']
['FIA', 'SP', 'PT', 'VM', 'DA', 'FP, TMP, EN', 'FIT']
['FIA', 'SP', 'DT','FP, TMP, EN', 'FIT']
['FIA', 'DT', 'FP, TMP, EN', 'FIT']
['FIA', 'DI', 'FP, TMP, EN', 'FIT']
['FIA', 'NC', 'SP', 'FP, TMP, EN', 'FIT']

Donde: *FIA* representa al signo de interrogación de apertura (¿), *PT* corresponde a la Palabra Interrogativa que será requerida según el tipo de pregunta, *VS,VM, P0, DT,...,n* indican el verbo, determinante o elemento gramatical para generar una oración coherente. Para el caso de los *infobox*: *FP, EN* y *TMP* foco de la pregunta, término multi-palabra y Entidades Nombradas respectivamente, se determinan según la posición en la tabla, la columna izquierda (FP), el título del *infobox* (EN); finalmente, *FIT* representa al signo de interrogación de cierre (?).

Cabe destacar que es necesario analizar cada elemento de forma individual de tal forma que se identifique información suficiente para generar preguntas coherentes, es decir, preguntas que respeten aspectos de concordancia de género y número, así como el uso adecuado de verbos, determinantes, preposiciones, entre otros. Del mismo modo, de ser necesario el módulo elimina el ruido tanto para la generación de preguntas como en sus respuestas.

4.4. Clasificación de preguntas

La importancia de identificar el tipo de pregunta que será procesada por un sistema de *Question-Answering* es fundamental para esta tarea tal y como lo demuestran [52–55, 96, 112, 119–122]. Por consecuente, se utilizó un modelo de Red Neuronal Convolucional para realizar la clasificación de preguntas en español.

Para comprender el funcionamiento de las CNN en [10, 152] se describe esta arquitectura, en la que, cada capa de la red es tridimensional, que tiene una extensión espacial y una profundidad correspondiente al número de características. La noción de profundidad de una sola capa en una CNN es diferente de la noción de profundidad dependiendo del número de capas.

En la capa de entrada, estas características corresponden a canales de color como RGB (rojo, verde, azul), y en los canales ocultos estas características representan mapas de características ocultos que codifican varios tipos de formas en la imagen. La arquitectura contiene dos tipos de capas: convolución y submuestreo.

Para las capas de convolución es necesario definir una operación de convolución, en la que se utiliza un filtro para asignar activaciones de una capa a la siguiente. Una operación de convolución utiliza un filtro de peso tridimensional con la misma profundidad que la capa actual, pero con una extensión espacial más pequeña. El producto punto entre todos los pesos en el filtro y cualquier opción de región espacial (del mismo tamaño que el filtro) en una capa define el valor del estado oculto en la siguiente capa (después de aplicar una función de activación como ReLU). La operación entre el filtro y las regiones espaciales de una capa se realiza en cada posición posible para definir la siguiente capa (en la que las activaciones conservan sus relaciones espaciales de la capa anterior).

Generalmente, en la tarea Procesamiento del Lenguaje Natural para la clasificación de texto [11] se utilizan filtros que se mueven sobre filas enteras de una matriz basada en palabras. Donde, el ancho de cada filtro suele ser el mismo que el ancho de la matriz de entrada. La altura o el tamaño de la región puede variar, pero el deslizamiento de las ventanas suele ser de dos a cinco palabras a la vez.

En esta investigación se utilizaron distintos conjuntos de preguntas de p palabras y la longitud de las oraciones de entrada definida por n palabras, de las cuales, cada palabra es representada por un vector, en este caso un Word Embedding en idioma español descrito en 4.2.3.

Por lo tanto, se obtiene una matriz de $n \cdot k$, donde k corresponde a la longitud del Word Embedding (300). A esto se añade el batch denotado por b, que significa el total de entradas (preguntas) que va a procesar la red neuronal de forma simultáneamente. Se aplica un padding a todas aquellas oraciones que no tengan una longitud máxima definida con anterioridad.

Para la operación de convolución se utiliza la entrada antes mencionada $n \cdot k$ ahora denominada x, y una matriz de pesos $W(m \cdot k)$, que genera como salida un vector h que se obtiene con las Ecuaciones 4.4 y 4.5.

$$h_{i,1} = \sum_{j=1}^{m} \sum_{l=1}^{k} w_{j,l} x_{i+j-1,l} \tag{4.4}$$

$$h = W * x + b \tag{4.5}$$

A su vez, se definen diferentes tamaños de filtro que son ejecutados en paralelo. Cada convolución tiene como salida un vector oculto de dimensión $1 \cdot n$, los cuales con concatenados

para generar la entrada de la siguiente capa con una dimensión $q \cdot n$, donde q es el número de capas paralelas que serán utilizadas. Por lo tanto, asumiendo que la salida de la capa h es de dimensión $q \cdot n$ la capa de pooling producirá una salida de tamaño $q \cdot 1$ denominada h' (Ecuación 4.6).

$$h'_{i,1} = \left\{ \max \left(h^{(i)} \right) where 1 \leq i \leq q \right\} \tag{4.6}$$

La Figura 4.1 muestra la arquitectura utilizada para esta Red Neuronal Convolucional. En esta se aprecia el vector correspondiente a los tokens de la oración de entrada, seguido del Word Embedding utilizado para calcular el vector de valor real para cada palabra utilizando las 300 dimensiones del mismo. Posteriormente, las capas de convolución y max pooling para finalmente llegar a la capa completamente conectada y softmax. Con un total de 100 filtros con dimensiones de tamaños 2, 3 y 4, una dimensión de la capa de embedding definida en 300, con un tamaño de entrada definido en función del tamaño del vocabulario, un dropout de 0.5, aplicando una convolución 2D. De esta forma, la red neuronal será capaz de clasificar preguntas en idioma español.

De los datasets ya mencionados, únicamente TREC se centra en la tarea de clasificación de preguntas, por lo tanto, se utilizaron las clases definidas para este conjunto de datos: Entidad, Descripción, Persona, Numérico, Ubicación y Abreviación. El resto de los datasets fueron clasificados manualmente y con la ayuda de una algoritmo basado en el uso de palabras clave para cada categoría. La Tabla 4.6 muestra el total de elementos por clase por cada dataset.

Tabla 4.6: Distribución de elementos por clase

	Clases					
	ENT	**DESC**	**PERS**	**NUM**	**UBI**	**ABR**
TREC	1,052	1,042	1,033	823	735	77
WM	45,441	33,459	11,700	5,263	319	3
WQ	542	494	374	202	154	0
SQ	37,012	18,983	18,648	10,752	1,272	1
CTREC	381	327	296	246	212	24

4.5. Recuperación de preguntas y Generación de respuestas

En esta sección se describe la arquitectura utilizada en la Red Neuronal Recurrente dividida en 2 módulos: recuperación de documentos y generación de respuestas. La arquitectura se muestra en la Figura 4.1.

4.5.1. Módulo de recuperación de documentos

La siguiente actividad, denominada *Recuperación de Documentos* tuvo como objetivo extraer los n temas relacionados a partir de una pregunta dada. En la arquitectura de un sistema QA estos módulos son fundamentales para minimizar los tiempos de respuesta en la recuperación de información, ya sea que ésta se encuentre disponible de forma local o que se realicen consultas a internet.

Este módulo implementa TF-IDF (del inglés, *Term-Frequency - Inverse Term Frequency*)[153, 154], el cual es una medida que expresa la relevancia de una palabra para un documento en un

corpus. Esta medida es comúnmente utilizada como factor de ponderación en las búsquedas de recuperación de información, minería de texto, entre otras. TF-IDF aumenta proporcionalmente según el número de veces que una palabra aparece en el documento y se compensa con la frecuencia de la palabra en el corpus.

Term Frequency (TF) se calcula como el número de veces que aparece una palabra en un documento dividido por el número total de palabras en éste. Cada documento tiene su propia frecuencia de término (Ecuación 4.7).

$$TF = \frac{(number\_of\_repetitions\_of\_word\_in\_a\_document)}{(number\_of\_words\_in\_a\_document)} \tag{4.7}$$

Por otra parte, *Inverse Document Frequency* (IDF) (Ecuación 4.8) se calcula como el logaritmo del número de documentos dividido por el número de documentos que contienen la palabra. IDF determina el peso de las palabras raras en todos los documentos del corpus.

$$IDF = \log\left(\frac{number\_of\_documents}{number\_of\_documents\_containing\_the\_word}\right) \tag{4.8}$$

Finalmente, TF-IDF se calcula como el producto entre TF e IDF.

$$TF - IDF = (TF)(IDF) \tag{4.9}$$

Buscando mejorar la velocidad de ejecución sin comprometer la memoria del sistema basándose en la literatura, se utilizó una función Hash [155] para mapear los bigramas a 16,777,216 de contenedores con un *unsigned murmur3 Hash*.

Para mostrar el proceso al que se somete una cadena de texto se propone el siguiente ejemplo, donde se recibe como entrada: ***fundadores de apple***. A diferencia del procesamiento para el corpus, esta cadena de texto no es sometida al filtrado de *stop-words* y signos de puntuación.

El siguiente paso es generar un ID único para cada bigrama en la entrada. Por ejemplo, la Tabla 4.7 muestra los valores correspondientes a la cadena de entrada. Estos identificadores son generados utilizando la función *murmur3 Harsh*.

Tabla 4.7: Cálculo de TF-IDF

Bigrama	**ID**	**Repeticiones (r)**	**TF=log1p(r)**	**NS**		**IDF**	**TF-IDF**
fundadores	4782522	1	0.69314718	15436	98.815049	4.593250	3.183798
fundadores de	1322473	1	0.69314718	7049	217.567984	5.382511	3.730873
de	6388646	1	0.69314718	1510368	0.020145	-3.904795	-2.706598
de apple	14813462	1	0.69314718	1456	1056.875043	6.963072	4.826434
apple	1501328	1	0.69314718	4081	376.507044	5.930937	4.111012

Posteriormente, se realiza el ordenamiento ascendente de los identificadores para realizar el calculo de TF-IDF sobre la información de entrada. En términos de este ejemplo ningún bigrama se repite por lo cual la frecuencia de cada uno de éstos es igual a 1. Aplicando la función *log1p* de *Numpy* se obtiene como resultado 0.69314718 respectivamente, este valor corresponde al valor de TF.

Posteriormente, para calcular el valor de IDF se utiliza la Ecuación 4.10, esta versión de IDF tiene como objetivo calcular este valor a partir de la consulta de entrada.

$$IDF = \log\left(\frac{number\_of\_documents - ns + 0.5}{ns + 0.5}\right) \tag{4.10}$$

Donde, ns es el total de repeticiones de un n-grama en el corpus y 0.5 representa el esquema de ponderación recomendado para tareas de este estilo. Una vez obtenidos TF e IDF se procede a calcular al valor final únicamente como el producto de ambas variables.

Finalmente, para obtener un valor significativo en relación al corpus se calcula el producto punto entre los valores obtenidos en la Tabla 4.7 contra todos los valores de cada elemento en el corpus. El resultado de este módulo será un conjunto de documentos ordenados de mayor a menor según el valor de TF-IDF obtenido, la Figura 4.14 muestra una captura de pantalla de la consola mostrando la información correspondiente a la consulta de este ejemplo.

```
E:  fundadores de apple
+---+---------+------------+--------+
| # |    ID   |   Título   | Score  |
+---+---------+------------+--------+
| 1 |   9551  |   Apple    | 314.92 |
| 2 |  41415  | Macintosh  | 254.59 |
| 3 |  221527 |  Apple II  | 225.22 |
| 4 |   1948  |   MacOS    | 224.45 |
| 5 | 3938591 | Apple IIGS | 217.36 |
+---+---------+------------+--------+
[25/May/2020 14:19:38] "POST / HTTP/1.1" 200 6023
```

Figura 4.14: Puntuación de documentos recuperados

4.5.2. Módulo de generación de respuestas

El tipo de Red Neuronal Recurrente utilizado fue de tipo LSTM bidireccional de tres capas con 128 unidades ocultas para la codificación de las preguntas y de los documentos recuperados. En conjunto al modelo se implementó un módulo para tareas de Procesamiento de Lenguaje Natural como: tokenización, lematización, etiquetado PoS y reconocimiento y clasificación de Entidades Nombradas utilizando Freeling.

Este modelo parte de una pregunta de entrada y un conjunto de documentos relacionados, los cuales, contienen un número finito de tokens. La diferencia entre éstos es que, en el caso de los documentos se tiene un número finito de párrafos con sus respectivos tokens. Los tokens por párrafo son convertidos a secuencias de vectores de características que son enviados a la Bi-LSTM, esto permite extraer información de contexto útil alrededor de un determinado token, resultando en la concatenación de las unidades ocultas de cada capa. Estos vectores de características consisten en:

- El valor real con relación a los *Word Embeddings*. Cabe recordar que los *Word Embeddings* utilizados en esta investigación fueron entrenados a partir del *Spanish Billion Word Corpus* utilizando GloVe de 300 dimensiones.

- Por otra parte, de forma binaria se identifica si una determinada palabra se encuentra exactamente (*Exact Match*) ha sido escrita en la pregunta de entrada o bien en minúscu-

las o en su lema. De forma muy similar se identifican ciertas propiedades de una palabra como el etiquetado PoS, entidades nombrada y la frecuencia de los términos.

- Posteriormente, se implementó el alineamiento de *embeddings* de preguntas (*aligned Question embedding*, AQE) propuesto por Lee et al. [156]. Este módulo tiene como objetivo mitigar los errores que se pueden producir con *Exact Match* (incluso agregando las características ya mencionadas), AQE agrega alineaciones suaves entre palabras similares, pero no idénticas, en su gran mayoría sinónimos; pero también algunos casos de generalización, por ejemplo: automóvil y vehículo, rana y anfibio. La AQE $f_{align}(p_i) = \sum_j a_{i,j}\mathbf{E}(q_j)$, donde la puntuación de atención $a_{i,j}$ captura la similitud entre un token p_i y cada palabra en una pregunta q_j. De forma más específica, $a_{i,j}$ se calcula como el producto punto entre mapeos no lineales de *word embeddings* tal y como se muestra en la Ecuación 4.11. Donde, $\alpha(\cdot)$ es una única capa densa ReLU con no linealidad.

$$a_{i,j} = \frac{\exp\left(\alpha\left(\mathbf{E}\left(p_i\right)\right) \cdot \alpha\left(\mathbf{E}\left(q_j\right)\right)\right)}{\sum_{j'} \exp\left(\alpha\left(\mathbf{E}\left(p_i\right)\right) \cdot \alpha\left(\mathbf{E}\left(q_{j'}\right)\right)\right)} \tag{4.11}$$

De forma muy similar, la codificación de las preguntas se realiza mediante una Red Neuronal Recurrente sobres los *word embeddings* de q_j y se combina con el resultado de las unidades ocultas, creando un único vector $\mathbf{q}$. Dado $\mathbf{q} = \sum_j b_j \mathbf{q}_j$, b_j codifica la importancia de cada palabra en la pregunta y $\mathbf{w}$ corresponde al vector de pesos que debe ser aprendido (Ecuación 4.12).

$$b_j = \frac{\exp\left(\mathbf{w} \cdot \mathbf{q}_j\right)}{\sum_{j'} \exp\left(\mathbf{w} \cdot \mathbf{q}_{j'}\right)} \tag{4.12}$$

Una vez terminada la codificación las preguntas y los párrafos, se procede a la predicción de la respuesta tomando como base cada párrafo de todo el documento de forma individual, para de esta forma predecir correctamente el token o conjunto de tokens correspondientes a la respuesta. Tomando como entradas los vectores de párrafos y preguntas, se entrenaron dos clasificadores de forma independiente para predecir los dos extremos del tramo. Se utilizó un término bilineal para capturar la similitud entre el párrafo y la pregunta y posteriormente calcula las probabilidades de inicio y fin de cada token. Durante este proceso de predicción, todas aquellas respuestas candidatas que excedan una longitud predefinida serán descartadas. Este proceso culmina con el cálculo de las puntuaciones utilizando un exponencial no normalizado y se toma argmax sobre todos los fragmentos de párrafo.

Una vez concluidas las 30 épocas del entrenamiento se obtuvieron los resultados que se muestran en la Figura 4.15.

Siendo parte fundamental del QA, la recuperación de documentos permito al modelo identificar y recuperar documentos o fragmentos de texto relevantes a partir de una consulta determinada. Considerando los resultados obtenidos anteriormente (Tabla 4.7) se decidió desarrollar un algoritmo capaz de determinar si un documento está relacionado o no a las Ciencias Computacionales o áreas afines.

Debido a que WikiExtractor no etiqueta los artículos a una determinada área, se recolectaron distintos términos relacionados a las áreas anteriormente mencionadas desde distintas fuentes de información, por ejemplo: glosarios de Wikipedia o sitios web de compendios de vocabularios. Con esta información se crearon cinco subconjuntos de palabras:

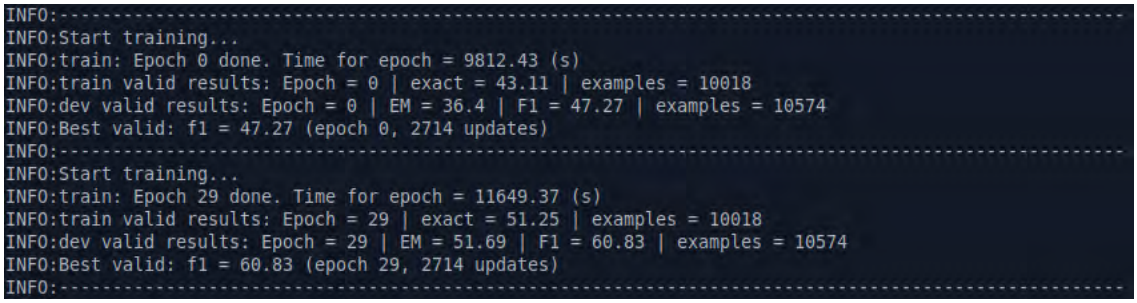


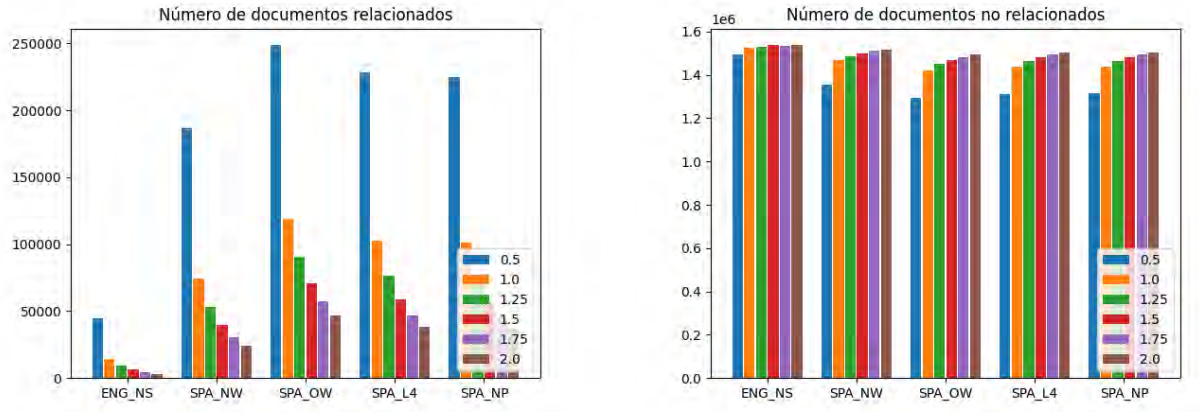
Figura 4.15: Resultados épocas 0 y 29

- **ENG_NS**: contiene un total de 1,257 palabras en inglés, sin ningún tipo de procesamiento. Es decir, si el término contiene un acrónimo entre paréntesis, por ejemplo: *Random Access Memory (RAM)*, éste se mantiene de esa forma.
- **SPA_NW**: consta de todos los términos en idioma español (662) a excepción de aquellos extraídos de los glosarios de Wikipedia.
- **SPA_OW**: contiene únicamente los términos de Wikipedia en español, un total de 595.
- **SPA_NP**: comprende todos los términos en español (1,257), pero elimina los paréntesis y el contenido en éstos. Por ejemplo: *Memoria de Acceso Aleatorio (RAM)* se conserva como *Memoria de Acceso Aleatorio*.

Utilizando las métricas para el cálculo de TF-IDF anteriormente descritas (4.10) se obtuvo una matriz de valores de $n \cdot m$, donde n corresponde al número de artículos de Wikipedia y m al número de términos relacionados.

$$
\begin{array}{c}
 \\
AID_1 \\
AID_2 \\
\vdots \\
AID_{65789} \\
AID_{65790} \\
\vdots \\
AID_n
\end{array}
\begin{array}{c}
\begin{array}{ccccccc}
kw_1 & kw_2 & \cdots & kw_{354} & kw_{355} & \cdots & kw_n
\end{array} \\
\begin{pmatrix}
0 & 1 & \cdots & 0 & 0 & \cdots & 1 \\
1 & 0 & \cdots & 0 & 0 & \cdots & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0 & 0 & \cdots & 0 & 1 & \cdots & 0 \\
0 & 0 & \cdots & 1 & 0 & \cdots & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0 & 0 & \cdots & 0 & 0 & \cdots & 0
\end{pmatrix}
\end{array}
$$

Adicionalmente, se definieron seis umbrales para determinar si un artículo está relacionado o no. Dichos umbrales son: 0.5, 1.0, 1.25, 1.5, 1.75 y 2.0. La Figura 4.16 muestra el número de documentos relacionados y no relacionados respectivamente. En la Figura 4.16(a) se aprecia que el subconjunto con menor número de documentos relacionados es ENG_NS y aquel con mayor cantidad es SPA_OW. En contraste, la Figura 4.16(b) muestra que los mismos subconjuntos tienen mayor y menor número de documentos no relacionados respectivamente.



(a) Relacionados

(b) No relacionados

Figura 4.16: Número de documentos por subconjunto y umbral

Para validar la mejora en la recuperación de documentos se realizó un experimento con las mismas 150 preguntas seleccionadas en la anterior etapa de evaluación. Cada pregunta recupera los 20 documentos mejor calificados de acuerdo al producto punto de las puntuaciones TF-IDF entre todos los artículos de Wikipedia; asignando 0 si el documento no supera el umbral definido o 1 si el umbral es superado. De esta forma, si un documento con un valor igual a 0 es recuperado, el algoritmo itera hasta que dentro de los 20 elementos recuperados no se encuentre ninguno con dicho valor. Por lo tanto, se evaluaron 3000 elementos por cada umbral.

Para realizar la evaluación se utilizaron las métricas de *precision*, *recall* y *F-Score*, Ecuaciones 4.13-4.15 respectivamente.

$$\text{precision} = \frac{|\{\text{ relevant documents }\} \cap \{\text{ retrieved documents }\}|}{|\{\text{ retrieved documents }\}|} \tag{4.13}$$

$$\text{recall} = \frac{|\{\text{ relevant documents }\} \cap \{\text{ retrieved documents }\}|}{|\{\text{ relevant documents }\}|} \tag{4.14}$$

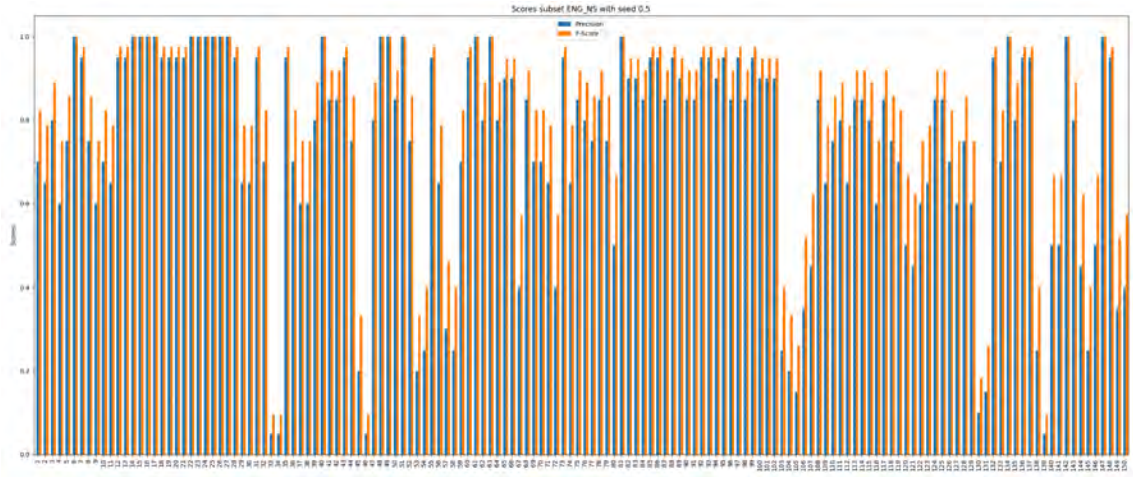
$$F = \frac{2 \cdot \text{ precision } \cdot \text{ recall}}{(\text{ precision } + \text{ recall })} \tag{4.15}$$

Cada uno de los documentos recuperados por las 150 preguntas de evaluación debieron ser clasificados manualmente como relevante (1) o no relevante (0) tal y como se muestra en la Tabla 4.8. Esta Tabla muestra el formato actual del archivo CSV para cada subconjunto y cada umbral. Los datos mostrados en ésta corresponden al subconjunto SPA_NW con umbral 1.75. Donde la columna P representa el valor obtenido por el algoritmo y R el valor real del artículo.

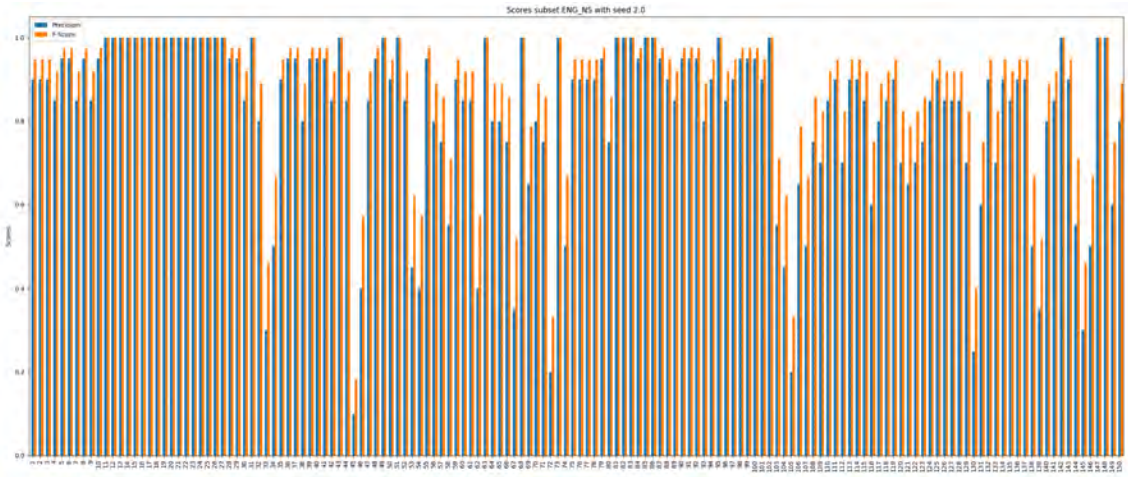
Cada pregunta de cada subconjunto y umbral fue evaluado obteniendo los resultados que se muestran en la Figura 4.17 en el caso del subconjunto ENG_NS y la Figura 4.18 para el subconjunto SPA_OW.

Tabla 4.8: CSV para evaluación

Pregunta	**Seed**	**AID**	**Título**	**Score**	**P**	**R**
¿Cuando apareció python?	1.75	2330	Python	162.044	1	1
¿Cuando apareció python?	1.75	3391	Monty Python	144.423	1	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮
¿Quién inventó javascript?	1.75	1568	JavaScript	181.690	1	1
¿Quién inventó javascript?	1.75	798701	JavaScript no obstructivo	133.736	1	1
⋮	⋮	⋮	⋮	⋮	⋮	⋮
¿qué es el polimorfismo dinámico?	1.75	1162613	Polimorfismo (informática)	496.417	1	1
¿qué es el polimorfismo dinámico?	1.75	6034993	Polimorfismo (biología)	422.931	1	0

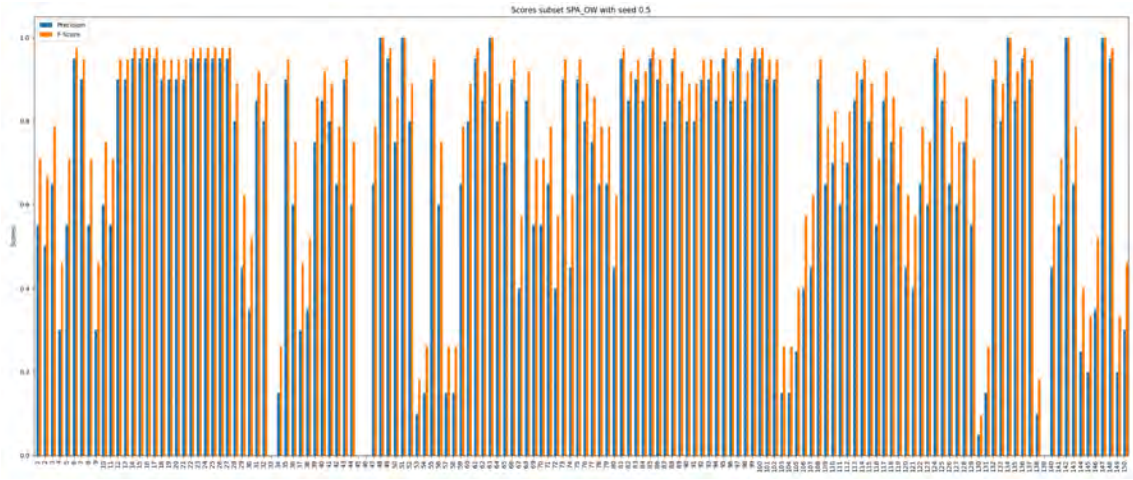


(a) Umbral 0.5

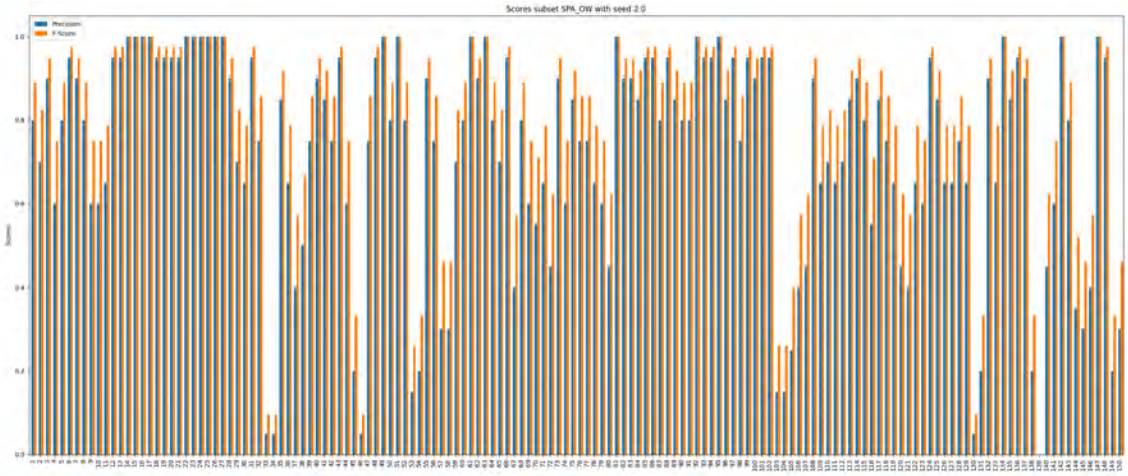


(b) Umbral 2.0

Figura 4.17: *Precision* y *F-Score* de cada pregunta (ENG_NS)



(a) Umbral 0.5



(b) Umbral 2.0

Figura 4.18: *Precision* y *F-Score* de cada pregunta (SPA_OW)

Tabla 4.9: *Precision* y *F-Score* promedio para cada subconjunto y umbral

	ENG_NS		**SPA_NW**		**SPA_OW**		**SPA_L4**		**SPA_NP**	
	AVG P	**AVG F**	**AVG P**	**AVG F**	**AVG P**	**AVG F**	**AVG P**	**AVG F**	**AVG P**	**AVG F**
0.5	0.74	0.82	0.68	0.77	0.68	0.77	0.68	0.77	0.68	0.77
1	0.77	0.85	0.71	0.79	0.70	0.79	0.70	0.79	0.70	0.79
1.25	0.79	0.86	0.71	0.80	0.71	0.79	0.71	0.79	0.71	0.79
1.5	0.80	0.87	0.72	0.80	0.71	0.79	0.72	0.80	0.71	0.80
1.75	0.81	0.87	0.73	0.81	0.71	0.80	0.72	0.80	0.72	0.81
2	**0.82**	**0.88**	**0.73**	**0.81**	**0.72**	**0.80**	**0.73**	**0.81**	**0.73**	**0.81**

Sorpresivamente el subconjunto ENG_NS con el umbral 2 obtuvo la mayor puntuación tanto en *precision* como en *F-Score* y SPA_OW obtuvo el valor más bajo con 0.72 y 0.80 respectivamente. Es importante destacar que esta subtarea tuvo como objetivo recuperar únicamente documentos relacionados sin importar si estos tienen relación directa con la pregunta. Todos los subconjuntos con el umbral 2 superaron con más del 20 % a los resultados anteriores.

4.6. Interfaz Web

Finalmente, se desarrolló la interfaz web para la interacción del usuario con el sistema, se utilizó el lenguaje de programación Python [157] en conjunto con el framework Django [158] en las versiones 3.6.9 y 3.0.3 respectivamente. La Figura 4.19 muestra la pantalla principal del sistema, en la cual, se encuentra un campo de texto para poder escribir la pregunta y enviarla al *Backend* del sistema que posteriormente obtener una respuesta.

Figura 4.19: Página principal de la interfaz web

Una vez enviada la información se muestra la pantalla de la Figura 4.20 que indica la clase correspondiente (color verde) para cada una de las preguntas en caso de ser múltiple (color azul) o una sola pregunta en caso de ser simple. En esta interfaz del lado izquierdo se muestran las primera tres respuestas generadas por nuestro modelo, además de información como: la puntuación que obtuvo la respuesta, el ID del documento en la base de datos del cual se obtuvo, el título del mismo y el párrafo del cual se extrajo. Por otra parte, del lado derecho se muestran los primeros tres enlaces de una búsqueda en Google.

Este proceso toma en promedio 17.24 segundos para una pregunta múltiple y 11.51 segundos para una pregunta simple. Sin embargo, aún no se han realizado pruebas relacionadas al rendimiendo con más usuarios de forma simultanea.

Figura 4.20: Interfaz de respuestas

Capítulo 5

Experimentación y Resultados

En este capítulo se describen los experimentos realizados para evaluar los distintos módulos desarrollados para la tarea de pregunta-respuesta. Este se estructura de la siguiente forma: la sección 5.1 describe las métricas y resultados logrados para la reformulación de preguntas. Seguido de esto, en la sección 5.2 describe los resultados obtenidos en la clasificación de preguntas. Finalmente, la sección 5.3 se muestran los resultados para la generación de respuestas.

5.1. Resultados de reformulación de preguntas

Para evaluar la reformulación de preguntas se seleccionaron 250 preguntas en español de forma aleatoria del corpus de preguntas múltiples generado en 4.3.1. Para medir la efectividad del método se seleccionaron las siguientes métricas con base en la literatura, si bien, éstas fueron originalmente creadas para evaluar otras tareas del PLN, han logrado buenos resultados en esta tarea. Estas métricas realizan el cálculo considerando una oración de referencia y una oración candidata.

5.1.1. BLEU

Esta métrica, del inglés *BiLingual Evaluation Understudy* (Ecuación 5.13) fue originalmente diseñada para evaluar la calidad de las traducciones hechas por sistemas computacionales. La Tabla 5.1 muestra los promedios obtenidos para cada categoría BLEU, donde: 1, 2, 3 y 4 representa la secuencia de n-gramas para evaluar una entrada, unigramas, bigramas, trigramas, entre otros.

$$BLEU = PB \cdot \exp\left(\sum_{n=1}^{N} w_n \log P_n\right) \tag{5.13}$$

Es importante mencionar que, si la cantidad de palabras en la entrada es menor al número de n-gramas a evaluar el valor de BLEU es automáticamente cero.

Tabla 5.1: Puntuación promedio para cada categoría BLEU.

BLEU-1	**BLEU-2**	**BLEU-3**	**BLEU-4**
0.8839	0.8371	0.7976	0.6113

5.1.2. METEOR

Metric for Evaluation of Translation with Explicit ORdering) al igual que BLEU fue originalmente diseñada para evaluar la traducción automática. Esta métrica se basa en la media armónica de precisión y recuperación de unigramas alcanzando una puntuación promedio de ***0.8805***. La Tabla 5.2 muestra algunos ejemplos de la puntuación obtenida entre una oración candidata contra la oración de referencia.

Tabla 5.2: Ejemplos de preguntas y sus puntuaciones METEOR obtenidas.

Referencia	**Candidata**	**METEOR**
¿Cuál es la sede de nvidía corporation?	¿Dónde es la sede de nvidía corporation?	0.8552
¿Qué nacionalidad tuvo alan turing?	¿Qué nacionalidad es alan turing?	0.75
¿Cuáles son las extensiones comunes de r?	¿Cuáles son las extensiones comunes de?	0.7217

5.1.3. ROUGE

Recall-Oriented Understudy for Gisting Evaluation comúnmente conocida como ROUGE fue originalmente diseñada para la evaluación de traducción automática y resumen automático.

Para esta investigación se utilizaron ROUGE-N (1 y 2) y ROUGE-L que pueden considerarse como la granularidad de los textos que se comparan entre la salida del sistema y los elementos de referencia. Por ejemplo, ROUGE-1 y ROUGE-2 se refieren a la superposición de unigramas y bigramas, respectivamente, entre la salida del sistema y la de referencia. A diferencia de ROUGE-L que mide la secuencia de palabras más larga usando LCS (*Longest Common Subsequence*).

$$precision = \frac{number\_of\_overlapping\_words}{total\_words\_in\_system\_output} \tag{5.14}$$

$$recall = \frac{number\_of\_overlapping\_words}{total\_words\_in\_reference\_input} \tag{5.15}$$

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{5.16}$$

La Tabla 5.3 muestra el promedio obtenido por cada categoría ROUGE, donde P (Ecuación 5.14), R (Ecuación 5.15) y F (Ecuación 5.16) corresponden a la precisión, cobertura y *F-Measure* respectivamente.

5.1.4. WER

Finalmente, *Word Error Rate* más conocida como WER es una métrica originalmente diseñada para evaluar el rendimiento de un sistema de reconocimiento de voz o traducción automática. WER (Ecuación 5.17) se deriva de la distancia de Levenshtein, trabajando en el nivel de palabra en lugar del nivel de fonema.

Tabla 5.3: Puntuación promedio de precisión, cobertura y *F-Measure* para cada categoría ROUGE.

	Puntuación promedio		
	Precision	**Recall**	**F-Measure**
Rouge-1	0.9101	0.8903	0.8985
Rouge-2	0.8569	0.8391	0.8461
Rouge-L	0.9120	0.8911	0.8999

$$WER = \frac{(S + D + I)}{N} = \frac{(S + D + I)}{(S + D + C)} \tag{5.17}$$

Donde: S, N, I, D y C representan el número de substituciones , total de palabras de referencia, inserciones, eliminaciones y palabras correctas en la salida, respectivamente. En la Tabla 5.4 se muestran algunos ejemplos de preguntas candidatas y el valor WER obtenido con respecto a la pregunta de referencia.

Tabla 5.4: Ejemplos de preguntas y sus puntuaciones WER obtenidas.

Referencia	**Candidata**	**C**	**S**	**D**	**I**	**WER**
¿Cuándo nació el fundador de apple?	¿Cuándo nació el fundador de apple?	6	0	0	0	0
¿Qué es la minería de datos?	¿Qué es minería de datos?	5	0	1	0	1
¿Quién es el desarrollador de mariadb?	¿Quién es Maria el desarrollador de db?	5	1	0	1	1

La Tabla 5.5 muestra los valores promedio de palabras correctas, substituciones, eliminaciones, inserciones y del índice de WER en el corpus.

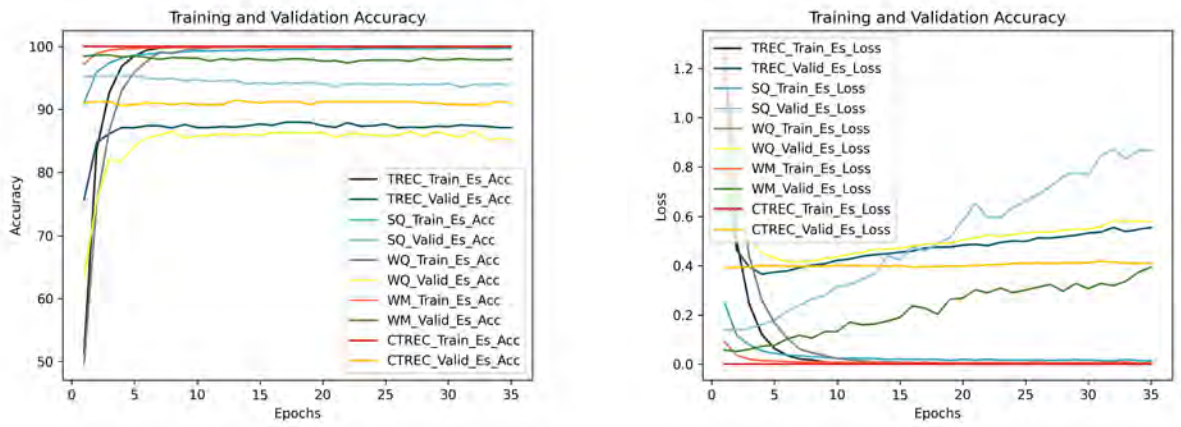
Tabla 5.5: Promedio de palabras para cada rubro de WER.

AVG C	**AVG S**	**AVG D**	**AVG I**	**AVG WER**
9	3	3	2	3

5.2. Resultados de Clasificación de preguntas

Tomando en cuenta la ausencia de información en idioma español para realizar el entrenamiento de la red neuronal convolucional, se tradujeron cada uno de los datasets mencionados con ayuda de herramientas en línea.

Para la evaluación se entrenó la CNN de forma individual para cada dataset correspondiente durante 35 épocas. La Figura 5.1 muestra los valores de Accuracy y Loss en cada época por cada dataset en los conjuntos de entrenamiento y validación.



(a) Accuracy (b) Loss

Figura 5.1: Accuracy y Loss para cada dataset

En la Figura 5.1 (a) se muestran el accuracy correspondiente. En esta se aprecia que el conjunto con mayor valor fue Curated TREC en el conjunto de entrenamiento con un 100 % a lo largo de las 35 épocas, lo que podría suponer un sobreajuste en las Red Neuronal. Por otra parte, el conjunto de validación de WebQuestions tuvo un accuracy por encima del 80 %.

En 5.1 (b) se muestra el loss para cada conjunto de entrenamiento y validación según el dataset correspondiente en las 35 épocas. Aquí es posible apreciar que el conjunto con menor pérdida fue Curated TREC en entrenamiento, lo que podría respaldar que el modelo se haya sobre ajustado. Por otra parte, el conjunto de validación de SimpleQuestions fue aquel con mayor pérdida.

A continuación, se muestra un conjunto de Figuras (5.2-5.6) que corresponden a las matrices de confusión para cada dataset respectivamente en el conjunto de prueba, esta herramienta permite visualizar el desempeño de la CNN implementada. Cada columna de la matriz representa el porcentaje de predicciones de cada clase, mientras que cada fila representa a las instancias en la clase real. Esta herramienta permite visualizar si el sistema está confundiendo n clases.

La Figura 5.2 muestra la matriz de confusión correspondiente para el dataset Curated TREC. Cada clase es representada por el número correspondiente: {'NUM': 0, 'UBI': 1, 'PER': 2, 'ENT': 3, 'DESC': 4, 'ABR': 5}. Aquí es posible apreciar que las clases 0, 1 y 2 son aquellas con mejores resultados de clasificación. Sin embargo, las clases 3, 4 y 5 presentan una deficiencia en la clasificación, esto principalmente por las similitudes sintácticas entre cada tipo de pregunta.

Para el dataset SimpleQuestions V2 se utilizó la siguiente notación: {'ENT': 0, 'PER': 1, 'UBI': 2, 'DESC': 3, 'NUM': 4}. Omitiendo la clase 5 por la ausencia de elementos. A diferencia de la figura anterior, la Figura 5.3 muestra resultados más uniformes en la clasificación de las clases 0, 1, 2 y 3. Sin embargo, el clasificador presenta un error significativo en la clase 4.

La matriz de confusión mostrada en la Figura 5.4 corresponde al dataset TREC 10. En ella se utilizó la siguiente notación: {'DESC': 0, 'ENT': 1, 'PER': 2, 'NUM': 3, 'UBI': 4, 'ABR': 5}. Para este dataset, las clases 2, 3 y 4 presentaron un menor índice de error. Sin embargo, la clase 5 tuvo un rendimiento claramente cuestionable, esto se asocia a la poca cantidad de datos de entrenamiento.

Por otra parte, la Figura 5.5 muestra los resultados correspondientes al dataset WebQues-

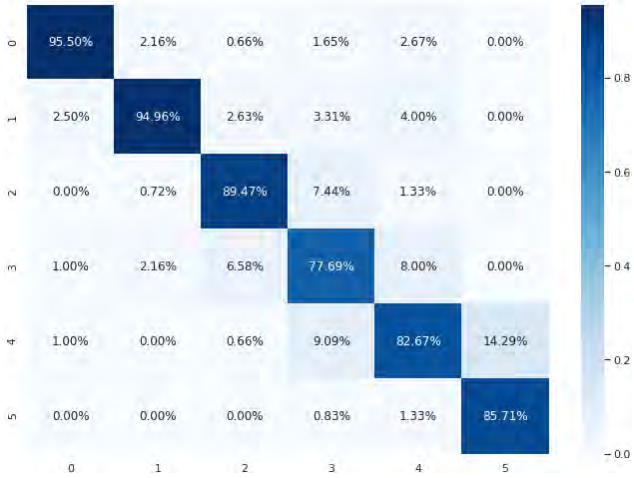


Figura 5.2: Matriz de confusión Curated TREC

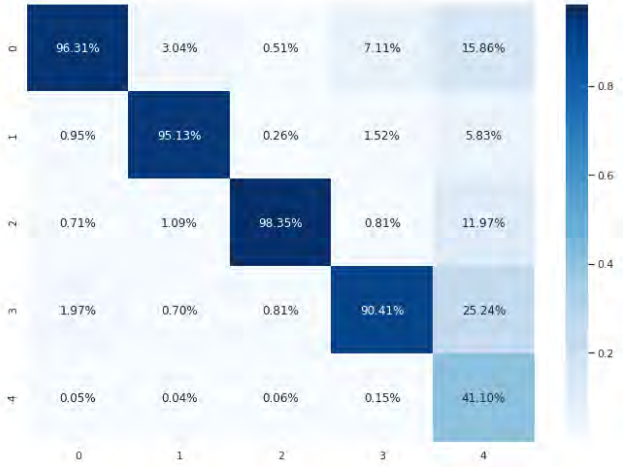


Figura 5.3: Matriz de confusión SimpleQuestions

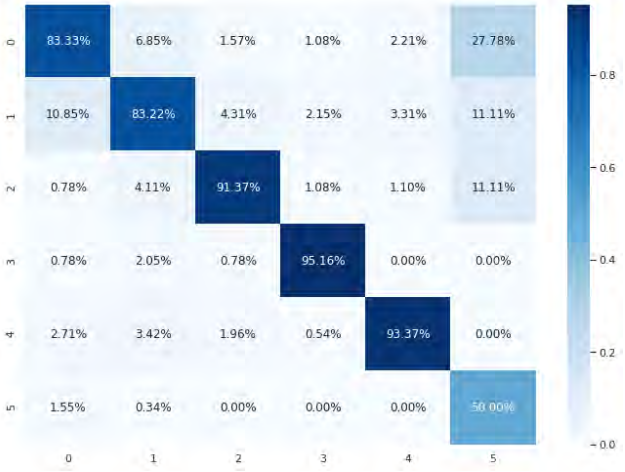


Figura 5.4: Matriz de confusión TREC 10

tions, y, al igual que con SimpleQuestions se omite la clase 5 por la misma razón. Utilizando la notación: {'ENT': 0, 'PER': 1, 'UBI': 2, 'DESC': 3, 'NUM': 4} se aprecia claramente un balance entre las clases. Sin embargo, la clase 3 muestra una mayor porcentaje de error.

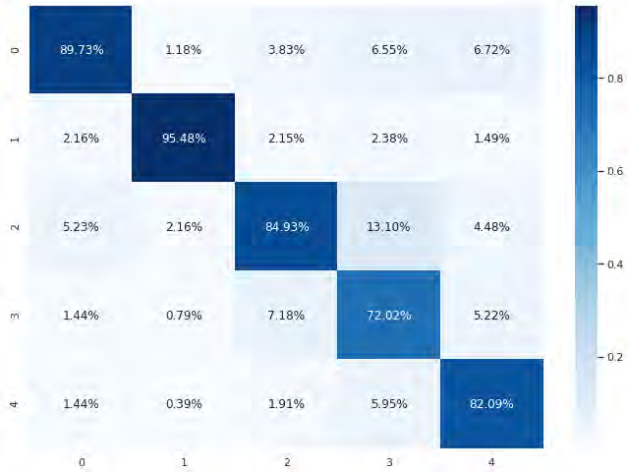


Figura 5.5: Matriz de confusión WebQuestions

Finalmente, La Figura 5.6 muestra la matriz de confusión correspondiente al dataset WikiMovies con la notación: {'PER': 0, 'ENT': 1, 'NUM': 2, 'DESC': 3, 'UBI': 4}. Obteniendo buenos resultados a excepción de la clase 4.

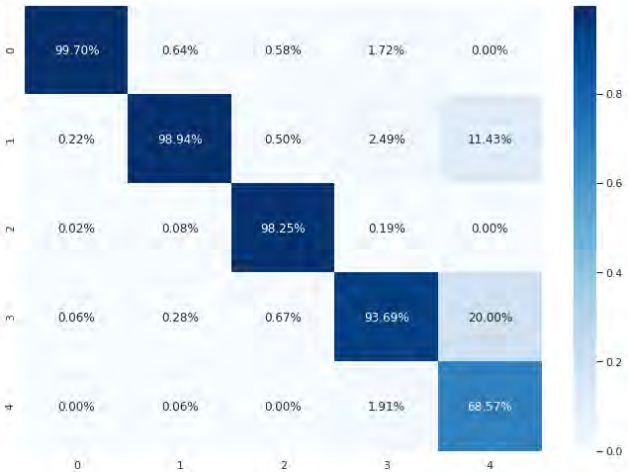


Figura 5.6: Matriz de confusión WikiMovies

A continuación, la Tabla 5.6 muestra los valores de accuracy y loss para cada dataset respectivamente, en el conjunto de validación. En esta se aprecia que Curated TREC logró el mayor porcentaje con un 98.82 %, así como el menor loss con 0.044. Por el contrario, SimpleQuestions tuvo la menor accuracy con 87.80 % y WebQuestions con el mayor loss con 0.347.

Tomando en cuenta que las preguntas reformuladas en 4.3 puede contener errores que interfieran en la identificación del tipo de pregunta, se realizó la clasificación manual y la predicción de la clase correspondiente utilizando la CNN. En la Figura 5.7 se muestra la matriz de confusión correspondiente.

Tabla 5.6: Accuracy y loss en el conjunto de validación por cada dataset.

	CTREC	WQ	WM	TREC	SQ
Accuracy	98.82 %	87.98 %	94.96 %	88.94 %	87.80 %
Loss	0.04	0.34	0.14	0.30	0.32

Figura 5.7: Matriz de confusión de la reformulación de preguntas

Aquí es posible apreciar que las clases 2 y 3 son aquellas con menor precisión en la clasificación. Esto se debe a la similitud sintáctica que tienen ambas clases de preguntas.

5.3. Resultados de generación de respuestas

Para evaluar este módulo se realizó un experimento al que se sometieron 250 preguntas de forma aleatoria. Estas preguntas fueron tomadas del corpus generado en 4.3.1. Cada pregunta fue sometida al proceso ya descrito; para este experimento se definió en 10 el máximo de documentos recuperados, el doble de acuerdo con literatura sobre Recuperación de Información. Todo aquel documento relevante en los primeros 5 documentos fueron considerados *verdadero positivo (tp)*, caso contrario se consideró *falso positivo (fp)*. Por otra parte, aquellos documentos en el rango de 6 a 10 que contengan información relevante fueron considerados *falsos negativos (fn)*, caso contrario *verdaderos negativos (tn)*. La Tabla 5.7 muestra los resultados promedio obtenidos con las métricas de precisión, cobertura y exactitud.

Tabla 5.7: Resultados de la recuperación de documentos relacionados

Precisión	Cobertura	Exactitud
57.67 %	75.37 %	67.46 %

Tomando en cuenta la mejora a los resultados en la etapa de recuperación de documentos relacionados, se continuó con la evaluación de la recuperación de respuestas. Para este experimento se utilizaron las mismas 250 preguntas de la etapa anterior.

Es importante mencionar que esta actividad aún no muestra resultados finales debido a

que el dataset SQuAD 1.1 aún se encuentra en el proceso de corrección de errores, lo que repercute en el entrenamiento del modelo.

El modelo recibe como entrada la pregunta expresada en lenguaje natural. La pregunta y el conjunto de documentos relacionados es codificado por un Red Neuronal Recurrente LSTM Bidireccional.

Para este experimento el modelo recupera cinco respuestas relacionadas a la pregunta original, éstas pueden ser de un mismo artículo o bien de distintos. Es importante mencionar que, una o más respuestas pueden pertenecer el mismo documento pero estar ubicadas en distintos párrafos y sin embargo ambas pueden resultar ser correctas. Por ejemplo, La Tabla 5.8 muestra algunas de las columnas del archivo csv de salida. Por ejemplo, la pregunta *¿qué es ia?* obtuvo las respuestas que se muestran.

Tabla 5.8: Formato

#	**Answer**	**Title**	**AID**	**An-Score**	**Ar-Score**
1	Inteligencia artificial	Inteligencia artificial	1503	101.01	130,88
2	inteligencia artificial	Historia de la inteligencia artificial	1503	6,0709	126,94
3	inteligencia artificial	Inteligencia artificial	1248	1.8404	130,88
4	Archivos y almacén de arte de China	Ai Weiwei	4290342	1,5609	145,41
5	Un rayo de esperanza	Ai Weiwei	4290342	1,5491	145,41

Tres de ellas en los primeros tres lugares y con valores correctos; de éstas, dos pertenecen al mismo documento pero de distinto párrafo y la respuesta restante es de un documento distinto. Los otros dos resultados en las posiciones 4 y 5 son respuestas incorrectas.

Tomando en cuenta esto, la tarea de recuperación de respuestas es, en cierta forma, distinto a la recuperación de información. Por lo tanto, se utilizó MRR.

Mean Reciprocal Rank (MRR) es una medida estadística para evaluar cualquier proceso que produce una lista de posibles respuestas a una muestra de consultas ordenadas por probabilidad de corrección. El rango recíproco de una respuesta a una consulta es el inverso multiplicativo del rango de la primera respuesta correcta: 1 para el primer lugar, 1/2 para el segundo, 1/3 para el tercero y así sucesivamente. El MRR es el promedio de los rangos recíprocos de resultados para una muestra de consultas Q tal y como se muestra en la ecuación 5.18:

$$MRR = \frac{1}{n}\sum_{i=1}^{n} RR_i \tag{5.18}$$

A modo de ejemplo, se han recuperado tres respuestas de tres preguntas distintas. Donde 1 representa una respuesta correcta y 0 una incorrecta:

$$\begin{array}{cccc} 0 & 0 & 1 & = 1/3 \\ 0 & 1 & 0 & = 1/2 \\ 1 & 0 & 0 & = 1 \end{array}$$

El resultado de MRR es calculado como: $\frac{(1/3 + 1/2 + 1)}{3} = 0.61$.

Por lo tanto, para este experimento se analizaron las 150 preguntas con 5 respuestas cada una, un total de 750 respuestas por umbral. La Figura 5.8 muestra el MRR obtenido para cada subconjunto y umbral respectivamente.

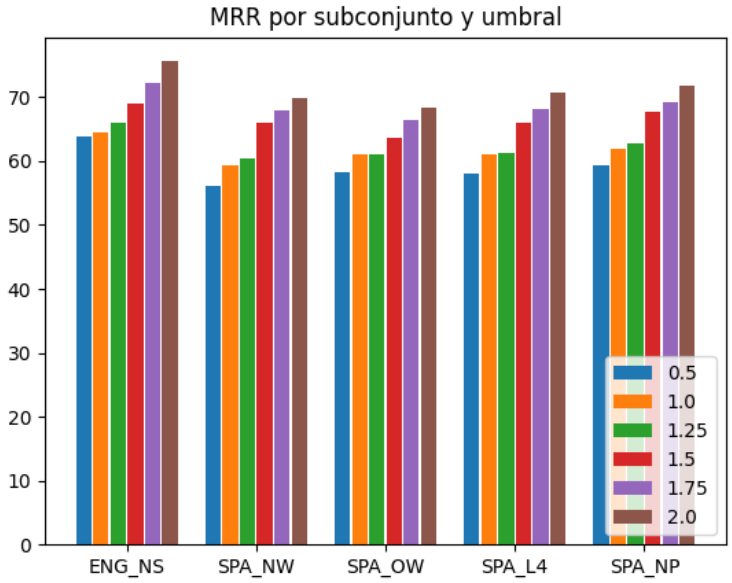


Figura 5.8: MRR por cada subconjunto y umbral

Al igual que en la recuperación de documentos relacionados, los subconjuntos ENG_NS y SPA_OW fueron aquellos con mejores y peores resultados respectivamente tanto en *precision*, *F-Score* y MRR. La Tabla 5.9 muestra los resultados correspondientes.

Tabla 5.9: *Precision*, *F-Score* y MRR por cada subconjunto y umbral

	ENG_NS			SPA_NW			SPA_OW			SPA_L4			SPA_NP		
	MRR	AVG P	AVG F	MRR	AVG P	AVG F	MRR	AVG P	AVG F	MRR	AVG P	AVG F	MRR	AVG P	AVG F
0.5	63.78	42.51	59.65	56.19	34.1	50.85	58.3	35.79	52.71	58.07	35.85	52.77	59.24	37.02	54.03
1	64.6	44.56	61.64	59.26	38.1	55.17	60.97	38.82	55.92	61.15	39.95	57.09	61.93	40.73	57.88
1.25	66	46.78	63.74	60.39	40.37	57.51	61	40.43	57.58	61.2	40.81	57.96	62.78	42.39	59.54
1.5	69.11	50.42	67.03	65.91	46.66	63.63	63.55	43.17	60.30	65.98	46.63	63.60	67.7	48.35	65.18
1.75	72.2	55.07	71.02	68.01	49.68	66.38	66.4	47.03	63.97	68.12	49.65	66.35	69.2	50.73	67.31
2	**75.63**	**59.44**	**74.56**	**69.9**	**52.52**	**68.86**	**68.36**	**50.44**	**67.05**	**70.7**	**53.62**	**69.80**	**71.91**	**54.83**	**70.82**

5.4. Discusión de los resultados

En esta sección se presentaron los distintos modelos desarrollados para resolver la tarea de pregunta-respuesta, así como la construcción y mejorado del corpus y base de conocimiento. La base de toda arquitectura basada en aprendizaje profundo fue la construcción de un corpus, para el cual se utilizó la herramienta WikiExtractor. Sin embargo, dicha herramienta permite descargar todo el compilado de la enciclopedia en línea, por lo cual, los primeros resultados sobre el corpus inicial arrojaba mucha información que no estaba relacionada con el tema de interés de esta investigación. Búscando la mejora continua, se desarrolló un algoritmo que

permitió determinar si un artículo pertenecía o no al tema de interés. Este algoritmo se centró en la generación de una matriz de valores en función de distintos subconjuntos de palabras clave obtenidas de fuentes varias. Con esta implementación se logró una mejora congruente para la recuperación de documentos relacionados.

La segunda parte de esta investigación se centró en la reformulación de preguntar múltiples, es decir, desarrollar un algoritmo capaz de descomponer una cadena de texto con dos o más preguntas en ésta. Este algoritmo logró determinar y capturar fragmentos clave de la cadena original y generar nuevas preguntas individuales concizas y entendibles.

La tercera parte tuvo como objetivo la clasificación de las preguntas utilizando una Red Neuronal Convolucional. Esta red logró resultados prometedores en la clasificación de preguntas utilizando la información obtenido por los *Word Embeddings* así como complementarla con información a nivel de tokens de palabras, partes del discurso e información sintáctica. La salida de este módulo permitió acotar el contexto sobre el cual se buscaría la respuesta correcta dentro del conjunto de respuestas candidatas. Por ejemplo, al lograr clasificar una preguntar como ubicación, el siguiente módulo se centrará en mayor medida a aquellos pasajes que contengan entidades nombradas clasificadas como ubicaciones.

La cuarta y última parte de esta investigación fue el módulo de recuperación de respuestas. Este módulo tuvo una mejora significativa en función de la mejora en la recuperación de documentos, debido a eso se logró reducir el número de artículos no relacionados sobre los cuales se extraían pasajes candidatos. Por ejemplo, uno de los errores que se observaron en unos inicios era el peso que palabras como Java o Python tenían en la recuperación de documentos relacionados, llegando a generar respuestas alejadas del área de interés de esta investigación.

Sin embargo, el área de mejora de esta investigación aún es bastante extensa. Comenzando por la mejora individual de cada uno de los modelos desarrollados. Por ejemplo, la reformulación de preguntas utiliza un enfoque basado en reglas, que si bien logró resultados buenos, este carece de flexibilidad al enfrentar patrones no definidos.

Capítulo 6

Conclusiones

En este capítulo se redactan las conclusiones de esta investigación previo al proceso de evaluación, considerando los resultados obtenido y los objetivos definidos en un inicio. De igual forma se detallan los productos generados y las principales aportaciones. Finalmente, se plaman los trabajos futuros para esta investigación.

6.1. Objetivos y alcances logrados

Las actividades desarrolladas en relación a los objetivos y alcances definidos para el desarrollo de esta investigación se presentan en la Tabla 6.1.

Tabla 6.1: Objetivos realizados

Objetivo	**Actividad**
Investigar y analizar los desafíos específico asociados con la comprensión y generación de respuestas en español	
Implementar un modelo de red neuronal convolucional para clasificar preguntas en español.	Red Neuronal Convolucional (4.4).
Implementar un modelo de red neuronal recurrente para extraer la respuesta correcta.	Red Neuronal Recurrente Bi-LSTM (4.5).
Investigar técnicas avanzadas de Procesamiento de Lenguaje Natural para la interpretación de las preguntas.	Módulo de reformulación de preguntas múltiples (4.3).
Realizar experimentos utilizando conjuntos de datos adaptados al español para entrenar y evaluar el rendimiento del sistema de preguntas y respuestas propuesto.	Módulo de reformulación de preguntas múltiples (4.3).
Realizar experimentos utilizando conjuntos adaptados al español para entrenar y evaluar el rendimiento del sistema de preguntas y respuestas propuesto.	Generación de corpus con Wikipedia y Wikiextractor (4.2.1), experimentación y resultados (5).
Explorar la utilización de técnicas de preentrenamiento de modelos de lenguaje en el contexto de sistemas de preguntas y respuestas en español, con el objetivo de mejorar la precisión y la capacidad de respuesta.	Implementación de Word Embedding 4.2.3
Evaluar el rendimiento de los modelos desarrollados utilizando métricas estándar de evaluación, como la precisión, la cobertura y la coherencia de las respuestas generadas, entre otras.	Experimentación y resultados con métricas BLEU, METEOR, ROUGE, WER, entre otras (5).
Contribuir al avance del campo de procesamiento del lenguaje natural en español, al proporcionar un modelo de preguntas y respuestas eficaz que pueda utilizarse para la búsqueda y recuperación de información en lenguaje natural.	Modelo para question-answering al español, dataset adaptado al español, contribución y divulgación científica.

6.2. Resultados de la investigación

6.2.1. Productos y aportaciones

Durante el desarrollo de esta investigación se obtuvieron los siguientes productos:

1. Reporte de estado del arte: documento en el que se describen un compendio de más de 100 artículos sobre la tarea de pregunta-respuesta, en el cual se describen brevemente

cada arquitectura y métricas de evaluación.

2. Algoritmo de reformulación de preguntas múltiples.

3. Algoritmo de clasificación de artículos de enciclopedias digitales según el subconjunto de palabras clave dado.

4. Modelo de Red Neuronal Convolucional para clasificación de preguntas en español.

5. Modelo de Red Neuronal Recurrente Bi-LSTM para la generación de respuestas.

6. Recursos digitales para el desarrollo de futuras investigaciones.

7. ***Semiautomatic metadata extraction from unstructured documents using Natural Language Processing and Typographic text properties***, artículo publicado en el XI Congreso Mexicano de Inteligencia Artificial - COMIA 2019. En este trabajo se presenta un método para la extracción semiautomático de metadatos en documentos digitales de arte no estructurados. Este tipo de documentos se caracteriza por no contar con orden definido en la distribución de los metadatos, lo cual dificulta la extracción de los mismos. Por lo tanto, para dar solución a este problema se implementan técnicas de PLN y propiedades tipográficas de los textos. Lo que permitió realizar pruebas de las herramientas analizadas en y así crear el módulo de pre-procesamiento.

8. **Heurística para la generación de preguntas y respuestas a partir de la estructura HTML**. Publicados en la Jornada de Ciencia y Tecnología Aplicada Vol. 2, julio - diciembre 2019. El primero de éstos utiliza la estructura HTML de enciclopedias en línea para generar preguntas de forma automática, debido a que en el ambito educativo la mayoría de las preguntas son generadas de forma manual. Este trabajo permitió explorar las estructuras tabulares HTML de los *infobox* de Wikipedia para la generación de preguntas.

9. **Comparación de plataformas de desarrollo de agentes conversacionales**. Publicado en la Jornada de Ciencia y Tecnología Aplicada Vol. 2, julio - diciembre 2019. El segundo artículo presenta una comparación entre las plataformas Watson Assistant perteneciente a IBM y Dialogflow desarrollado por Google, para determinar cuál de éstas tiene un mejor funcionamiento en conversaciones donde los errores ortográficos y las peticiones no previstas están involucradas. Este trabajo permitió identificar las fortalezas y debilidades de los chatbots para la tarea de QA.

10. **Tratamiento automático de preguntas complejas para un sistema *Question-Answering***, artículo publicado en el XII Congreso Mexicano de Inteligencia Artificial - COMIA 2020. En esta investigación se presenta un método de preprocesamiento para un sistema QA, capaz de determinar si una pregunta es múltiple o sencilla y en caso de ser del primer tipo, el método reformula la consulta original en las n preguntas individuales según hayan sido identificadas.

11. **Detección de paráfrasis basada en la energía, entropía y temperatura textual**, publicado en Revista RISTI - Revista Ibérica de Sistemas e Tecnologias de Informação Ingenierías Universidad de Medellín, categoría **Q4** de **Scopus/SCImago Journal Rank**, con **ISSN 1646-9895**. En este trabajo se presenta un método para la detección de paráfrasis incorporando los conceptos de Entropía y Temperatura Textual a un

modelo previo que centró su contribución en la implementación de las redes neuronales recurrentes de Hopfield para generar una medida de distancia llamada Energía Textual. Utilizando la Entropía y la Temperatura se generó un Contexto de Afinidad Libre, basándose en el Model Ising, lo que permitió medir la distribución semántica entre pares de oraciones. Este modelo fue evaluado utilizando el recurso Microsoft Research Paraphrase Corpus, permitiendo superar los resultados del modelo anterior y logrando identificar más de la mitad de la paráfrasis de la muestra analizada. Esta investigación permitió explorar otra arquitectura de Redes Neuronales Artificiales en el área de Procesamiento de Lenguaje Natural, así como la codificación y entrenamiento de la misma.

12. ***Rule-based Spanish Multiple Question Reformulation and their Classification using a Convolutional Neural Network***, artículo publicado a Computación y Sistemas, categoría **Q3** de **Scopus/SCImago Journal Rank ISSN 2007-9737**, **doi: 10.13053/CyS-25-1-3895**. correspondiente. En sta investigación se presenta un método para la reformulación de preguntas múltiples en español, como parte de la etapa de pre-procesamiento en un sistema QA. Utilizando la categoría léxica de cada palabra, Entidades Nombradas y Terminos Multi-Palabra fue posible reformular preguntas múltiples en nuevas preguntas individuales, para posteriormente utilizar una Red Neuronal Convolucional que clasifica cada una de éstas, permitiendo mejorar la calidad de las respuestas, siendo una tarea fundamental en el QA.

6.2.2. Conclusiones

Esta investigación tuvo como objetivo desarrollar los modelos de redes neuronales artificiales tanto recurrentes como convolucionales para dar solución a la tarea de pregunta-respuesta en español. Sin embargo, para lograr cumplir los objetivos establecidos, fue necesario desarrollar una serie de algoritmos que inicialmente no estaban contemplados. Por ejemplo, el módulo de reformulación de preguntas y el algoritmo para la clasificación de artículos de Wikipedia previamente extraídos con WikiExtractor.

Además de esto, se lograron otras aportaciones para posibles investigaciones a futuro, como son la creación de recursos digitales, entre los cuales destacan pero no se limitan a: los modelos neuronales para la clasificación de preguntas y extracción de respuestas, los *Word Embeddings*, el corpus de SQuAD 1.1 adaptado al idioma español, la base de conocimiento, entre otros.

Durante el proceso de evaluación de resultados se implementaron las métricas más utilizadas en la literatura. Por ejemplo, para la reformulación de preguntas las métricas BLEU, METEOR, ROUGE y WER. Las cuales, pese a ser independientes una de la otra lograron resultados similares, debido a que se centran en el análisis de una respuesta candidata contra la respuesta esperada. Por parte de la clasificación de preguntas la métrica que nos permitió evaluar el desempeño del modelo fue *accuracy*. Finalmente, en la generación de respuestas además de las métricas de precisión, cobertura y exactitud, seleccionamos una cuarta métrica creada especialmente para evaluar listas de posibles respuestas a una consulta determinada.

Sin embargo, el área de oportunidad en los sitemas pregunta-respuesta aún es bastante grande. Esta investigación puede verse robustecida desde la creación de un corpus y bases de conocimiento de mayores dimensiones hasta la implementación de arquitecturas emergentes como es el caso de los *Transformers*.

6.2.3. Aportaciones científicas

Con la elaboración de esta investigación las principales aportaciones científicas obtenidas son las siguientes:

- Red Neuronal Convolucional para la clasificación de preguntas.
- Red Neuronal Recurrente Bi-LSTM para la generación de respuestas.
- Recursos digitales para el desarrollo de futuras investigaciones para distintas tareas del PLN.
- Algoritmo de reestructuración de preguntas múltiples.
- Algoritmo de clasificación de artículos de Wikipedia a un dominio específico.

6.2.4. Trabajos futuros

- Mejora para la reformulación de preguntas múltiples.
- Análisis e implementación de una posible mejora para la reformulación de preguntas con el paradigma de aprendizaje profundo.
- Extensión de la base de conocimiento.
- Robustecimiento de los *Word Embeddings*.
- Implementación de arquitecturas tipo *Transformers*.

Referencias

[1] Beysolow, T., Applied Natural Language Processing with Python: Implementing Machine Learning and Deep Learning Algorithms for Natural Language Processing. Apress, 2018, https://books.google.com.mx/books?id=Ujy2wwEACAAJ.

[2] Kochmar, E., Getting Started with Natural Language Processing. Manning, 2022, libgen.li/file.php?md5=a57d1b996409ce91b69840ee787f4712.

[3] Weizenbaum, J., "ELIZA-A computer program for the study of natural language communication between man and machine," Commun. ACM, vol. 9, pp. 36–45, 1966, doi:10.1145/365153.365168.

[4] Mielke, S. J., "Language diversity in ACL 2004 - 2016," 2016, https://sjmielke.com/acl-language-diversity.htm.

[5] Kodra, L. y Meçe, E. K., "Question answering systems: A review on present developments, challenges and trends," 2017.

[6] "Ethnologue : languages of the world," 2023.

[7] Alarcón, R., Sánchez, O., y Mijangos, V., "Exploiting wikipedia as a knowledge base : Towards and ontology of movies," 2009.

[8] Fausett, L. y Fausett, L., Fundamentals of Neural Networks: Architectures, Algorithms, and Applications. Prentice-Hall international editions, Prentice-Hall, 1994, https://books.google.com.mx/books?id=ONylQgAACAAJ.

[9] Goodfellow, I., Bengio, Y., y Courville, A., Deep Learning. Adaptive Computation and Machine Learning series, MIT Press, 2016, https://books.google.com.mx/books?id=Np9SDQAAQBAJ.

[10] Sewak, M., Karim, M. R., y Pujari, P., Practical Convolutional Neural Networks: Implement Advanced Deep Learning Models Using Python. Packt Publishing, 2018.

[11] Kim, Y., "Convolutional Neural Networks for Sentence Classification," Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014, doi:10.3115/v1/d14-1181.

[12] Rumelhart, D. E., Hinton, G. E., y Williams, R. J., "Learning representations by back-propagating errors," Nature, vol. 323, pp. 533–536, 1986.

[13] Elman, J. L., "Finding Structure in Time," Cognitive Science, vol. 14, pp. 179–211, 1990.

[14] Werbos, P. J., "Generalization of backpropagation with application to a recurrent gas market model.," Neural Networks, vol. 1, no. 4, pp. 339–356, 1988, http://dblp.uni-trier.de/db/journals/nn/nn1.html#Werbos88.

[15] Hochreiter, S. y Schmidhuber, J., "Long Short-Term Memory," Neural Comput., vol. 9, no. 8, pp. 1735–1780, 1997, doi:10.1162/neco.1997.9.8.1735.

[16] Bahdanau, D., Cho, K., y Bengio, Y., "Neural machine translation by jointly learning to align and translate," 2014.

[17] Luong, T., Pham, H., y Manning, C. D., "Effective approaches to attention-based neural machine translation," Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, 2015, doi:10.18653/v1/d15-1166.

[18] Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A. C., Salakhutdinov, R., Zemel, R. S., y Bengio, Y., "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention," CoRR, vol. abs/1502.03044, 2015, http://arxiv.org/abs/1502.03044.

[19] Ganegedara, T., Natural Language Processing with TensorFlow: Teach Language to Machines Using Python's Deep Learning Library. Packt Publishing, 2018.

[20] Mikolov, T., Chen, K., Corrado, G., y Dean, J., "Efficient estimation of word representations in vector space," Proceedings of Workshop at ICLR, vol. 2013, 2013.

[21] Bojanowski, P., Grave, E., Joulin, A., y Mikolov, T., "Enriching word vectors with subword information," Transactions of the Association for Computational Linguistics, vol. 5, pp. 135–146, 2017.

[22] Green, Jr., B. F., Wolf, A. K., Chomsky, C., y Laughery, K., "Baseball: An automatic question-answerer," en Papers Presented at the May 9-11, 1961, Western Joint IRE-AIEE-ACM Computer Conference, IRE-AIEE-ACM '61 (Western), (New York, NY, USA), pp. 219–224, ACM, 1961, doi:10.1145/1460690.1460714.

[23] Androutsopoulos, I., Ritchie, G., y Thanisch, P., "Masque/sql– an efficient and portable natural language query interface for relational databases," 1994.

[24] Harabagiu, S., Moldovan, D., Pasca, M., Mihalcea, R., Surdeanu, M., Bunescu, R., Girju, R., Rus, V., y Morarescu, P., "Falcon: Boosting knowledge for answer engines," pp. 479–488, 2000.

[25] Zheng, Z., "Answerbus question answering system," en Proceedings of the Second International Conference on Human Language Technology Research, HLT '02, (San Francisco, CA, USA), pp. 399–404, Morgan Kaufmann Publishers Inc., 2002, http://dl.acm.org/citation.cfm?id=1289189.1289238.

[26] Brill, E., Dumais, S., y Banko, M., "An analysis of the askmsr question-answering system," en Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10, EMNLP '02, (Stroudsburg, PA, USA), pp. 257–264, Association for Computational Linguistics, 2002, doi:10.3115/1118693.1118726.

[27] Popescu, A.-M., Etzioni, O., y Kautz, H., "Towards a theory of natural language interfaces to databases," en Proceedings of the 8th International Conference on Intelligent User Interfaces, IUI '03, (New York, NY, USA), pp. 149–157, ACM, 2003, doi:10.1145/604045.604070.

[28] Parthasarathy, S. y Chen, J., "A web-based question answering system for effective e-learning," en Seventh IEEE International Conference on Advanced Learning Technologies (ICALT 2007), pp. 142–146, 2007, doi:10.1109/ICALT.2007.42.

[29] Unger, C. y Cimiano, P., "Pythia: Compositional meaning construction for ontology-

based question answering on the semantic web," en Natural Language Processing and Information Systems (Muñoz, R., Montoyo, A., y Métais, E., eds.), (Berlin, Heidelberg), pp. 153–160, Springer Berlin Heidelberg, 2011.

[30] Dong, T., Furbach, U., Glöckner, I., y Pelzer, B., "A natural language question answering system as a participant in human Q&A portals," en Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Three, IJCAI'11, pp. 2430–2435, AAAI Press, 2011, doi:10.5591/978-1-57735-516-8/IJCAI11-405.

[31] He, S., Zhang, Y., Liu, K., y Zhao, J., "Casia@v2: A mln-based question answering system over linked data," en CLEF, 2014.

[32] Dhingra, B., Danish, D., y Rajagopal, D., "Simple and effective semi-supervised question answering," Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers), 2018, doi:10.18653/v1/n18-2092.

[33] Kim, M. y Kim, H., "Design of question answering system with automated question generation," en 2008 Fourth International Conference on Networked Computing and Advanced Information Management, vol. 2, pp. 365–368, 2008, doi:10.1109/NCM.2008.236.

[34] Hu, D., Wang, W., Xie, N., y Cao, C., "Acqa_onto: An ontology approach for restrain domain question answering system," en IET International Conference on Information Science and Control Engineering 2012 (ICISCE 2012), pp. 1–5, 2012, doi:10.1049/cp.2012.2341.

[35] Moghaddam, S. y Ester, M., "Aqa: Aspect-based opinion question answering," en 2011 IEEE 11th International Conference on Data Mining Workshops, pp. 89–96, 2011, doi:10.1109/ICDMW.2011.34.

[36] Fikri, A. y Purwarianti, A., "Case based indonesian closed domain question answering system with real world questions," en 2012 7th International Conference on Telecommunication Systems, Services, and Applications (TSSA), pp. 181–186, 2012, doi:10.1109/TSSA.2012.6366047.

[37] Wei, Z., Xuan, Z., y Junjie, C., "Design and implementation of influenza question answering system based on multi-strategies," en 2012 IEEE International Conference on Computer Science and Automation Engineering (CSAE), vol. 1, pp. 720–723, 2012, doi:10.1109/CSAE.2012.6272693.

[38] Perera, R., "Ipedagogy: Question answering system based on web information clustering," en 2012 IEEE Fourth International Conference on Technology for Education, pp. 245–246, 2012, doi:10.1109/T4E.2012.48.

[39] Lu, W., Cheng, J., y Yang, Q., "Question answering system based on web," en 2012 Fifth International Conference on Intelligent Computation Technology and Automation, pp. 573–576, 2012, doi:10.1109/ICICTA.2012.150.

[40] Yamada, T. y Arakawa, T., "A study on output sentence generation method for question answering using statistical machine translation," en 2013 13th International Conference on Control, Automation and Systems (ICCAS 2013), pp. 1199–1202, 2013, doi:10.1109/ICCAS.2013.6704146.

[41] Xia, T., Chai, Y., Lu, H., y Wang, T., "E-learning support system aided by vsm based question answering system," en 2013 8th International Conference on Computer Science Education, pp. 1281–1285, 2013, doi:10.1109/ICCSE.2013.6554118.

[42] Liu, Z., Wang, X., Chen, Q., Zhang, Y., y Xiang, Y., "A chinese question answering system based on web search," en 2014 International Conference on Machine Learning and Cybernetics, vol. 2, pp. 816–820, 2014, doi:10.1109/ICMLC.2014.7009714.

[43] Te, R., "Research on question classification method of tibetan online automatic question-answering system," en 2011 4th International Conference on Intelligent Networks and Intelligent Systems, pp. 211–213, 2011, doi:10.1109/ICINIS.2011.42.

[44] Biswas, P., Sharan, A., y Malik, N., "A framework for restricted domain question answering system," en 2014 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT), pp. 613–620, 2014, doi:10.1109/ICICICT.2014.6781351.

[45] Huang, X., Zhang, Y., Wei, B., y Yao, L., "A question-answering system over traditional chinese medicine," en 2015 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), pp. 1737–1739, 2015, doi:10.1109/BIBM.2015.7359945.

[46] Nguyen, H. V. y Nguyen, D. T., "Application of first-order logic inference in vietnamese question answering system," en 2015 6th International Conference on Intelligent Systems, Modelling and Simulation, pp. 10–15, 2015, doi:10.1109/ISMS.2015.14.

[47] Chalabi, H. M. A., Ray, S. K., y Shaalan, K., "Question classification for arabic question answering systems," en 2015 International Conference on Information and Communication Technology Research (ICTRC), pp. 310–313, 2015, doi:10.1109/ICTRC.2015.7156484.

[48] Bayoudhi, A., Belguith, L. H., y Ghorbel, H., "Question focus extraction and answer passage retrieval," en 2014 IEEE/ACS 11th International Conference on Computer Systems and Applications (AICCSA), pp. 658–665, 2014, doi:10.1109/AICCSA.2014.7073262.

[49] Xie, X., Song, W., Liu, L., Du, C., y Wang, H., "Research and implementation of automatic question answering system based on ontology," en The 27th Chinese Control and Decision Conference (2015 CCDC), pp. 1366–1370, 2015, doi:10.1109/CCDC.2015.7162131.

[50] Pudaruth, S., Boodhoo, K., y Goolbudun, L., "An intelligent question answering system for ict," en 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT), pp. 2895–2899, 2016, doi:10.1109/ICEEOT.2016.7755228.

[51] Huang, X., Wei, B., y Zhang, Y., "Automatic question-answering based on wikipedia data extraction," en 2015 10th International Conference on Intelligent Systems and Knowledge Engineering (ISKE), pp. 314–317, 2015, doi:10.1109/ISKE.2015.78.

[52] Tayyar Madabushi, H. y Lee, M., "High accuracy rule-based question classification using question syntax and semantics," en Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, (Osaka, Japan), pp. 1220–1230, The COLING 2016 Organizing Committee, 2016, https://www.aclweb.org/anthology/C16-1116.

[53] Tayyar Madabushi, H., Lee, M., y Barnden, J., "Integrating question classification and

deep learning for improved answer selection," en Proceedings of the 27th International Conference on Computational Linguistics, (Santa Fe, New Mexico, USA), pp. 3283–3294, Association for Computational Linguistics, 2018, https://www.aclweb.org/anthology/C18-1278.

[54] Mohd, M. y Hashmy, R., "Question classification using a knowledge-based semantic kernel," en Soft Computing: Theories and Applications (Pant, M., Ray, K., Sharma, T. K., Rawat, S., y Bandyopadhyay, A., eds.), (Singapore), pp. 599–606, Springer Singapore, 2018.

[55] Mohasseb, A., Bader-El-Den, M., y Cocea, M., "Question categorization and classification using grammar based approach," Information Processing & Management, vol. 54, no. 6, pp. 1228 – 1243, 2018, doi:https://doi.org/10.1016/j.ipm.2018.05.001.

[56] Xu, S., Cheng, G., y Kong, F., "Research on question classification for automatic question answering," en 2016 International Conference on Asian Language Processing (IALP), pp. 218–221, 2016, doi:10.1109/IALP.2016.7875972.

[57] Devi, M. y Dua, M., "Adans: An agriculture domain question answering system using ontologies," en 2017 International Conference on Computing, Communication and Automation (ICCCA), pp. 122–127, 2017, doi:10.1109/CCAA.2017.8229784.

[58] Jayalakshmi, S. y Sheshasaayee, A., "Automated question answering system using ontology and semantic role," en 2017 International Conference on Innovative Mechanisms for Industry Applications (ICIMIA), pp. 528–532, 2017, doi:10.1109/ICIMIA.2017.7975515.

[59] Pathak, S. y Mishra, N., "Context aware restricted tourism domain question answering system," en 2016 2nd International Conference on Next Generation Computing Technologies (NGCT), pp. 534–539, 2016, doi:10.1109/NGCT.2016.7877473.

[60] Garg, S. y Kumar, S., "Josn: Java oriented question-answering system combining semantic web and natural language processing techniques," en 2016 1st India International Conference on Information Processing (IICIP), pp. 1–6, 2016, doi:10.1109/IICIP.2016.7975361.

[61] Ranjan, P. y Balabantaray, R. C., "Question answering system for factoid based question," en 2016 2nd International Conference on Contemporary Computing and Informatics (IC3I), pp. 221–224, 2016, doi:10.1109/IC3I.2016.7917964.

[62] Kahaduwa, H., Pathirana, D., Arachchi, P. L., Dias, V., Ranathunga, S., y Kohomban, U., "Question answering system for the travel domain," en 2017 Moratuwa Engineering Research Conference (MERCon), pp. 449–454, 2017, doi:10.1109/MERCon.2017.7980526.

[63] Menaha, R., Surya, A. U., Nandhni, K., y Ishwarya, M., "Question answering system using web snippets," en 2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), pp. 387–390, 2017, doi:10.1109/I-SMAC.2017.8058377.

[64] Dönmez, I. y Adali, E., "Turkish question answering application with course-grained semantic matrix representation of sentences," en 2017 International Conference on Computer Science and Engineering (UBMK), pp. 6–11, 2017, doi:10.1109/UBMK.2017.8093464.

[65] Samadi, A., Hanaa, E. F., Qbadou, M., Youssfi, M., y Akef, F., "A syntactic and semantic multi-agent based question answering system for collaborative e-learning," en 2018 4th International Conference on Optimization and Applications (ICOA), pp. 1–4, 2018, doi:10.1109/ICOA.2018.8370588.

[66] Guerrieri, A., Ghiani, G., y Manni, A., "A tourist advisor based on a question answering system," en 2017 Intelligent Systems Conference (IntelliSys), pp. 1173–1176, 2017, doi:10.1109/IntelliSys.2017.8324280.

[67] Philip, B. A., Jog, M., y Upasani, A. M., "Dr. tux: A question answering system for ubuntu users," 2018.

[68] Heie, M. H., Whittaker, E. W., y Furui, S., "Question answering using statistical language modelling," Computer Speech & Language, vol. 26, no. 3, pp. 193 – 209, 2012, doi:https://doi.org/10.1016/j.csl.2011.11.001.

[69] Fukumoto, J., Aburai, N., y Yamanishi, R., "Interactive document expansion for answer extraction of question answering system," Procedia Computer Science, vol. 22, pp. 991 – 1000, 2013, doi:https://doi.org/10.1016/j.procs.2013.09.184. 17th International Conference in Knowledge Based and Intelligent Information and Engineering Systems - KES2013.

[70] Ryu, P.-M., Jang, M.-G., y Kim, H.-K., "Open domain question answering using wikipedia-based knowledge model," Information Processing & Management, vol. 50, no. 5, pp. 683 – 692, 2014, doi:https://doi.org/10.1016/j.ipm.2014.04.007.

[71] Yang, M.-C., Lee, D.-G., Park, S.-Y., y Rim, H.-C., "Knowledge-based question answering using the semantic embedding space," Expert Systems with Applications, vol. 42, no. 23, pp. 9086 – 9104, 2015, doi:https://doi.org/10.1016/j.eswa.2015.07.009.

[72] Neves, M. y Leser, U., "Question answering for biology," Methods, vol. 74, pp. 36 – 46, 2015, doi:https://doi.org/10.1016/j.ymeth.2014.10.023. Text mining of biomedical literature.

[73] Shekarpour, S., Marx, E., Ngomo, A.-C. N., y Auer, S., "Sina: Semantic interpretation of user queries for question answering on interlinked data," Journal of Web Semantics, vol. 30, pp. 39 – 51, 2015, doi:https://doi.org/10.1016/j.websem.2014.06.002. Semantic Search.

[74] Jovita, Linda, Hartawan, A., y Suhartono, D., "Using vector space model in question answering system," Procedia Computer Science, vol. 59, pp. 305 – 311, 2015, doi:https://doi.org/10.1016/j.procs.2015.07.570. International Conference on Computer Science and Computational Intelligence (ICCSCI 2015).

[75] Seena, I., Sini, G., y Binu, R., "Malayalam question answering system," Procedia Technology, vol. 24, pp. 1388 – 1392, 2016, doi:https://doi.org/10.1016/j.protcy.2016.05.155. International Conference on Emerging Trends in Engineering, Science and Technology (ICETEST - 2015).

[76] Sarrouti, M. y Alaoui, S. O. E., "A passage retrieval method based on probabilistic information retrieval model and umls concepts in biomedical question answering," Journal of Biomedical Informatics, vol. 68, pp. 96 – 103, 2017, doi:https://doi.org/10.1016/j.jbi.2017.03.001.

[77] Albarghothi, A., Khater, F., y Shaalan, K., "Arabic question answering using ontology,"

Procedia Computer Science, vol. 117, pp. 183 – 191, 2017, doi:https://doi.org/10.1016/j.procs.2017.10.108. Arabic Computational Linguistics.

[78] Elalfy, D., Gad, W., y Ismail, R., "A hybrid model to predict best answers in question answering communities," Egyptian Informatics Journal, vol. 19, no. 1, pp. 21 – 31, 2018, doi:https://doi.org/10.1016/j.eij.2017.06.002.

[79] Gunawan, A. A. S., Mulyono, P. R., y Budiharto, W., "Indonesian question answering system for solving arithmetic word problems on intelligent humanoid robot," Procedia Computer Science, vol. 135, pp. 719 – 726, 2018, doi:https://doi.org/10.1016/j.procs.2018.08.213. The 3rd International Conference on Computer Science and Computational Intelligence (ICCSCI 2018) : Empowering Smart Technology in Digital Era for a Better Life.

[80] Rani, M., Muyeba, M. K., y Vyas, O. P., "A hybrid approach using ontology similarity and fuzzy logic for semantic question answering," Advanced Computing, Networking and Informatics- Volume 1, p. 601–609, 2014, doi:10.1007/978-3-319-07353-8_69.

[81] Wiese, G., Weissenborn, D., y Neves, M., "Neural domain adaptation for biomedical question answering," Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017), 2017, doi:10.18653/v1/k17-1029.

[82] Yang, Y., Yu, J., Hu, Y., Xu, X., y Nyberg, E., "Cmu livemedqa at trec 2017 liveqa: A consumer health question answering system," 2017.

[83] Dong, L., Wei, F., Zhou, M., y Xu, K., "Question answering over Freebase with multi-column convolutional neural networks," en Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), (Beijing, China), pp. 260–269, Association for Computational Linguistics, 2015, doi:10.3115/v1/P15-1026.

[84] Juárez-González, A., Téllez-Valero, A., Denicia-Carral, C., Montes-y Gómez, M., y Villaseñor-Pineda, L., "Using machine learning and text mining in question answering," en Evaluation of Multilingual and Multi-modal Information Retrieval (Peters, C., Clough, P., Gey, F. C., Karlgren, J., Magnini, B., Oard, D. W., de Rijke, M., y Stempfhuber, M., eds.), (Berlin, Heidelberg), pp. 415–423, Springer Berlin Heidelberg, 2007.

[85] A. Ferrucci, D., Brown, E., Chu-Carroll, J., Fan, J., Gondek, D., Kalyanpur, A., Lally, A., Murdock, J. W., Nyberg, E., Prager, J., Schlaefer, N., y Welty, C., "Building watson: An overview of the deepqa project," AI Magazine, vol. 31, pp. 59–79, 2010.

[86] Cao, Y., Liu, F., Simpson, P., Antieau, L., Bennett, A., Cimino, J. J., Ely, J., y Yu, H., "Askhermes: An online question answering system for complex clinical questions," Journal of Biomedical Informatics, vol. 44, no. 2, pp. 277 – 288, 2011, doi:https://doi.org/10.1016/j.jbi.2011.01.004.

[87] Madotto, A. y Attardi, G., "Question dependent recurrent entity network for question answering," 2017.

[88] Wang, T., Yuan, X., y Trischler, A., "A joint model for question answering and question generation," 2017.

[89] Qu, Y., Liu, J., Kang, L., Shi, Q., y Ye, D., "Question answering over freebase via attentive rnn with similarity matrix based cnn," 2018.

[90] Xiong, C., Zhong, V., y Socher, R., "Dynamic coattention networks for question answering," 2016.

[91] Htut, P. M., Bowman, S. R., y Cho, K., "Training a ranking function for open-domain question answering," 2018.

[92] Hong, D., "Attention-based recurrent neural networks for question answering," 2017.

[93] Zhang, K. y Zhao, J., "A chinese question-answering system with question classification and answer clustering," en 2010 Seventh International Conference on Fuzzy Systems and Knowledge Discovery, vol. 6, pp. 2692–2696, 2010, doi:10.1109/FSKD.2010.5569607.

[94] Sherkat, E. y Farhoodi, M., "A hybrid approach for question classification in persian automatic question answering systems," en 2014 4th International Conference on Computer and Knowledge Engineering (ICCKE), pp. 279–284, 2014, doi:10.1109/ICCKE.2014.6993377.

[95] Gallagher, S. y Zadrozny, W., "Leveraging large corpora using internet search for question answering," en 2016 IEEE/WIC/ACM International Conference on Web Intelligence (WI), pp. 532–535, 2016, doi:10.1109/WI.2016.0090.

[96] Dodiya, T. y Jain, S., "Question classification for medical domain question answering system," en 2016 IEEE International WIE Conference on Electrical and Computer Engineering (WIECON-ECE), pp. 204–207, 2016, doi:10.1109/WIECON-ECE.2016.8009118.

[97] Ansari, A., Maknojia, M., y Shaikh, A., "Intelligent question answering system based on artificial neural network," en 2016 IEEE International Conference on Engineering and Technology (ICETECH), pp. 758–763, 2016, doi:10.1109/ICETECH.2016.7569350.

[98] Chen, L., Zeng, G., Zhang, Q., Chen, X., y Wu, D., "Question answering over knowledgebase with attention-based lstm networks and knowledge embeddings," en 2017 IEEE 16th International Conference on Cognitive Informatics Cognitive Computing (ICCI*CC), pp. 243–246, 2017, doi:10.1109/ICCI-CC.2017.8109757.

[99] Minaee, S. y Liu, Z., "Automatic question-answering using a deep similarity neural network," 2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP), 2017, doi:10.1109/globalsip.2017.8309095.

[100] Ma, R., Zhang, J., Li, M., Chen, L., y Gao, J., "Hybrid answer selection model for non-factoid question answering," en 2017 International Conference on Asian Language Processing (IALP), pp. 371–373, 2017, doi:10.1109/IALP.2017.8300620.

[101] Zhong, W., Tang, D., Duan, N., Zhou, M., Wang, J., y Yin, J., "Improving question answering by commonsense-based pre-training," 2018.

[102] Joty, S., Marquez, L., y Nakov, P., "Joint multitask learning for community question answering using task-specific embeddings," 2018.

[103] Ni, J., Zhu, C., Chen, W., y McAuley, J., "Learning to attend on essential terms: An enhanced retriever-reader model for open-domain question answering," 2018.

[104] Sun, H., Dhingra, B., Zaheer, M., Mazaitis, K., Salakhutdinov, R., y Cohen, W. W., "Open domain question answering using early fusion of knowledge bases and text," 2018.

[105] Cao, N. D., Aziz, W., y Titov, I., "Question answering by reasoning across documents with graph convolutional networks," 2018.

[106] Yue, C., Cao, H., Xiong, K., Cui, A., Qin, H., y Li, M., "Enhanced question understanding with dynamic memory networks for textual question answering," Expert Systems with Applications, vol. 80, pp. 39 – 45, 2017, doi:https://doi.org/10.1016/j.eswa.2017.03.006.

[107] Das, R., Zaheer, M., Reddy, S., y McCallum, A., "Question answering on knowledge bases and text using universal schema and memory networks," Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), 2017, doi:10.18653/v1/p17-2057.

[108] Yu, M., Yin, W., Hasan, K. S., dos Santos, C., Xiang, B., y Zhou, B., "Improved neural relation detection for knowledge base question answering," Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2017, doi:10.18653/v1/p17-1053.

[109] Watanabe, Y., Dhingra, B., y Salakhutdinov, R., "Question answering from unstructured text by retrieval and comprehension," 2017.

[110] Do, P.-K., Nguyen, H.-T., Tran, C.-X., Nguyen, M.-T., y Nguyen, M.-L., "Legal question answering using ranking svm and deep convolutional neural network," 2017.

[111] Wang, S., Yu, M., Guo, X., Wang, Z., Klinger, T., Zhang, W., Chang, S., Tesauro, G., Zhou, B., y Jiang, J., "R^3: Reinforced reader-ranker for open-domain question answering," 2017.

[112] Li, Y., Su, L., Chen, J., y Yuan, L., "Semi-supervised learning for question classification in cqa," Natural Computing: An International Journal, vol. 16, no. 4, p. 567–577, 2017, doi:10.1007/s11047-016-9554-5.

[113] Tay, Y., Tuan, L. A., y Hui, S. C., "Hyperbolic representation learning for fast and efficient neural question answering," Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining - WSDM '18, 2018, doi:10.1145/3159652.3159664.

[114] Ma, F., Chitta, R., Kataria, S., Zhou, J., Ramesh, P., Sun, T., y Gao, J., "Long-term memory networks for question answering," 2017.

[115] Sequiera, R., Baruah, G., Tu, Z., Mohammed, S., Rao, J., Zhang, H., y Lin, J., "Exploring the effectiveness of convolutional neural networks for answer selection in end-to-end question answering," 2017.

[116] Palangi, H., Smolensky, P., He, X., y Deng, L., "Question-answering with grammatically-interpretable representations," 2017.

[117] Vakulenko, S. y Savenkov, V., "Tableqa: Question answering on tabular data," 2017.

[118] Xiong, C., Zhong, V., y Socher, R., "Dcn+: Mixed objective and deep residual coattention for question answering," 2017.

[119] Wasim, M., Mahmood, W., Asim, M. N., y Khan, M. U., "Multi-label question classification for factoid and list type questions in biomedical question answering," IEEE Access, vol. 7, pp. 3882–3896, 2019, doi:10.1109/ACCESS.2018.2887165.

[120] Hu, S., Du, Y., Luo, X., Kong, D., y Li, Q., "A hybrid bi-attention mechanisms and

long short-term memory model for chinese question classification," en 2019 2nd International Conference on Safety Produce Informatization (IICSPI), pp. 609–612, 2019, doi:10.1109/IICSPI48186.2019.9096024.

[121] Liu, J., Yang, Y., Lv, S., Wang, J., y Chen, H., "Attention-based BiGRU-CNN for chinese question classification," Journal of Ambient Intelligence and Humanized Computing, 2019, doi:10.1007/s12652-019-01344-9.

[122] Xu, D., Jansen, P., Martin, J., Xie, Z., Yadav, V., Tayyar Madabushi, H., Tafjord, O., y Clark, P., "Multi-class hierarchical question classification for multiple choice science exams," en Proceedings of the 12th Language Resources and Evaluation Conference, (Marseille, France), pp. 5370–5382, European Language Resources Association, 2020, https://www.aclweb.org/anthology/2020.lrec-1.661.

[123] Li, X. y Roth, D., "Learning question classifiers," en Proceedings of the 19th International Conference on Computational Linguistics - Volume 1, COLING '02, (USA), p. 1–7, Association for Computational Linguistics, 2002, doi:10.3115/1072228.1072378.

[124] "Natural language toolkit — nltk 3.5 documentation." https://www.nltk.org/index.html. (Accessed on 01/18/2021).

[125] "Working with text data — scikit-learn 0.24.0 documentation." https://scikit-learn.org/stable/tutorial/text_analytics/working_with_text_data.html. (Accessed on 01/18/2021).

[126] Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S. J., y McClosky, D., "The Stanford CoreNLP natural language processing toolkit," en Association for Computational Linguistics (ACL) System Demonstrations, pp. 55–60, 2014, http://www.aclweb.org/anthology/P/P14/P14-5010.

[127] Padró, L. y Stanilovsky, E., "Freeling 3.0: Towards wider multilinguality," en Proceedings of the Language Resources and Evaluation Conference (LREC 2012), (Istanbul, Turkey), ELRA, 2012.

[128] "spacy · industrial-strength natural language processing in python." https://spacy.io/. (Accessed on 01/18/2021).

[129] Berant, J., Chou, A., Frostig, R., y Liang, P., "Semantic parsing on Freebase from question-answer pairs," en Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, (Seattle, Washington, USA), pp. 1533–1544, Association for Computational Linguistics, 2013, https://www.aclweb.org/anthology/D13-1160.

[130] Bordes, A., Usunier, N., Chopra, S., y Weston, J., "Large-scale simple question answering with memory networks," CoRR, vol. abs/1506.02075, 2015, http://arxiv.org/abs/1506.02075.

[131] Serban, I. V., García-Durán, A., Gulcehre, C., Ahn, S., Chandar, S., Courville, A., y Bengio, Y., "Generating factoid questions with recurrent neural networks: The 30M factoid question-answer corpus," en Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), (Berlin, Germany), pp. 588–598, Association for Computational Linguistics, 2016, doi:10.18653/v1/P16-1056.

[132] Miller, A., Fisch, A., Dodge, J., Karimi, A.-H., Bordes, A., y Weston, J., "Key-value memory networks for directly reading documents," en Proceedings of the 2016 Conference

on Empirical Methods in Natural Language Processing, (Austin, Texas), pp. 1400–1409, Association for Computational Linguistics, 2016, doi:10.18653/v1/D16-1147.

[133] Dodge, J., Gane, A., Zhang, X., Bordes, A., Chopra, S., Miller, A. H., Szlam, A., y Weston, J., "Evaluating prerequisite qualities for learning end-to-end dialog systems," CoRR, vol. abs/1511.06931, 2016.

[134] Su, Y., Sun, H., Sadler, B., Srivatsa, M., Gür, I., Yan, Z., y Yan, X., "On generating characteristic-rich question sets for QA evaluation," en Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, (Austin, Texas), pp. 562–572, Association for Computational Linguistics, 2016, doi:10.18653/v1/D16-1054.

[135] Joshi, M., Choi, E., Weld, D., y Zettlemoyer, L., "TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension," en Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), (Vancouver, Canada), pp. 1601–1611, Association for Computational Linguistics, 2017, doi:10.18653/v1/P17-1147.

[136] Rajpurkar, P., Zhang, J., Lopyrev, K., y Liang, P., "Squad: 100, 000+ questions for machine comprehension of text," en EMNLP, 2016.

[137] Rajpurkar, P., Jia, R., y Liang, P., "Know what you don't know: Unanswerable questions for squad," en ACL, 2018.

[138] Jiang, K., Wu, D., y Jiang, H., "FreebaseQA: A new factoid QA data set matching trivia-style question-answer pairs with Freebase," en Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), (Minneapolis, Minnesota), pp. 318–323, Association for Computational Linguistics, 2019, doi:10.18653/v1/N19-1028.

[139] Abujabal, A., Saha Roy, R., Yahya, M., y Weikum, G., "ComQA: A community-sourced dataset for complex factoid question answering with paraphrase clusters," en Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), (Minneapolis, Minnesota), pp. 307–317, Association for Computational Linguistics, 2019, doi:10.18653/v1/N19-1027.

[140] Trivedi, P., Maheshwari, G., Dubey, M., y Lehmann, J., "Lc-quad: A corpus for complex question answering over knowledge graphs," en The Semantic Web – ISWC 2017 (d'Amato, C., Fernandez, M., Tamma, V., Lecue, F., Cudré-Mauroux, P., Sequeda, J., Lange, C., y Heflin, J., eds.), (Cham), pp. 210–218, Springer International Publishing, 2017.

[141] Talmor, A. y Berant, J., "The web as a knowledge-base for answering complex questions," en Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), (New Orleans, Louisiana), pp. 641–651, Association for Computational Linguistics, 2018, doi:10.18653/v1/N18-1059.

[142] Nguyen, T., Rosenberg, M., Song, X., Gao, J., Tiwary, S., Majumder, R., y Deng, L., "Ms marco: A human generated machine reading comprehension dataset," 2016, https://www.microsoft.com/en-us/research/publication/ms-marco-human-generated-machine-reading-comprehension-dataset/.

[143] Kwiatkowski, T., Palomaki, J., Redfield, O., Collins, M., Parikh, A., Alberti, C., Epstein, D., Polosukhin, I., Kelcey, M., Devlin, J., Lee, K., Toutanova, K. N., Jones, L., Chang, M.-W., Dai, A., Uszkoreit, J., Le, Q., y Petrov, S., "Natural questions: a benchmark for question answering research," Transactions of the Association of Computational Linguistics, 2019.

[144] Keysers, D., Schärli, N., Scales, N., Buisman, H., Furrer, D., Kashubin, S., Momchev, N., Sinopalnikov, D., Stafiniak, L., Tihon, T., Tsarkov, D., Wang, X., van Zee, M., y Bousquet, O., "Measuring compositional generalization: A comprehensive method on realistic data," en ICLR, 2020, https://arxiv.org/pdf/1912.09713.pdf.

[145] Voorhees, E. M., "The trec-8 question answering track report," en In Proceedings of TREC-8, pp. 77–82, 1999.

[146] "Trec 2019 deep learning track guidelines | trec-2019-deep-learning." https://microsoft.github.io/TREC-2019-Deep-Learning/. (Accessed on 02/12/2020).

[147] "Github - attardi/wikiextractor: A tool for extracting plain text from wikipedia dumps." https://github.com/attardi/wikiextractor. (Accessed on 01/27/2020).

[148] Cardellino, C., "Spanish Billion Words Corpus and Embeddings," 2019, https://crscardellino.github.io/SBWCE/.

[149] Cho, K., van Merriënboer, B., Bahdanau, D., y Bengio, Y., "On the properties of neural machine translation: Encoder–decoder approaches," en Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation, (Doha, Qatar), pp. 103–111, Association for Computational Linguistics, 2014, doi:10.3115/v1/W14-4012.

[150] Koehn, P. y Knowles, R., "Six challenges for neural machine translation," en Proceedings of the First Workshop on Neural Machine Translation, (Vancouver), pp. 28–39, Association for Computational Linguistics, 2017, doi:10.18653/v1/W17-3204.

[151] Silva, J., Dias, G., Billot, S., y Lopes, G., "Using localmaxs algorithm for the extraction of contiguous and non-contiguous multiword lexical units," vol. 1695, pp. 113–132, 1999, doi:10.1007/3-540-48159-1_9.

[152] Aggarwal, C. C., Neural Networks and Deep Learning - A Textbook. Springer, 2018, doi:10.1007/978-3-319-94463-0.

[153] Salton, G. y McGill, M. J., Introduction to Modern Information Retrieval. USA: McGraw-Hill, Inc., 1986.

[154] Salton, G. y Buckley, C., "Term-weighting approaches in automatic text retrieval," Inf. Process. Manage., vol. 24, no. 5, p. 513–523, 1988, doi:10.1016/0306-4573(88)90021-0.

[155] Weinberger, K., Dasgupta, A., Langford, J., Smola, A., y Attenberg, J., "Feature hashing for large scale multitask learning," en Proceedings of the 26th Annual International Conference on Machine Learning, ICML '09, (New York, NY, USA), p. 1113–1120, Association for Computing Machinery, 2009, doi:10.1145/1553374.1553516.

[156] Lee, K., Salant, S., Kwiatkowski, T., Parikh, A., Das, D., y Berant, J., "Learning recurrent span representations for extractive question answering," 2016.

[157] Van Rossum, G. y Drake, F. L., Python 3 Reference Manual. Scotts Valley, CA: CreateSpace, 2009.

[158] Django, "Django documentation | django." https://docs.djangoproject.com/en/3.1/, 2020. (Accessed on 01/11/2021).