BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA
FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

Plataforma de apoyo para personas con trastornos de memoria

TESIS
PARA OBTENER EL GRADO DE:
Maestro en Ciencias de la Computación

PRESENTA:
Erick Quezada Mosqueda

DIRECTOR:
Dra. María Josefa Somodevilla García
ASESOR:
Dr. Ivo Humberto Pineda Torres

PUEBLA, PUE., NOVIEMBRE 2020

"Siempre que escucho a la gente decir que la IA perjudicará a las personas en el futuro, creo que, en general, la tecnología siempre se puede usar para bien y para mal y hay que tener cuidado con cómo se construye ... Si estás argumentando en contra de la IA, entonces estás argumentando en contra de automóviles más seguros que no van a tener accidentes, y estás argumentando en contra de poder diagnosticar mejor a las personas cuando están enfermas."

"Whenever I hear people saying AI is going to hurt people in the future I think, yeah, technology can generally always be used for good and bad and you need to be careful about how you build it ... if you're arguing against AI then you're arguing against safer cars that aren't going to have accidents, and you're arguing against being able to better diagnose people when they're sick."

Mark Zuckerberg

Agradecimientos

- A mis padres Consuelo y Alfonso, por su cariño, guía y apoyo en todas mis decisiones.
- A mi hermana Fernanda, por sus consejos y por siempre motivarme a realizar este proyecto.
- A mi novia Valeria, por su apoyo incondicional, por siempre creer en mi y acompañarme con un a sonrisa en este y en todos los procesos de mi vida.
- A mis amigos de "El Gremio", por sus consejos y su lealtad.
- A todas las personas que me motivaron a realizar este proyecto, por creer en mi.
- Al CONACYT, por los recursos económicos otorgados para el estudio de este grado de maestría.
- A mis asesores el Dr. Ivo Pineda y la Dra. María Josefa Somodevilla, por su apoyo constante.
- A la BUAP y a la Facultad de Ciencias de la Computación, por las facilidades proporcionadas durante mi estancia académica.

Índice general

Capítulo 1

Introducción

1.1. Presentación

Existen diversas causas por las cuales una persona puede sufrir de pérdida de memoria, en algunos casos este problema es solo temporal causado por una condición médica y que desaparece en cuanto la persona recibe un tratamiento correcto, sin embargo existen casos en donde esta pérdida de memoria es mucho más grave y además es progresiva. Síndromes como la demencia senil y enfermedades como el Alzheimer afectan con mayor frecuencia a personas mayores de los 65 años y las personas que sufren este tipo de impedimentos no pueden vivir de forma independiente. Con el aumento de la esperanza de vida de la población mundial y siendo que la posibilidad de contraer una enfermedad neurodegenerativa se multiplica con los años, surge la necesidad de desarrollar sistemas que asistan a personas con este tipo de padecimientos. Actualmente existen algunos sistemas cuyo objetivo es ayudar a personas con estas condiciones, sin embargo la mayoría de estos están más enfocados en monitorear las condiciones de salud del paciente y prevenir accidentes en lugar de asistirlos en realizar sus actividades diarias. Es por ello que el objetivo de este trabajo es desarrollar un sistema de apoyo, en la forma de un agente conversacional, para personas con problemas leves de memoria, que interactúe con los usuarios por medio de comandos de voz y los ayude a recordar detalles de eventos pasados, dándoles así un mayor grado de autonomía y facilitando el trabajo de sus cuidadores. Para desarrollar este asistente se utilizaran técnicas de estado del arte en inteligencia artificial y procesamiento de lenguaje natural así como técnicas de cómputo en la nube.

1.2. Motivación

La amnesia es la perdida inusual de recuerdos que pueden ser hechos, información o experiencias pasadas. Usualmente las personas con amnesia si pueden identificarse a sí mismas pero tienen problemas para retener nueva información [28]. Si bien la pérdida

de memoria puede ser transitoria y resolverse después de un corto periodo de tiempo, también puede ser permanente y empeorar progresivamente, siendo en este caso un impedimento para realizar las actividades diarias. El envejecimiento puede causar ciertos problemas de memoria, además de una ligera disminución de otras habilidades de razonamiento y de la capacidad de aprender nueva información, sin embargo estos cambios no deben de ser drásticos e impedir que las personas vivan una vida independiente y productiva.

La mayoría de las personas con amnesia tienen problemas de memoria a corto plazo, es decir, tienen dificultad para retener información nueva, mientras que información de recuerdos más remotos, como recuerdos de la niñez, por lo general queda intacta. La amnesia por sí sola no afecta la inteligencia, razonamiento, personalidad o juicio de la persona afectada, de hecho las personas que la padecen usualmente pueden entender palabras escritas o habladas, aprender nuevas habilidades e incluso es posible que entiendan el hecho de que tienen un problema de pérdida de memoria [28].

La pérdida de memoria puede ser causada por diversos factores, ya que diversas áreas del cerebro se encargan de crear y recuperar recuerdos, y por ello cualquier problema que se presente en alguna de estas áreas puede interferir con la memoria. La amnesia causada por un daño en el cerebro es conocida como amnesia neurológica y entre sus posibles causas están las siguientes [28]:

- Derrame cerebral.
- Encefalitis.
- Falta de oxigenación adecuada en el cerebro.
- Abuso del alcohol.
- Tumores en las áreas del cerebro que controlan la memoria.
- Convulsiones.
- Enfermedades neurodegenerativas como la enfermedad de Alzheimer.

Además de las anteriores, existen otras causas de pérdida de memoria, como es el caso de las lesiones en la cabeza, las cuales si bien en el caso de ser lesiones leves no suelen causar una amnesia duradera, las lesiones fuertes si pueden causar una pérdida de memoria permanente. Otro tipo de amnesia es la conocida como amnesia disociativa, la cual es causada por un shock emocional o un trauma, en este caso las personas pueden perder sus recuerdos personales e incluso información autobiográfica, pero usualmente solo de forma temporal [28].

La pérdida de memoria también puede ser un signo de la demencia, la cual no solo afecta la memoria sino que también trae consigo otros problemas cognitivos como afectaciones al pensamiento, el lenguaje y la conducta de los individuos que la padecen. La Organización Mundial de la Salud (OMS) define la demencia como “un síndrome causado por una enfermedad del cerebro -usualmente de naturaleza crónica o progresiva-

en la cual hay una alteración de múltiples funciones corticales superiores, incluyendo la memoria, el pensamiento, la orientación, la comprensión, el lenguaje, la capacidad de aprender y de realizar cálculos, y la toma de decisiones" [13]. Por ello cabe señalar que la demencia no se refiere a una enfermedad en específico sino a un conjunto de síntomas adquiridos de tipo cognitivo, siendo la enfermedad de Alzheimer es la forma más común de demencia.

Si bien la demencia no solo afecta a personas de la tercera edad, el envejecimiento es uno de los factores de riesgo más grandes, por lo que suele presentarse con mayor frecuencia en dicha población. Algunos otros factores de riesgo para contraer demencia son la hipertensión, diabetes, uso de tabaco y alcohol, obesidad, inactividad física, bajo nivel de estudios, depresión, entre otros. Es importante aclarar que la demencia no es una consecuencia inevitable del envejecimiento.

Según cifras de la Organización Mundial de la Salud [48] se estima que cada años alrededor de 9.9 millones de personas desarrollan demencia, además en el año 2018 se estimaban alrededor de 50 millones de casos de demencia en el mundo, se pronostica que para el 2030 habrán aproximadamente 82 millones y para el 2050 152 millones de casos.

La demencia senil incluye el deterioro de funciones cognitivas siendo la memoria una de las funciones más afectadas. Los pacientes que sufren de demencia senil sufren de una disminución de la capacidad para almacenar nueva información y, al mismo tiempo, puede haber pérdida de las memorias formadas previamente. Los pacientes pueden también presentar dificultades en el reconocimiento de caras o reconocimiento de lugares destacados. Asimismo, otra función cognitiva que afecta al paciente con demencia senil es la dificultad para producir y comprender el lenguaje, lo que pueden impedir una buena interacción y relación con las personas de su entorno.

Algunos de los síntomas de la demencia son dificultad para llevar a cabo tareas cotidianas, confusión en entornos familiares, dificultades con las palabras y los números, pérdida de memoria y cambios de humor y comportamiento. La OMS divide los problemas relacionados con la demencia en tres etapas: fase temprana, media y avanzada [13].

Fase temprana. La persona se torna olvidadiza especialmente con cosas que acaban de suceder, puede tener dificultad para la comunicación encontrando las palabras adecuadas para comunicarse, se pierde en lugares conocidos, puede tener dificultades en la toma de decisiones y manejo de sus finanzas, pierde la noción del tiempo desconociendo la hora, día, mes y año actuales. Esta fase tiende comúnmente a ser ignorada por amigos y familiares del paciente ya que ven estos comportamientos como normales en el proceso de envejecimiento.

Fase media. Los problemas derivados de la demencia se vuelven más obvios y limitantes. La persona se torna muy olvidadiza especialmente en eventos recientes y nombres de personas, puede tener dificultad para comprender el tiempo, fechas, lugares y eventos; podría perderse tanto en el hogar como en la comunidad, puede aumentar su dificultad de comunicación, puede necesitar ayuda con el cuidado personal, no puede preparar comida exitosamente ni ir de compras, y es incapaz de vivir solo de manera

segura.

Fase avanzada. La fase final es casi de total dependencia al cuidador e inactividad de la persona, las alternaciones en la memoria son más severas. Usualmente la persona no está al tanto de la fecha o el lugar en donde se encuentra, tiene dificultad para comprender lo que sucede a su alrededor, no reconoce familiares, amigos u objetos conocidos, puede tener dificultad para comer sin ayuda, tiene una necesidad cada vez mayor de recibir asistencia para el autocuidado, puede sufrir de incontinencia renal e intestinal, puede no ser capaz de caminar o estar confinado a una silla de ruedas o cama, puede intensificar los cambios de comportamiento e incluir agresión hacia su cuidador, puede perderse en el hogar.

Aunque la demencia principalmente afecta a adultos mayores, no es una parte normal del envejecimiento. Se estima que entre el 2 % y el 10 % de todos los casos de demencia comienzan antes de los 65 años y la probabilidad de desarrollarla se duplica con cada intervalo de 5 años después de los 65 años de edad. En promedio las personas viven durante muchos años después de la aparición de los síntomas de la demencia y con un apoyo apropiado, muchos pueden y deben tener la posibilidad de seguir participando y de contribuir en la sociedad, así como de tener una buena calidad de vida.

En mayo del año 2017 en la Septuagésima Asamblea Mundial de la Salud que tuvo lugar en Ginebra, Suiza; se creó un plan de acción global con el objetivo de mejorar las condiciones de vida tanto de las personas que padecen de demencia, como de sus cuidadores y familiares. El plan de acción global contra la demencia 2017-2025 se divide en siete puntos:

- Reconocer la demencia como una prioridad de salud pública.
- Fomentar una cultura de conciencia y amabilidad hacia la demencia.
- Reducir los riesgos de contraer demencia.
- Diagnosticar la demencia y proporcionar tratamiento, atención y apoyo.
- Brindar apoyo y entrenamiento a los cuidadores y familiares de personas con demencia.
- Recolectar datos de importancia para indicadores.
- Duplicar la investigación e innovación relacionada con la demencia.

Las personas que padecen de demencia requieren de muchos cuidados, por lo tanto, es importante considerar el impacto de la condición no solo en las personas afectadas sino también en sus familiares. En la mayoría de los entornos, los cuidadores son los cónyuges, y las mujeres superan en número a los hombres. El cuidado de la demencia es difícil y requiere tiempo, energía y frecuentemente del esfuerzo físico del cuidador. Como la enfermedad progresa de manera lenta, los familiares que proveen los cuidados están sometidos a altos niveles de estrés durante largos periodos de tiempo. Es por ello que, los cuidadores de pacientes con demencia senil suelen ser más propensos a la

depresión que otras personas, siendo las cuidadoras mujeres las que presentan niveles más altos de síntomas de depresión y ansiedad y un nivel menor de satisfacción de vida. El grado de estrés que presentan los cuidadores de personas con demencia es mucho mayor al que presentan otros cuidadores que encargan de personas mayores con otro tipo de enfermedades.

Es importante señalar que el bienestar psicológico de los cuidadores tiene una gran influencia en el estado del paciente con demencia, ya que estos, al ser atendidos por cuidadores estresados, suelen presentar problemas de conducta, estrés y episodios de agitación.

1.3. Objetivos

1.3.1. Objetivo general

Desarrollar una plataforma de apoyo para personas con trastornos leves de memoria, que ayude a los usuarios a recordar eventos pasados, mediante el uso de técnicas de inteligencia artificial y servicios de computación en la nube.

1.3.2. Objetivos específicos

1. Diseñar un agente conversacional (chatbot) enfocado en apoyar a personas con trastornos leves de memoria.
2. Desarrollar un sistema de preguntas-respuestas, adaptado al idioma español, utilizando técnicas de estado del arte en inteligencia artificial.
3. Emplear servicios en la nube para integrar los distintos módulos que componen el asistente de memoria.
4. Elaborar una interfaz de usuario que permita la comunicación con el asistente de memoria de forma intuitiva.

1.4. Estructura de la tesis

Este trabajo está dividido en seis capítulos, siendo el primero este capítulo dedicado a la introducción del proyecto de tesis, donde principalmente se describieron la problemática a tratar, la motivación del proyecto y los objetivos del mismo. El segundo capítulo está compuesto por el marco teórico, ahí se introducirán los principales fundamentos teóricos que son la base de este proyecto de tesis. El correcto entendimiento de los temas presentados en este capítulo son fundamentales para poder comprender en su totalidad la propuesta tecnológica presentada en este trabajo. En esta sección se introducirán diversos conceptos y sistemas computacionales con los que hay que estar

previamente familiarizado, correspondiendo la mayoría de estos a nuevas tecnologías que han emergido en los últimos años como es el caso del aprendizaje automático y la computación en la nube.

El tercer capítulo de este trabajo corresponde al estado del arte, en esta sección se describen a detalle los últimos avances tecnológicos relacionados con las distintas tecnologías utilizadas para la elaboración de este proyecto. Con fines solamente de comparación, en este capítulo se presentarán otras propuestas de sistemas de apoyo para personas con problemas de memoria, los cuales utilizan otros enfoques tecnológicos para asistir a dicho grupo de personas. Además, en esta sección también se hace referencia a diversas herramientas disponibles para la elaboración de agentes conversacionales, las cuales serán utilizadas en este trabajo ya que el sistema de asistencia a elaborar tendrá la forma de un chatbot. Finalmente, dado que una parte importante del proyecto es el uso de nuevas técnicas pertenecientes al campo de la inteligencia artificial, y al ser este un campo en constante desarrollo y crecimiento, esta sección estará compuesta en su mayoría por los diversos avances realizados por grandes empresas tecnológicas en esta área. En específico, se detallarán los avances que combinan técnicas de procesamiento de lenguaje natural con un nuevo enfoque de la inteligencia artificial conocido como aprendizaje profundo, y que servirán para el desarrollo de este proyecto.

El cuarto capítulo de este trabajo de tesis corresponde al desarrollo del proyecto. Esta sección inicia con la metodología de desarrollo, seguida del diseño del proyecto. Estas secciones servirán para obtener una visión general del funcionamiento esperado del asistente de memoria, además de servir como introducción para las secciones siguientes, en donde se explicará a detalle el desarrollo de componentes clave del sistema. El primero de estos componentes es un sistema de preguntas y respuestas (QA), el cual será necesario para obtener el funcionamiento deseado del sistema de asistencia de memoria. El desarrollo de un sistema de QA robusto, y que además funcione en el idioma español, involucró diversas pruebas que serán detalladas en esta sección. Para la elaboración de este proyecto se decidió hacer uso de diversos servicios en la nube, por lo que también se dedica una sección para describir el por que se utilizaron dichos servicios, sus beneficios y la forma en cómo se integran en el proyecto. Finalmente, para finalizar este capítulo se describe la interfaz de usuario que se desarrolló, junto con las elecciones de hardware y software que se consideraron para brindar la mejor experiencia de usuario.

El capítulo 5 de este trabajo se compone del análisis de resultados, en esta sección se presenta el sistema de asistencia finalizado, así como sus funciones y limitaciones. Además, en este capítulo se describen algunas de las formas en las cuales se espera que el usuario interactúe con el sistema de asistencia, así como el funcionamiento del mismo para resolver las solicitudes del usuario. Finalmente, el capítulo 6 de este trabajo de tesis corresponde a la sección de conclusiones, el cual es seguido por la sección de referencias.

Capítulo 2

Marco Teórico

En este capítulo se presentan los principales fundamentos teóricos que engloban este proyecto de tesis. El entendimiento de dichos fundamentos, así como la familiarización con el lenguaje técnico de los mismos, es necesario para la correcta comprensión de los futuros capítulos de este proyecto.

2.1. Procesamiento de Lenguaje Natural

El procesamiento del lenguaje natural (NLP por sus siglas en inglés) es un término que se refiere al procesamiento automático del lenguaje humano por medio de sistemas computacionales. De forma general, esto incluye tanto algoritmos que toman como entrada texto producido por humanos, como algoritmos que producen texto en lenguaje natural [16]. El procesamiento del lenguaje natural es un área de investigación muy amplia ya que, como tal cubre cualquier manipulación computacional del lenguaje natural, lo cual puede ser desde una tarea simple como contar la frecuencia con la que aparecen ciertas palabras en un texto, hasta la realización de tareas mucho más complejas como el entender las solicitudes específicas de un usuario y ofrecerle respuestas útiles [5].

El procesamiento de lenguaje natural es un campo de investigación muy desafiante, ya que las palabras de un texto poseen relaciones no lineares muy complejas entre sí, lo cual hace muy difícil poder modelaras y capturar su información correctamente. Además cada idioma tiene su propia gramática, sintaxis y vocabulario, lo cual hace que tanto los avances tecnológicos, como los recursos disponibles, estén ligados a un idioma específico. El campo del NLP provee tanto la teoría como las herramientas para resolver distintas tareas, a continuación se enlistaran algunas de las más comunes [15]:

- **Tokenización**: Es la tarea de dividir un texto en unidades atómicas, comúnmente palabras pero pueden ser también caracteres o sub-palabras.
- **Reconocimiento de entidades nombradas (Named Entity Recognition o NER)**: Esta tarea consiste en identificar de forma automática las entidades nombradas en un texto y clasificarlas en categorías predefinidas. Algunos ejemplos

de entidades pueden ser nombres de personas, lugares, organizaciones, cantidades, fechas, entre otras.

- **Etiquetado de parte del discurso (Part-of-Speech (PoS) tagging)**: El etiquetado de PoS es la tarea de asignar palabras a sus respectivas partes del habla. Puede ser etiquetas básicas como sustantivo, verbo, adjetivo, adverbio y preposición, o puede ser granular como sustantivo propio, sustantivo común, verbo, etc.
- **Clasificación de oraciones**: La clasificación de oraciones es el proceso de categorizar texto en distintos grupos pre-definidos. Esta tarea tiene muchos casos de uso comunes, como detección de spam, clasificación de artículos de noticias (en por ejemplo, política, tecnología y deportes) y análisis de sentimiento (es decir, indicar si un comentario o reseña es positivo, negativo o neutro).
- **Respuesta a preguntas (Question Answering o QA)**: De forma general, esta tarea consiste en desarrollar sistemas capaces de responder preguntas hechas en lenguaje natural por humanos. Las técnicas de QA son ampliamente usadas comercialmente, y se encuentran en la base de los chatbots y asistente virtuales como Google Assistant y Alexa de Amazon. Los chatbots han sido adoptados por muchas compañías para el soporte al cliente ya que pueden responder las inquietudes directas de los clientes sin la necesidad de intervención humana. Un sistema de QA toca muchos otros aspectos de la NLP, como la recuperación de información y la representación del conocimiento. En consecuencia, todo esto hace que el desarrollo de un sistema de QA sea una tarea compleja.
- **Traducción automática (Machine Translation o MT)**: MT es la tarea de transformar una oración de un idioma de origen a un idioma de destino. Esta es una tarea muy desafiante, ya que diferentes idiomas tienen estructuras morfológicas muy distintas, lo que significa que por lo general no es una transformación de uno a uno. Las relaciones de palabra a palabra entre idiomas pueden ser de uno a muchos, de uno a uno, de muchos a uno o de muchos a muchos, lo cual se conoce como el problema de alineación de palabras en la literatura de MT.

El enfoque tradicional para resolver tareas de NLP implica un conjunto de subtareas distintas. Primero, los cuerpos de texto deben pre-procesarse enfocándose en reducir el vocabulario, quitar signos de puntuación y palabras no útiles. A continuación, vienen varios pasos de ingeniería de características. El objetivo principal de la ingeniería de características es facilitar el aprendizaje de los algoritmos. A menudo las características están diseñadas a mano y están influenciadas por la comprensión humana de un lenguaje. La ingeniería de características es de suma importancia para los algoritmos de NLP clásicos y, en consecuencia, los sistemas con mejor rendimiento a menudo tenían las funciones mejor diseñadas. A continuación, se utiliza un algoritmo computacional el cual aprende a desempeñarse bien en la tarea dada utilizando las características obtenidas. Finalmente, se utiliza el algoritmo previamente entrenado para realizar inferencias o predicciones.

Con el propósito de mejorar los resultados de los modelos NLP, se optó por combinar las herramientas de NLP tradicionales con técnicas de aprendizaje profundo. Los modelos profundos crearon una ola de cambios de paradigma en muchos de los campos del aprendizaje automático. Algunos ejemplos de su uso en el campo del procesamiento de lenguaje natural son el uso de redes neuronales convolucionales (CNN) para la clasificación de oraciones, redes neuronales recurrentes (RNN) para introducir el concepto de "memoria" en las modelos y tener poder percibir e interpretar las secuencias de caracteres. Estos temas se detallaran con mayor profundidad en la sección 2.3.

El año 2018 fue un año marcado con grandes avances en el área del procesamiento de lenguaje natural, ya que se descubrieron nuevas formas de representar las palabras capturando de mejor forma el significado de las mismas haciendo uso del contexto en donde se usan. Esto será revisado con mayor detalle en la sección 3.3.1. Además, junto con estas nuevas formas de representar palabras, se crearon nuevas arquitecturas de aprendizaje profundo las cuales permiten entrenar poderosos modelos con menor cantidad de datos de entrenamiento utilizando como base modelos pre-entrenados y modificándolos ligeramente para realizar tareas NLP específicas (técnica conocida como aprendizaje transferido o "transfer learning" en inglés). Esto será explicado con mayor detalle en la sección 3.3.2.

2.2. Aprendizaje Automático

El aprendizaje automático es una rama de la inteligencia artificial, y de forma general puede definirse como el estudio de algoritmos computacionales que mejoran automáticamente a través de la experiencia. En este sentido, el Dr. Tomas M. Mitchell, profesor de la universidad de Carnegie Mellon; pionero en estudios de inteligencia artificial y aprendizaje automática; y fundador del primer departamento de investigación de aprendizaje automático en el mundo, propone la siguiente definición formal del término "aprender": "Se dice que un programa computacional aprende de la experiencia E con respeto a alguna clase de tareas T y medida de rendimiento P, si su rendimiento en tareas en T, medido por P, mejora con experiencia E" [30].

Los algoritmos de aprendizaje automático utilizan técnicas estadísticas para encontrar patrones en grandes cantidades de datos los cuales pueden ser números, imágenes, archivos de texto o cualquier cosa que pueda ser guardada de forma digital. El enfoque utilizado en el aprendizaje automático difiere del enfoque de programación tradicional debido a que en el segundo se generan programas que pueden realizar una tarea debido a que un programador ha definido previamente todos los parámetros necesarios para realizarla, mientras que en el caso del aprendizaje automático los programas construyen un modelo matemático que representa el problema a resolver y encuentran los parámetros óptimos para resolverlo.

Para entrenar un modelo de aprendizaje automático se necesitan conseguir o generar datos y los modelos se benefician mientras mayor sea la cantidad de estos. De manera general los datos de entrenamiento se dividen en dos partes: conjunto de datos de

entrenamiento y conjunto de datos de prueba, en ocasiones se dividen en tres partes agregando un conjunto extra llamado conjunto de datos de evaluación. El conjunto de datos de entrenamiento es el que utiliza el modelo para aprender las características del conjunto de datos y modificar los parámetros del modelo con el fin de obtener el mejor desempeño posible, el conjunto de prueba es donde se evalúa el desempeño el modelo bajo el criterio de que ese conjunto de datos no ha sido visto previamente por el modelo y por ello se considera una buena representación de como funcionara una vez implementado.

El conjunto de evaluación, que en ocasiones se suele agregar, se utiliza para validar los resultados obtenidos del modelo después de haber sido entrenado y en caso de no ser satisfactorios modificar parámetros del mismo conocidos como hiper-parámetros. Los hiper-parámetros son una serie de valores que son utilizados por el modelo pero que no son obtenidos de los datos, ni son modificados automáticamente por el modelo sino por el programador. Dado que la modificación de hiper-parámetros ocurre con base en los resultados obtenidos durante la evaluación, se considera que esto hace que el modelo este influenciado y por ende, una vez implementado ofrezca un menor desempeño al esperado. Al agregar ese conjunto de datos extra de puede entrenar el modelo, evaluarlo, realizar modificaciones a los hiper-parámetros y repetir hasta que los resultados de la evaluación sean satisfactorios, una vez pase esto se puede hacer una evaluación final del desempeño del modelo con los datos de prueba.

El proceso de generar un modelo de aprendizaje automático puede resumirse de la siguiente forma:

1. Definir el problema a resolver.
2. Conseguir o generar datos requeridos para realizar el entrenamiento. La cantidad y calidad de los datos indicara que tan preciso será el modelo.
3. Pre-procesar los datos de entrada. Se debe asegurar que todos datos de entrada se encuentren en el formato deseado removiendo elementos o caracteres indeseados.
4. Seleccionar un modelo adecuado para la tarea.
5. Entrenar el algoritmo de aprendizaje automático utilizando el conjunto de datos de entrenamiento.
6. Probar el modelo.
7. Evaluar el modelo. En caso de que los resultados obtenidos por el modelo no sean satisfactorios, regresar a 4.
8. Implementar el modelo.

Los modelos de aprendizaje automático suelen dividirse principalmente en tres categorías basados en su forma de aprendizaje:

- Aprendizaje Supervisado

- Aprendizaje No Supervisado
- Aprendizaje por Refuerzos

2.2.1. Aprendizaje Supervisado

Los algoritmos de aprendizaje supervisado son un tipo de algoritmos que utilizan durante el proceso de entrenamiento un conjunto de datos que fue previamente etiquetado, es decir, que cuenta con información tanto de los parámetros de entrada como del valor de salida al que el modelo debe llegar. El propósito de este tipo de entrenamiento es que el modelo logre encontrar una relación entre los parámetros del modelo y su valor de salida, para después tener la capacidad de predecir una respuesta para datos nuevos que no están etiquetados. Los modelos de aprendizaje supervisado intentan modelar relaciones y dependencias entre los valores de entrada y el valor de salida objetivo, para así poder predecir los valores de salida en futuros nuevos datos que no se encuentran etiquetados. En los algoritmos de aprendizaje supervisado el valor de salida puede ser un valor categórico o un continuo, en el primer caso se considera un algoritmo de clasificación y en el segundo un algoritmo de regresión.

2.2.2. Aprendizaje No Supervisado

En este tipo de aprendizaje, el conjunto de datos de entrenamiento no se encuentra etiquetado, es decir no se le provee al modelo con un valor de salida objetivo. En este tipo de algoritmos, durante el proceso de entrenamiento el modelo no busca predecir un valor categórico o continuo a la salida sino que busca encontrar patrones dentro del conjunto de datos de entrada. Un ejemplo de este tipo de algoritmos son los algoritmos de agrupamiento ("clustering" en inglés), en donde se busca formar grupos compuestos de los elementos del conjunto de datos de entrada, que presentan un grado de similitud entre ellos. Este tipo de entrenamiento es útil en caso de tener un conjunto de datos de entrada muy grande del cual no se tiene suficiente información.

2.2.3. Aprendizaje por Refuerzos

El aprendizaje por refuerzos es un enfoque orientado a tareas de la inteligencia artificial en donde un sistema, también llamado "agente", aprende automáticamente a realizar una tarea mediante sus interacciones con el ambiente. Un ejemplo común de este tipo de aprendizaje es el hecho de enseñarle a una maquina a jugar un videojuego. En este caso el maquina jugador es el agente, el juego es el ambiente y la meta es ganar el juego. Para lograr esta tarea el agente adaptara automáticamente sus parámetros basados en las interacciones con su entorno y utilizara dichos parámetros para tomar las mejores decisiones que le ayuden a ganar el juego. En el aprendizaje por refuerzos el agente no solo busca completar la tarea sino que busca la forma más eficiente de realizarla.

2.3. Aprendizaje Profundo

Los algoritmos de aprendizaje automático son útiles para realizar una gran cantidad de tareas, sin embargo comienzan a presentar limitaciones cuando se intentan analizar conjuntos de datos con una gran cantidad de dimensiones, este problema es conocido como la "maldición de la dimensionalidad" y surge debido al hecho de que el número de posibles configuraciones distintas de un conjunto de variables aumenta exponencialmente a medida de que el número variables aumenta [17].

El aprendizaje profundo es un subcampo de la inteligencia artificial, y específicamente del aprendizaje automático, compuesto de algoritmos inspirados vagamente en la estructura y funcionamiento del cerebro humano, donde se buscan obtener representaciones de datos mediante una estructura multi-capas de algoritmos conocidos como redes neuronales artificiales.

Goodfellow I., Bengio Y. and Courville A. (2016) [17], definen el aprendizaje profundo como una variación del aprendizaje automático que consigue un gran poder y flexibilidad al aprender a representar el mundo como una jerarquía anidada de conceptos, con cada concepto definido en términos de conceptos más simples, y representaciones más abstractas calculadas en términos de otras menos abstractas. Así mismo, François Chollet, creador de la popular librería de aprendizaje profundo Keras, define el aprendizaje profundo como "una forma de aprender representaciones de datos que pone énfasis en aprender capas sucesivas de representaciones cada vez más significativas" [10]. Estas representaciones en capas se aprenden mediante las ya mencionadas redes neuronales, por lo que es necesario explicar con mayor detalle dicho concepto.

Las redes neuronales artificiales o ANN por su nombre en inglés (Artificial Neural Networks) son sistemas computacionales inspirados en el funcionamiento del cerebro humano, y si bien no se consideran modelos matemáticos de sus contrapartes biológicas, si tienen principalmente dos similitudes. La primera similitud es que los bloques de construcción de ambas redes son dispositivos computacionales simples llamados "neuronas" que están altamente interconectados. La segunda es que las conexiones entre las neuronas determinan el funcionamiento de toda la red [32]. En la figura 2.1 se muestra una representación gráfica de una neurona dentro del marco del aprendizaje profundo, donde se pueden observar sus principales componentes[43]:

- Una o más conexiones de entrada (denotadas como X1, X2, ..., Xn) que sirven para traer información de otras neuronas.
- A cada entrada se le asigna un peso relativo (denotados como W1, W2, ..., Wn) que afectan el impacto de cada entrada en la neurona.
- Una o más conexiones de salida que sirven para llevar información a otras neuronas.
- Una función de activación que determina el valor de la señal de salida de la neurona con base en los valores de entrada proporcionados por las otras neuronas,

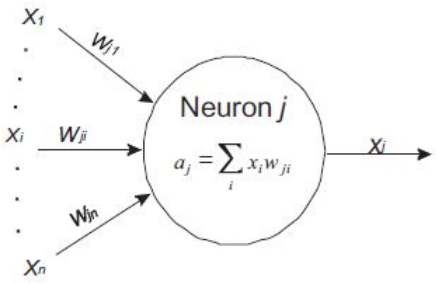


Figura 2.1: Diagrama de una neurona artificial.

los pesos asignados a cada enlace y la función de activación en sí misma. Si la neurona no tiene una función de activación, el valor de salida de la neurona corresponde a la sumatoria de todas sus entradas multiplicadas por los pesos de las mismas.

Existen varios tipos de redes neuronales, sin embargo uno de los más comunes y ampliamente usado es el Perceptrón Multicapa (MLP por su nombre en inglés, Multilayer Perceptron) también conocido como "Feedforward Deep Network". Los MLP basan su aprendizaje en un procedimiento supervisado, es decir, la red construye un modelo basado en ejemplos de datos donde se conocen las entradas y las salidas esperadas.

Un perceptrón multicapa puede tener cualquier número de neuronas las cuales se organizan en grupos llamados "capas". En la capa de entrada se encuentran representadas las condiciones iniciales de los datos de entrada, y en la capa de salida las categorías o valores salida del modelo. Es importante mencionar que la capa de salida puede tener más de una neurona, esto es útil en tareas de clasificación donde se tienen muchas posibles clases en cuyo caso cada neurona representaría una clase distinta. Es posible introducir más capas en el MLP entre la capa inicial y la de salida, dichas capas se conocen como "capas ocultas" y permiten que el modelo pueda resolver problemas no lineales obteniendo una mayor cantidad de representaciones del conjunto de datos dentro de cada capa. Una red neuronal con múltiples capas ocultas se conoce como red neuronal profunda y de ahí que viene el nombre de aprendizaje profundo. Un diagrama de un perceptrón multicapa [32] puede ser visualizado en la figura 2.2.

Si bien las redes neuronales profundas tienen la capacidad de clasificar correctamente clases no lineales, el solo aumentar la cantidad de capas ocultas no es suficiente para llevar a cabo esta tarea. Como se mencionó previamente, una parte importante de las neuronas de la red es lo que se conoce como función de activación, la cual es una función por la cual pasa la sumatoria de los parámetros de entrada de la neurona multiplicados por sus pesos. Si las neuronas de una red no tienen una función de activación, entonces la salida de la misma será solo dicha sumatoria, la cual es una función lineal. Es así que, aun si se tiene una red muy profunda, si las neuronas no cuentan con una función de activación, la red neuronal seguirá siendo lineal y será equivalente un modelo de

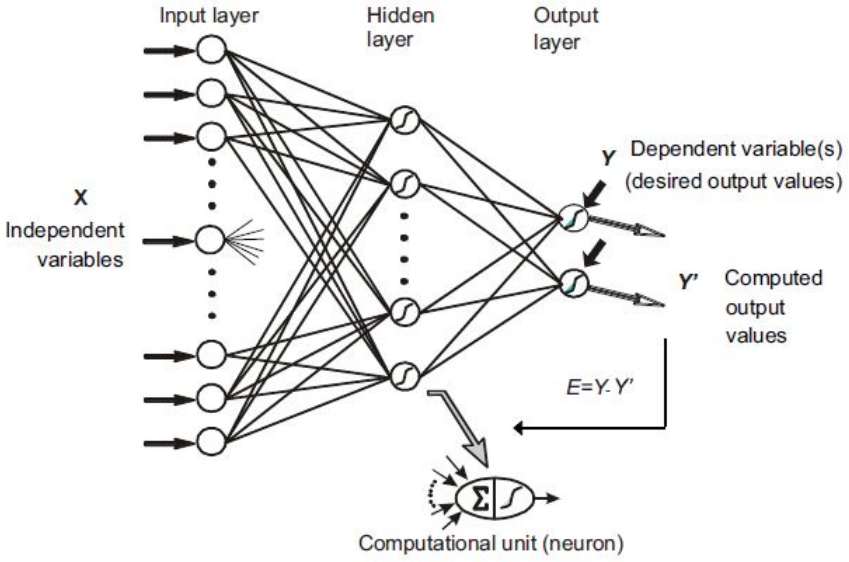


Figura 2.2: Diagrama de un perceptrón multicapa con una capa oculta.

regresión lineal. Es por ello que, para lograr resolver problemas no lineales, se induce en cada neurona de la red una función no lineal como función de activación. Usualmente todas las neuronas de una capa tienen la misma función de activación siendo las comunes las siguientes [43]:

- Función Identidad

$$f(a) = a \tag{2.1}$$

- Función de Actividad de Umbral

$$f(a) = \begin{cases} 1, a \geq 0 \\ 0, a < 0 \end{cases} \tag{2.2}$$

- Función Logística o Sigmoidal

$$f(a) = \frac{1}{1 + exp(-a)} \tag{2.3}$$

- Función Sigmoidal Bipolar

$$f(a) = \frac{2}{1 + exp(-a)} - 1 \tag{2.4}$$

- Función Tangente Hiperbólica

$$f(a) = \frac{exp(a) - exp(-a)}{exp(a) + exp(-a)} \tag{2.5}$$

- Función Rectificadora o ReLU

$$f(a) = \begin{cases} a, a \geq 0 \\ 0, a < 0 \end{cases} \tag{2.6}$$

El proceso de aprendizaje de una red neuronal es un proceso iterativo y consiste en la optimización de los valores de los pesos con el propósito de disminuir el valor proporcionado por una función de perdida, la cual funciona como un indicador del desempeño de la red en determinada tarea. En un inicio de este proceso de entrenamiento, los pesos de la red se inicializan con valores aleatorios. Después la red realiza una predicción del valor de salida y calcula su error, el cual es la diferencia entre el valor de salida esperado con el obtenido. A continuación la red identifica los pesos que influenciaron mayormente en el valor de salida obtenido y actualiza los valores de los pesos de toda la red empezando por las capas finales y terminando en la capa inicial, este algoritmo se conoce como algoritmo de "propagación hacia atrás" o "backpropagation" en inglés.

Es importante mencionar que un componente importante de las redes neuronales es el algoritmo que le permite optimizar sus pesos, ya que gracias a éste es que la red neuronal puede aprender. Durante el proceso de construcción de la red neuronal se debe seleccionar un algoritmo de optimización siendo el más común el algoritmo de "Descenso del Gradiente" y otros que derivan éste como el "Descenso del Gradiente Estocástico" el cual más rápido y por ende ampliamente utilizado.

No existe una única o mejor forma de construir una red neuronal, ya que es posible variar la forma en cómo se conectan los distintos nodos de la red, las funciones de activación de los nodos, su cantidad de capas, etc. Estas diferentes formas de construir una red neuronal definen lo que se conoce como "arquitectura de la red neuronal" y cada arquitectura se utiliza para realizar tareas distintas, por ejemplo, las Redes Neuronales Profundas son redes neuronales artificiales con múltiples capas ocultas y que permiten procesar datos más complejos al poder encontrar relaciones no lineales entre ellos. Una arquitectura de red neuronal muy conocida es la de las Redes Neuronales Convolucionales, las cuales son utilizadas para la clasificación de imágenes ya que cuentan con capas compuestas de filtros encargados de identificar patrones dentro de la imagen.

En este trabajo surgió principal interés en arquitecturas útiles para el procesamiento de secuencias de texto, es por ello que a continuación de describirán con mayor detalle las arquitecturas más utilizadas para este propósito.

2.3.1. Redes Neuronales Recurrentes

Las redes neuronales recurrentes o RNN por sus siglas en inglés ("Recurrent Neural Networks") son un tipo de red neuronal diseñada para procesar datos secuenciales y de longitud variable, es decir elementos que se encuentran relacionados entre sí y en donde el orden en el que son procesados los datos es importante por ejemplo secuencias de texto o datos que dependen del tiempo. A diferencia de las redes neuronales artificiales comunes, los valores de salida proporcionados por este tipo de arquitectura no solo

dependen solamente de los valores de entrada de la red y de los pesos de la misma, sino que también dependen de los valores que anteriormente pasaron por la red, es decir cada estado de la red depende de todos los estadios anteriores de la misma.

En teoría la arquitectura de las RNN les permitiría tener cierto grado de memoria sobre eventos pasados, sin embargo de forma práctica se descubrió que su memoria es muy limitada, es decir, que las RNN tienen problemas al tratar de retener información por largos periodos de tiempo, por ejemplo al tratar de procesar oraciones largas. Este problema fue estudiado por Bengio et al. (1994) [4], quienes plantearon las principales razones por las cuales es muy difícil entrenar una red neuronal recurrente que pueda aprender dependencias a largo plazo.

Los problemas que presentan las RNN con las dependencias a largo plazo son problemas generales que presentan las redes neuronales artificiales cuyo entrenamiento se basa en métodos basados en gradientes, y se presenta durante la fase de entrenamiento en redes muy profundas, es decir, con muchas capas ocultas. Estos problemas se conocen como los problemas de desvanecimiento y explosión del gradiente "Vanishing Gradient" y "Exploding Gradient" en inglés) y surgen durante el algoritmo de propagación hacia atrás, donde de forma general, los valores que llega a las primeras capas de la red sean números muy pequeños (en el caso del desvanecimiento del gradiente) impidiendo que la red pueda seguir aprendiendo y básicamente se estanque, o que sea lo contrario y se lleguen valores muy grandes que hace que la red no pueda converger en un valor óptimo.

Los problemas de desvanecimiento y explosión de gradientes son muy difíciles de identificar y se vuelven mucho peores en redes profundas. Es por ello que la arquitectura recursiva de las RNN, donde la información del gradiente pasa varias veces por las mismas capas de la red, hace que estos problemas se presenten con mayor facilidad, inclusive en redes que no tienen un gran número de capas ocultas. Surgieron varias propuestas para solucionar los problemas de gradientes sin embargo la más conocida y usada es la de las Redes LSTM o ("Long Short Term Memory Networks" en inglés).

2.3.2. Redes LSTM (Long Short Term Memory Networks)

Las redes LSTM son un tipo de redes neuronales recurrentes específicamente diseñadas para tratar con dependencias a largo plazo, evitando así los problemas de gradientes que presentaban las RNN sencillas. Las redes LSTM se componen de unidades llamadas "celdas" donde cada una de estas celdas guarda un "estado", los estados de celda son básicamente la información que la red neuronal decide almacenar o borrar en cada espacio de tiempo. El estado de la celda sólo puede ser modificado por otros componentes de la misma llamados "puertas". De forma general una celda LSTM está compuesta de al menos tres puertas: la puerta de entrada, de salida y de olvido, donde se implementan las funciones de escritura, lectura y eliminación de información dentro de la celda. Las puertas se componen de una función sigmoidal y una multiplicación elemento a elemento, además sus valores de salida no son binario sino analógicos dentro del rango de 0 a 1 [31]. La arquitectura general de una red LSTM puede verse en la

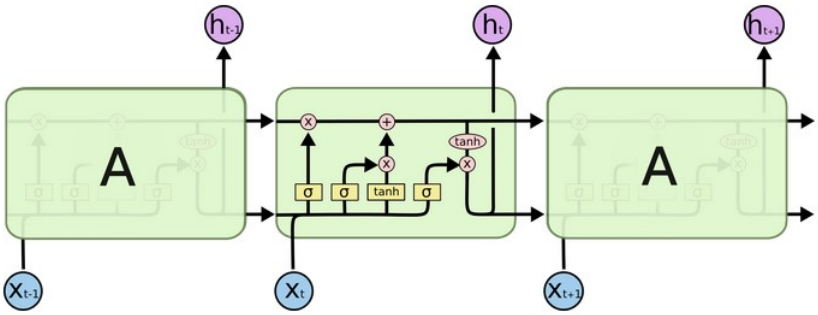


Figura 2.3: Diagrama de una celda LSTM

figura 2.3.

La arquitectura de las LSTM les permite almacenar información por un periodo de tiempo indefinido, agregando y quitando un poco de esa información en cada periodo de tiempo. Detallando un poco más el funcionamiento de cada puerta, si la puerta de entrada se encuentra debajo de un umbral de activación, la celda retendrá la información del estado pasado y en el caso contrario, combinara el estado pasado con la nueva información de entrada. La puerta de olvido reinicia el estado actual de la celda y la puerta de salida decide si el estado actual de la celda debe pasar a otras celdas o no.

2.3.3. Mecanismo de atención

Las redes neuronales recurrentes y las redes LSTM fueron las bases de modelos más complejos en el campo del aprendizaje profundo, un ejemplo de dichos modelos es el modelo "*seq2seq*" el cual demostró un enfoque muy efectivo para resolver tareas de traducción automática y modelado del lenguaje. De forma general el modelo "*seq2seq*" funciona recibiendo una secuencia de entrada y convirtiéndola en una secuencia distinta de salida que puede o no tener una longitud diferente a la secuencia de entrada. Ejemplos del uso de modelos "*seq2seq*" son el de traducir una oración de un idioma a otro o el de la creación de un chatbot que responda a una secuencia de entrada, es importante notar que en ambos casos la oración de entrada no tendrá necesariamente los mismos elementos que la oración de salida.

El modelo "*seq2seq*" se compone de un arquitectura codificador-decodificador o "encoder-decoder" en inglés, donde la función del codificador es la de comprimir la información de la secuencia de entrada en un vector de dimensión fija llamado "vector de contexto", el cual guardara (en teoría) un resumen de toda la secuencia de entrada y que a su vez sera la entrada del decodificador. La función del decodificador es la de producir una secuencia de salida con base en la información del vector de contexto. Tanto el codificador como del decodificador se componen de redes neuronales recurrentes como las LSTM, un ejemplo de un modelo "*seq2seq*" puede verse en la figura 2.4.

Los modelos "*seq2seq*" presentaron un gran avance en el campo del procesamiento de lenguaje natural profundo, sin embargo presentaban problemas al tratar con secuencias

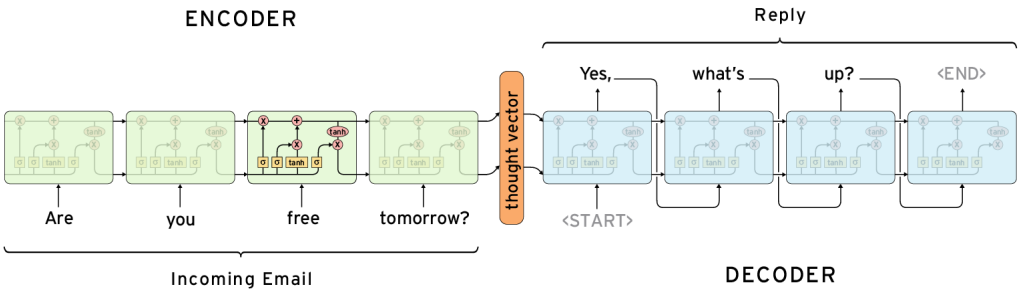


Figura 2.4: Ejemplo de modelo seq2seq

de entrada muy largas, esto es debido a que el codificador genera el vector de contexto después de procesar el último elemento de la secuencia, por lo que en secuencias muy largas hay una probabilidad alta de que el contexto inicial de la secuencia se haya perdido al momento de generar el vector de contexto.

Con el propósito de ayudar a los modelos "*seq2seq*" a memorizar el contexto de secuencias de entrada largas surge el concepto de "atención". El mecanismo de atención surge como propuesta para mejorar los modelos de traducción automática y ayudarlos a memorizar información en oraciones de entrada largas. Este mecanismo surge inspirado de la forma en como los humanos traducimos un texto y fue inicialmente introducido por Bahdanau et. al (2014) [3] y Luong et. al (2015) [26].

El mecanismo de atención propone que en una arquitectura codificador-decodificador como la "*seq2seq*" no se utilice un único vector de contexto de longitud fija, sino que además se agreguen pesos entre el vector de contexto y cada uno de los valores de entrada, estos pesos varían dependiendo del elemento de la secuencia que el decodificador este tratando de generar y son un indicador de que elementos de la entrada son más importantes para generar dicha salida. Este proceso busca simular lo que los humanos realizamos al momento de traducir una oración, y es el hecho de ver solo las palabras que nos otorgan información importante de la palabra que buscamos traducir en el momento y no ver toda la oración. De forma general una ilustración del mecanismo de atención puede observarse en la figura 2.5, así mismo, en la figura 2.6 se observan los pesos del mecanismo de atención durante una tarea de traducción del inglés al francés, en dicha figura los ejes horizontal y vertical corresponden a las palabras de la oración original en inglés y la oración generada en francés respectivamente. Los pesos en la gráfica están en escala de grises donde se ven completamente negros si el valor es 0 y blancos si es 1.

Es importante mencionar que si bien el mecanismo de atención surgió para mejorar las tareas de traducción automática, gracias a los excelentes resultados que produjo actualmente se utiliza para una gran cantidad de tareas tanto el campo del procesamiento de lenguaje natural como el de visión por computadora.

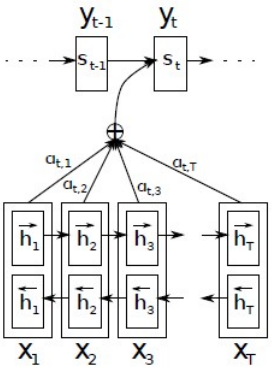


Figura 2.5: Representación gráfica del mecanismo de atención

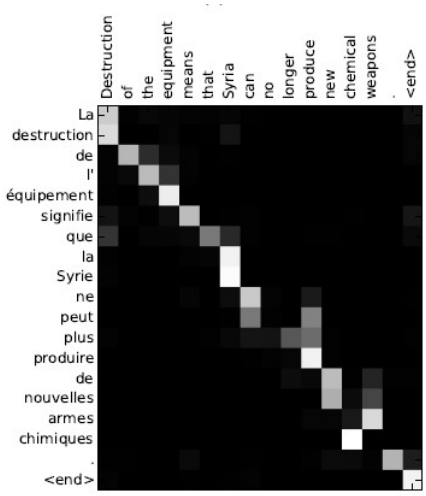


Figura 2.6: Visualización de la atención en una tarea de traducción de francés a inglés

2.3.4. Redes Transformer

Las redes Transformer son un tipo de redes neuronales propuestas por la empresa Google en el año 2017 y que fueron presentadas en el artículo "Attention es all you need" por Vaswani et. al (2017) [44]. Como su nombre lo indica, estas redes hace uso del mecanismo de atención presentado en la sección 2.3.3 y su uso ha ganado popularidad en los últimos años debido a los excelentes resultados que presentan.

Las redes Transformer surgen con el propósito de mejorar los modelos que entregaban los mejores resultados en tareas donde se procesan secuencias de texto como son el caso del modelado del lenguaje y la traducción automática. En este tiempo dichos modelos utilizaban redes neuronales recurrentes como las LSTM y las GRU y tenían arquitecturas codificador-decodificador como el modelo seq2seq mencionado en la sección 2.3.3, donde además se utilizaba el mecanismo de atención para conectar el codificador

con el decodificador y mejorar el rendimiento del modelo.

Los modelos basados en redes neuronales recurrentes fueron establecidos como enfoques efectivos para resolver tareas de procesamiento secuencial de texto, sin embargo, debido a su arquitectura recurrente, estos modelos presentan una naturaleza secuencial que impide la paralelización de procesos durante el entrenamiento de los mismos. Esto se vuelve un factor importante cuando se procesan largas secuencias de texto donde el rendimiento del modelo puede disminuir debido a restricciones en la memoria. Debido a esto se comenzó a incorporar de forma más frecuente en estos modelos el mecanismo de atención, que ayuda a los mismos a preservar información aun si se procesan secuencias de texto muy grandes.

Las redes Transformer hacen uso solamente del mecanismo de atención, removiendo así cualquier componente recurrente de la red y por ende eliminando sus limitaciones. Esto permite alcanzar un mayor nivel de paralelización durante el entrenamiento y por ende mejores resultados. Las redes Transformer tienen una estructura codificador-decodificador donde se integran varias capas de "atención multi-cabeza", este tipo de atención le permite al modelo enfocarse en diferentes posiciones de la secuencia de entrada a diferencia de la atención sencilla donde principalmente el mecanismo busca enfocarse en una posición. La arquitectura de la red Transformer presentada en el articulo original de Google [44] puede verse en la figura 2.7.

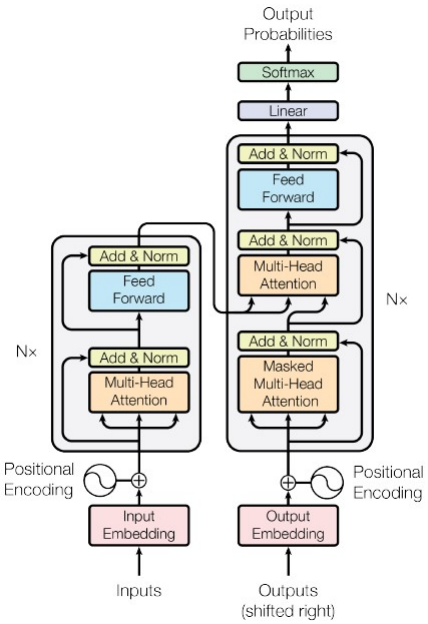


Figura 2.7: Arquitectura de red Transformer

2.4. Sistemas de Diálogo Orientados a Tareas

Un sistema de diálogos es un sistema computacional capaz de comunicarse con los seres humanos mediante, texto, voz, gestos u otros medios. También son conocidos como "agentes conversacionales" y se utilizan para resolver una amplia variedad de problemas, ejemplos comerciales de este tipo de sistemas son "Assistant" de Google, "Alexa" de Amazon y "Siri" de Apple.

De manera general, un sistema de diálogo orientado a tareas puede resolver varias tareas para las que fue programado, por ejemplo enviar un mensaje de texto, reproducir una canción en un dispositivo, hacer una llamada, hacer compras en línea, etc. Es por ello que lo primero que este tipo de sistemas debe aprender a realizar es a identificar lo que le está solicitando el usuario. Esta actividad se conoce como "clasificación de intenciones". La clasificación de intenciones puede definirse como, dada una declaración del usuario, ya sea por texto, voz u otros medios, identificar cuál de los escenarios predeterminados esta tratado de ejecutar el usuario. Esta es una tarea de clasificación ya que se debe identificar la intención correcta de una lista finita de intenciones disponibles.

Junto con la clasificación de intenciones el sistema de diálogo debe de llevar a cabo una tarea de "llenado de ranura" (llamada en ingles "slot filling" o también "slot tagging"), esta tarea consiste en que, una vez identificada la intención y mediante la interacción con el usuario, el sistema recolecta toda la información que necesita para ejecutar la solicitud del usuario. Un ejemplo de lo anterior puede ser si el usuario le pide al sistema de diálogo reproducir una canción, el sistema identificara la intención como "reproducir música" pero deberá solicitar información al usuario como el nombre de la canción a reproducir y el artista que la interpreta. Como una analogía, se puede considerar la intención como un formulario para una solicitud y las ranuras o "slot" son el conjunto de campos que deben de ser rellenados para llevar a cabo la solicitud.

En la figura 2.8 se muestra un diagrama de la estructura general de un sistema de diálogo orientado a tareas. Dicho diagrama fue obtenido del curso en línea de Procesamiento de Lenguaje Natural ofrecido por la National Research University Higher School of Economics en la plataforma Coursera [36]. En dicho diagrama se muestra de forma general el flujo de interacción entre un sistema de dialogo orientado a tareas y un usuario. Primero el usuario hace una solicitud al sistema por medio de un comando de voz o texto, en caso de que sea por comando de voz la solicitud pasa primero por un sistema de reconocimiento automático del habla que la convertirá a texto. A continuación la solicitud pasa por un programa de entendimiento de lenguaje natural, es en este módulo donde el sistema identifica la intención y las ranuras proporcionadas por el usuario y es importante resaltar que, un usuario puede solicitar la ejecución de una intención de múltiples formas pero un buen sistema de entendimiento de lenguaje natural debe identificar la intención solicitada correctamente sin importar como lo exprese el usuario. Una vez identificados los campos anteriores el sistema ejecuta un módulo llamado "administrador de diálogo", este módulo será el encargado de indicar si todas las ranuras de la intención ya fueron llenadas. En caso negativo deberá solicitarle al usuario mediante un comando de voz que proporcione la información faltante, y en

caso positivo el administrador de dialogo deberá ejecutar la intención (normalmente accediendo a un servicio de backend), y una vez completado deberá retroalimentar al usuario mediante un comando de voz.

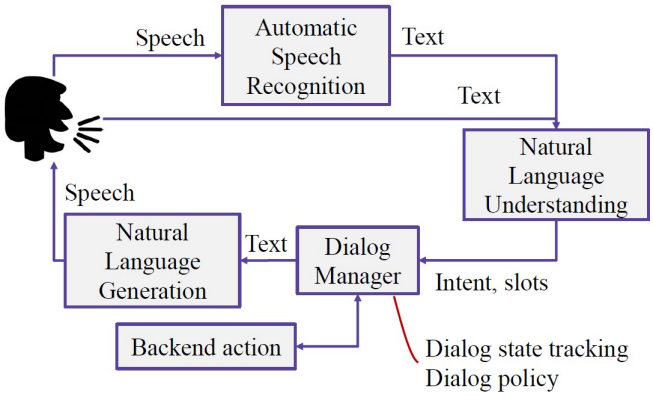


Figura 2.8: Estructura general de un sistema de diálogo orientado a tareas

También es importante mencionar que, en el caso de los asistentes virtuales que funcionan por voz, y que son un tipo de sistema de dialogo orientado a tareas, no se espera que el asistente reaccione a todas las conversaciones que suceden en su entorno, solo debe reaccionar cuando se invoca al asistente por medio de lo que se conoce como "Wake Word". Por ejemplo, considerando el comportamiento de los productos comerciales que existen actualmente, el asistente virtual de la empresa Amazon solo va a reaccionar cuando se le invoca diciendo la palabra "Alexa", de la misma forma el asistente de la empresa Apple solo reaccionara cuando se diga la palabra "Siri".

2.5. Computación en la Nube

De forma general, la computación en la nube pude definirse como la distribución de servicios computacionales a través del Internet y donde el usuario solo paga por la cantidad por la cantidad de recursos que utiliza, ejemplos de estos servicios son el almacenamiento de archivos, bases de datos, servidores, software, entre otros. El uso de servicios computacionales en la nube está siendo una práctica más recurrente actualmente dado que ofrece varias ventajas durante el desarrollo e implementación de aplicaciones. Entre los principales beneficios de utilizar servicios de computación en la nube están los siguientes:

- **Reducción de costos.** Uno de los principales beneficios de utilizar servicios en la nube es la reducción de costos al desarrollar y mantener aplicaciones, ya que, al usar servidores previamente configurados y con el software necesario instalado, la computación en la nube elimina la necesidad tanto de comprar hardware y software, como de desplegar y mantener infraestructura. Además es importante

aclarar que los costos de uso de servicios en la nube son mucho menores a lo que se pagaría si los recursos fueran completamente administrados por el usuario.

- **Escalamiento.** Otro de los grandes beneficios de usar servicios en la nube es la posibilidad de aumentar la cantidad de recursos computacionales que se consumen según sea necesario. Esta característica elimina la necesidad de aprovisionar recursos en exceso para lidiar con niveles pico de actividad y solo se consumen los recursos que se necesitan.
- **Agilidad.** Los servicios en la nube son de fácil acceso y ofrecen una amplia gama de tecnologías diferentes lo que le permite a los desarrolladores implementar y mejorar servicios y aplicaciones de forma rápida y sencilla.
- **Desempeño y Confiabilidad.** Los grandes proveedores de servicios en la nube como Microsoft, Amazon y Google cuentan con varios centros de datos a lo largo del mundo, donde los servidores son constantemente actualizados y se mantienen en condiciones óptimas para su funcionamiento. Además constantemente se busca reducir la latencia para los usuarios y se generan copias redundantes de los datos para que, en caso de un desastre, no se pierda información y el servicio no se interrumpa.
- **Seguridad.** La mayoría de proveedores de servicios en la nube implementan una serie de políticas y tecnologías que fortalecen las medidas de seguridad en sus centros de datos y así, ofrecen un alto nivel de protección de los datos y aplicaciones de los usuarios.

Utilizar servicios en la nube facilita la labor de desarrolladores y departamentos de Tecnologías de la Información (TI) al ofrecer distintas herramientas pre-configuradas y de fácil acceso, y así, permitiéndoles enfocarse más en realizar sus tareas principales e invirtiendo menos tiempo en tareas secundarias como mantenimiento y aprovisionamiento de recursos o instalación de software. Conforme ha incrementado la popularidad de los servicios en la nube, han surgido diferentes modelos que buscan satisfacer las necesidades de distintos tipos de usuarios al ofrecer distintos niveles de control, flexibilidad y administración. Los principales modelos de servicios en la nube son los siguientes:

- **Infraestructura como Servicio (IaaS).** Es la categoría más básica de servicios de computación en la nube, donde se pueden rentan los bloques fundamentales para construir una infraestructura TI. Algunos ejemplos de estos servicios son servidores, máquinas virtuales con sistemas operativos específicos, redes y almacenamiento. Este tipo de modelo ofrece el mayor grado de flexibilidad y control sobre los recursos TI, sin embargo, también requiere de una mayor cantidad de conocimientos técnicos para utilizar correctamente los servicios.
- **Plataforma como Servicio (PaaS).** Este tipo de modelo ofrece servicios que suministran un ambiente para desarrollar, probar, desplegar o administrar aplicaciones. Estos servicios eliminan la necesidad de las organizaciones de administrar

infraestructura IT y en su lugar, enfocarse en el desarrollo, despliegue y administración de sus aplicaciones. Los servicios PaaS están una capa por encima de los servicios IaaS, por ello todos los servicios IaaS como almacenamiento, redes y servidores, también son incluidos dentro de los servicios PaaS, con la diferencia que el usuario no puede administrar directamente dichos servicios, y en su lugar se le proporcionan una serie de herramientas y software que puede utilizar para construir aplicaciones por encima de los servicios IaaS.

- **Software como Servicio (SaaS).** Este modelo le ofrece al usuario un producto completo y funcional, es decir, en la mayoría de las ocasiones SaaS se refiere a un producto final destinado a ser utilizado por múltiples usuarios. Los servicios bajo el modelo SaaS son completamente alojados y administrados por el proveedor de servicios en la nube, el cual también se encarga de mantener la aplicación funcionando correctamente realizando las actualizaciones de software y seguridad necesarias. Al utilizar una aplicación bajo el modelo SaaS, el usuario no tiene que preocuparse sobre la infraestructura de dicha aplicación o el cómo adminístrala, sino solo en cómo usarla. Para conectarse a una aplicación SaaS en la nube el usuario tiene que utilizar por lo general un navegador web. Algunos ejemplos comunes de este tipo de aplicaciones son el correo electrónico, calendarios y servicios de Office.

Capítulo 3

Estado del Arte

En este capítulo se presentarán los avances tecnológicos más recientes relacionados con este proyecto de tesis. Este capítulo se divide principalmente en dos secciones, primero se presentarán otros proyectos recientes diseñados para asistir a personas con problemas de pérdida de memoria, lo cual servirá para conocer otros enfoques abordados para tratar el mismo tema de investigación. Después se presentaran los principales avances en inteligencia artificial, en particular dentro del área del aprendizaje profundo, para resolver tareas de procesamiento natural. El conocimiento de estos avances servirá para conocer, tanto la forma más efectiva para resolver tareas de procesamiento de lenguaje natural, como las limitantes que tendrá el proyecto.

3.1. Sistemas de asistencia para personas con trastornos de memoria

Actualmente existen diversas propuestas de sistemas que ayudan a pacientes con trastornos de memoria, sin embargo, la mayoría de estos sistemas están orientados al monitoreo de pacientes con el propósito de prevenir accidentes, vigilar cambios repentinos en sus condiciones de salud o están enfocados a la detección temprana de alguna enfermedad, en lugar de ayudarlos a vivir una vida independiente. A continuación se presentaran algunas propuestas desarrolladas en torno a los sistemas de asistencia de memoria.

Algunos estudios utilizan herramientas como el Internet de las Cosas (IoT por sus siglas en inglés) para hacer sistemas de monitoreo robustos, ejemplo de ello es la propuesta de Haruka Ishii et. al (2016) [22] donde se presenta una herramienta que busca detectar de forma temprana los síntomas de la demencia en personas mayores que viven solas. Este sistema consiste en instalar sensores en la vivienda de las personas para después analizar los datos obtenidos de los mismos y así identificar si la persona presenta los síntomas iniciales de demencia.

Algunos investigadores han propuesto otros enfoques para el cuidado de pacien-

tes con trastornos de memoria orientándose menos al monitoreo de los mismos y enfocándose más en ayudarlos a alcanzar un mayor grado de independencia. Airehrour et. al (2018) [1] presenta el diseño de una herramienta de asistencia para adultos mayores y personas con demencia la cual consiste de un sistema de monitoreo y recordatorios. El sistema de monitoreo consiste de varios sensores ubicados en partes específicas de la casa del paciente y cuyo propósito es identificar comportamientos de riesgo, como es el caso de si el paciente dejo una llave del agua abierta o si se encuentra durmiendo demasiadas horas al día, en cuyo caso de manera automática notificaría a un contacto de emergencia. El sistema de recordatorios hace uso de un sistema de almacenamiento en la nube y permite que un cuidador suba videos e imágenes de los seres queridos del paciente, eventos recientes en su vida e información de sus medicamentos. Seguido a esto, el sistema enviará al paciente recordatorios (en forma de correos y mensajes de texto) sobre los horarios de consumo de sus medicamentos, así como recomendaciones para que observar el material audio visual cargado en la nube. Esta herramienta no solo busca ayudar a las personas mayores con sus tareas diarias sino también funciona como un sistema de monitoreo para los cuidadores de los pacientes.

Weerakoon et. al (2018) [46] desarrollan una aplicación interactiva para retrasar el deterioro de funciones cognitivas y motoras en pacientes que sufren la enfermedad de Alzheimer. Esta aplicación se centra en la realización de actividades que permitan fortalecer los cuatro pilares de la enfermedad de Alzheimer: memoria a largo plazo, memoria a corto plazo, habilidades de lenguaje y habilidades motoras finas. Además esta aplicación permite a los doctores monitorear los puntajes obtenidos por los pacientes y con base en ello, verificar el avance de la enfermedad y decidir tratamientos adecuados.

Tokunaga et. al (2016) [42] implementan un sistema de recordatorios en la forma de un chatbot animado para asistir a personas mayores con problemas de demencia. La motivación para realizar este sistema bajo el enfoque de interacción Humano-Computadora es el de hacer una herramienta que sea más amigable y que pueda asistir a los pacientes de forma menos mecánica y simulando más una interacción humana.

3.2. Plataformas para desarrollo de agentes conversacionales

Construir todos los componentes necesarios para implementar un agente conversacional funcional es una tarea muy compleja, ya que como visto en las sección 2.4 es necesario implementar e integrar diversos componentes, afortunadamente actualmente existen diversas plataformas de Procesamiento de Lenguaje Natural que facilitan esta tarea al ofrecer a los usuarios la posibilidad de entrenar un agente conversacional al definir una lista finita de intenciones, entidades y frases ejemplo para cada intención.

Debido a la creciente popularidad de los sistemas de dialogo, actualmente la mayoría de la empresas de tecnología más grandes del mundo cuentan con una plataforma para crear agentes conversacionales, algunas de las plataformas mas populares son las

siguientes [6]:

- DialogFlow de Google
- Wit.ai de Facebook
- Luis.ai de Microsoft
- IBM Watson de IBM
- Amazon Lex de Amazon
- Snips (Código abierto)
- Rasa NLU (Código abierto)

Se analizaron las características principales de estas plataformas con el propósito de determinar cuáles de ellas pueden ser utilizando en este proyecto. Entre las principales características que se tomaron en consideración fueron que el Framework tuviera suporte para el idioma español, sus costos de uso, los servicios y sistemas que incluye, su capacidad de integración con otros servicios de backend y su interfaz para programar diálogos e intenciones. A continuación se describen las características, ventajas y desventajas de los principales Frameworks considerados para la elaboración de este proyecto.

3.2.1. Snips

Snips es una plataforma de desarrollo de aplicaciones de voz con inteligencia artificial que, a diferencia de sus competidores, se integra dentro del dispositivo a utilizar y por ello no tiene la necesidad de enviar datos a la nube para su funcionamiento. Esto ofrece a las empresas y desarrolladores interfaces conversacionales privadas para interactuar con sus productos y con un desempeño similar al de otras plataformas basadas en la nube [41]. La plataforma de Snips tiene integrados módulos de Reconocimiento Automático del Habla (ASR) y Entendimiento de Lenguaje Natural (NLU), además cuenta con una interfaz de usuario muy fácil de utilizar y que permite la definición de intenciones y ranuras del asistente. Entre las principales ventajas de Snips con respecto a sus competidores se encuentra la posibilidad de personalizar la "Wake Word" del asistente, la posibilidad de integrarlo en plataformas de desarrollo de fácil acceso como la Raspberry Pi, además de la privacidad que por diseño ofrece el sistema. Para la elaboración de este proyecto es importante considerar los idiomas que soporta la plataforma, y en este caso Snips actualmente tiene soporte para 7 idiomas incluidos el español y el inglés. La principal desventaja de Snips con respecto a sus competidores es que el manejo del dialogo con el usuario no se hace de forma directa en la interfaz web, y en su lugar se debe programar en el código de la aplicación.

3.2.2. Dialogflow

DialogFlow es un producto de la empresa Google, anteriormente conocido como API.AI, que ofrece un SDK para el desarrollo de aplicaciones por voz con inteligencia artificial. La SDK contiene un sistema automático de reconocimiento del habla, un módulo de procesamiento de lenguaje natural y un módulo de conversión de texto a habla. Además ofrece un interfaz web para construir las diferentes intenciones del asistente así como la posibilidad de probar distintos escenarios de conversación con los usuarios [19]. Al ser un producto de Google, DialogFlow permite una sencilla integración con los servicios en la nube de la misma empresa lo cual agiliza en gran medida la creación de nuevas y más complejas intenciones durante el desarrollo de un asistente virtual. Además su interfaz web permite un desarrollo más robusto tanto de los diálogos entre el sistema y los usuarios como de las intenciones disponibles en el asistente virtual. Actualmente DialogFlow soporta más de 14 idiomas, incluidos el español, inglés, francés, alemán, portugués, chino y japonés. También es importante mencionar que DialogFlow incluye una herramienta de análisis que permite obtener información y estadísticas del comportamiento del sistema, en dicha herramienta podemos ver información como la cantidad de solicitudes que se le hacen al asistente por día, intenciones solicitadas por el usuario y tiempo de respuesta del sistema. DialogFlow es también una herramienta que permite crear aplicaciones (o también llamadas "Acciones") para el asistente de Google llamado "Google Assistant". Si bien esto permite una rápida implementación de los asistentes de voz creados al volverlos una extensión de Google Assistant, también limita la personalización de los mismos al no permitir que se pueda cambiar la "Wake Word" o instalarlos en un dispositivo que no tenga instalado también a Google Assistant.

3.2.3. RASA

Al igual que Snips, RASA es una plataforma de código abierto y provee una infraestructura estándar para la creación de sistemas de IA conversacionales. Esta plataforma ofrece un alto nivel de personalización así como la posibilidad de ejecutarse en un dispositivo local (lo que permite un mayor nivel de privacidad) o en la nube, además al ser una plataforma de código abierto su uso es gratuito [39]. El módulo de entendimiento de lenguaje natural (NLU) en RASA debe ser definido y entrenado por el usuario. Esto último si bien permite un alto nivel de personalización al poder modificar la tubería de procesos ("pipeline" en inglés) para el procesamiento del texto de entrada, también hace más tardado el proceso de implementación del asistente. RASA pude ser utilizado para construir asistentes virtuales en cualquier idioma que se desee, ya que se puede modificar la tubería de procesos creando representaciones de palabras con los datos específicos de nuestro proyecto o utilizando representaciones de palabras predefinidas. Es importante mencionar que este Framework no cuenta con una interfaz web para la creación de intenciones y el manejo del dialogo entre el asistente y el usuario, por lo que esto se debe hacer de forma manual modificando los archivos ejemplo proporcionados por la empresa.

3.2.4. Wit.ai

Wit.ai es un framework propiedad de Facebook con capacidades avanzadas de procesamiento de lenguaje natural. Al ser propiedad de Facebook es una herramienta muy utilizada para el desarrollo de bots para los servicios de mensajería instantánea ("Messenger") de la misma empresa [14]. Wit.ai fue inicialmente desarrollado por una empresa pequeña y después fue adquirido por Facebook en el año 2015, sin embargo este framework sigue manteniéndose como un proyecto de código abierto por lo que su uso es gratuito. Entre las principales ventajas de Wit.ai es que tiene soporte para una amplia cantidad de idiomas incluido el español, además al ser un proyecto de código abierto se beneficia de contar con una amplia comunidad de desarrolladores donde es posible observar el trabajo de otras personas y aprender de ello. Otro factor importante a mencionar es que Wit.ai destaca principalmente por sus capacidades de procesamiento de lenguaje natural que se encuentran al nivel de otras grandes empresas como Google, Amazon e IBM. Entre las principales desventajas de este framework es que su interfaz web para administrar diálogos y crear de intenciones es mucha más compleja de utilizar y está más limitada que la de otros frameworks como DialogFlow.

3.2.5. Plataformas no consideradas para el proyecto

Existen otras tres plataformas populares para el desarrollo de asistentes virtuales pero no están siendo considerados para la elaboración de este proyecto por las razones siguientes:

Amazon Lex. Actualmente solo cuenta con soporte para el idioma ingles por lo que no es posible desarrollar en él este asistente que deberá trabajar en el idioma español.

Luis de Microsoft. Actualmente el uso de este framework tiene un costo, si bien existe una versión de prueba gratuita dicha versión no permite el procesamiento de comandos de voz.

IBM Watson. Actualmente el uso de este framework tiene un costo, si bien existe una versión de prueba gratuita dicha versión cuenta con funciones limitadas.

3.3. Aprendizaje Profundo en Procesamiento de Lenguaje Natural

La inteligencia artificial juega un papel importante en la elaboración de este proyecto, ya que lo últimos años han sido caracterizados por la gran cantidad de avances que han surgido en distintas áreas de las ciencias computacionales, como visión por computadora y procesamiento de lenguaje natural, gracias a la utilización de técnicas disruptivas como el reconcomiendo de patrones utilizando aprendizaje profundo.

El año 2018 fue marcado con grandes avances en el área del procesamiento de lenguaje natural. Se descubrieron nuevas y más eficientes formas procesar palabras al

capturar mejor su significado haciendo uso del contexto se utilizan, además se desarrollaron nuevas arquitecturas y modelos de aprendizaje profundo, que aprovechan la grandes cantidades de texto que se encuentran en Internet de forma gratuita y en múltiples idiomas.

En esta sección se revisarán los últimos y más significativos avances dentro del área de aprendizaje profundo para resolver tareas de procesamiento de lenguaje natural. Primero se examinarán las mejores formas representar palabras en cualquier idioma como números para después ser procesados y analizados por un modelo computacional. Después, se analizarán los modelos de aprendizaje profundo considerados para el desarrollo de este proyecto, donde se tomaron en cuenta principalmente dos factores: su efectividad para resolver la tarea de responder preguntas y la capacidad del modelo para funcionar eficientemente en el idioma español.

3.3.1. Representaciones de palabras dependientes de contexto

Para que las computadoras puedan procesar palabras es necesario convertir estas últimas en una representación numérica. Durante los últimos años se utilizaron principalmente dos métodos conocidos como Word2Vec [29] y GloVe [33], ambos métodos demostraron que es posible utilizar vectores, de múltiples dimensiones, para representar palabras y estos vectores pueden capturar de cierta forma relaciones semánticas y sintácticas entre ellas. Un ejemplo de la anterior es que el vector de la palabra “rey” está esta relacionado con los vectores de palabras como “corona”, “reina” y “principie”. En la figura 3.1 se detallan ejemplos de relaciones obtenidas por el modelo Word2Vec entrenado con 783M de palabras y utilizando vectores de 300 dimensiones presentado por Pennington et. al (2014) [33], donde en la primera columna se realiza una resta de dos vectores y después, en cada una de las columnas subsecuentes, se muestra el resultado de sumar otro vector. Por ejemplo, en la primera fila de la tabla se muestran los resultados de restar los vectores “France” - “Paris” (donde se obtiene la noción de “capital”) y después sumar al resultado de dicha resta los vectores de las palabras “Italy”, “Japan” y “Florida”; los resultados finales de esta operación son “Rome”, “Tokyo” y “Tallahassee” respectivamente, es decir, las capitales de los lugares indicados.

El enfoque previamente descrito fue usado durante varios años y permitió la opción de descargar y utilizar vectores de palabras previamente entrenados en grandes cantidades de texto, sin embargo este enfoque presentaba un grave problema y es que para cada palabra solo existía un vector relacionado, por lo que palabras que se escriben igual pero que tiene múltiples significados como “banco” (donde puede referirse a la institución encargada de realizar operaciones financieras o un asiento de madera) no eran representadas correctamente. En respuesta a esto último Peters et. al, 2018 [34] proponen ELMo (Embeddings for Language Models), una forma de representar palabras que, a diferencia del enfoque tradicional, toma en cuenta el contexto de la palabra cuando se calcula su vector correspondiente, es decir, para cada palabra se tienen vectores diferentes que están en función de toda la oración analizada.

Los vectores de ELMo son derivados de una LSTM bidireccional entrenada en un

Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

Figura 3.1: Ejemplos de relaciones entre pares de palabras usando un modelo Word2Vec

corpus grande con el objetivo de modelar lenguaje. El modelado de lenguaje consiste en intentar predecir la siguiente palabra en una secuencia de palabras y de esta forma ELMo logra un entendimiento del lenguaje. Experimentalmente se descubrió que los niveles superiores de la LSTM de ELMo capturan aspectos dependientes del contexto del significado de las palabras mientras que los niveles más bajos capturan aspectos de sintaxis. El éxito de ELMo se debió a que estas representaciones pueden ser agregadas a modelos existentes para mejorarlos de manera significativa. Después de la publicación de ELMo surgieron varios modelos que hacían uso de estas representaciones vectoriales basadas en contexto siendo un modelo llamado BERT el más reciente y más exitoso de todos.

3.3.2. BERT

El Procesamiento de Lenguaje Natural o NLP por sus siglas en inglés, es un área de investigación que ha sido objeto de estudio desde hace varias décadas debido a la utilidad que presenta el poder hacer que una maquina logre entender el significado de palabras utilizadas por los humanos. Como se revisó en la sección 2.1, el procesamiento de lenguaje natural estudia formas de resolver, con el mejor desempeño posible, una gran cantidad de tareas como es el caso de la traducción automática, el reconocimiento de entidades del lenguaje, el responder a preguntas, entre otras. Sin embargo, a diferencia de otras áreas de investigación, el crear modelos que puedan resolver correctamente tareas de procesamiento de lenguaje natural presenta un grado de dificultad extra, ya que no solo se debe resolver una tarea específica, sino que además que existe una dependencia directa al lenguaje en el que se desarrolla el modelo, es decir, un modelo creado para resolver una tarea específica en, por ejemplo, el idioma inglés no funcionaría para resolver la misma tarea en español.

BERT (Bidirectional Encoder Representations from Transformers) es un modelo de aprendizaje profundo desarrollado por Devlin et. al (2018) [12] de la empresa Google, que es considera un hito dentro del campo del procesamiento de lenguaje natural, ya que no solo obtuvo un desempeño mucho más alto del que se tenía en el momento

en más de 11 tareas de procesamiento de lenguaje natural, sino que además propone una forma de entrenar modelos que entiendan un idioma específico y después reutilizar dichos modelos para resolver tareas específicas en el idioma seleccionado [2].

De forma más específica, el modelo BERT busca "aprender" un idioma específico al obtener representaciones vectoriales de las palabras que componen el idioma tomando en cuenta su contexto, es decir, no solo tomando en cuenta cada palabra individualmente, sino considerando también otras palabras que aparecen junto a ella en las mismas oraciones del conjunto de datos de entrenamiento 3.3.1. Es importante mencionar que, para obtener la representación vectorial de una palabra dentro de una oración, el modelo BERT utiliza al mismo tiempo tanto el contexto del lado izquierdo de la palabra como el del lado derecho, es decir, el modelo no solo toma en cuenta las palabras previas sino también las que siguen en la oración. Esto presenta una diferencia de su predecesor ELMo [34], donde el modelo solo era capaz de considerar en un mismo instante de tiempo el contexto del lado izquierdo o el del derecho, pero no ambos.

Existen dos fases principales para entrenar un modelo BERT: pre-entrenamiento y sintonización fina, representadas en la figura 3.2. Durante el pre-entrenamiento el modelo es entrenado, utilizando datos no etiquetados, para resolver dos tareas llamadas: modelado de lenguaje enmascarado y predicción de oración siguiente. Para la sintonización fina el modelo es inicializado con los paramentos obtenidos durante el pre-entrenamiento, estos son modificados al entrenar nuevamente el modelo pero esta vez utilizando datos etiquetados en tareas específicas. El rasgo característico de BERT es su arquitectura unificada utilizada para resolver distintas tareas, es decir, todos los modelos BERT completamente entrenados tienen la misma arquitectura pero sus parámetros son distintos entre sí, ya que a pesar de a ver sido inicializados con los mismos valores, estos se modifican durante la fase de sintonización fina para resolver una tarea específica de forma más efectiva.

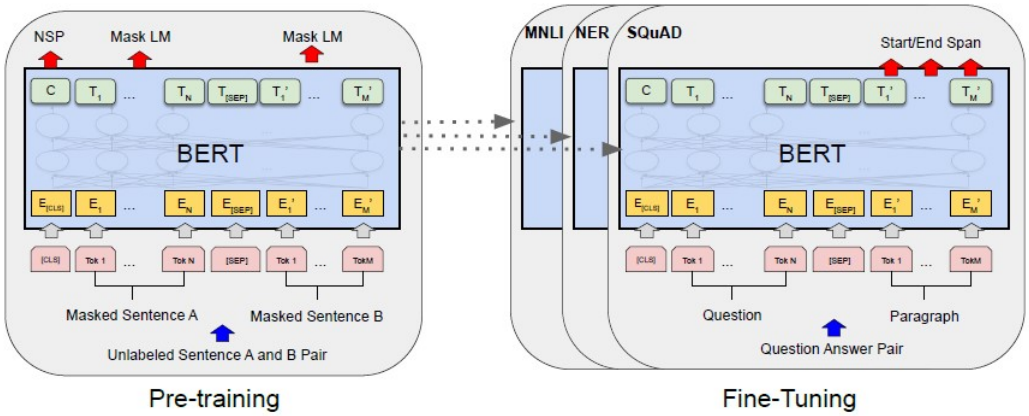


Figura 3.2: Representación gráfica de las fases de pre-entrenamiento y sintonización fina del modelo BERT.

La arquitectura del modelo BERT consiste de un modelo Transformer bidireccional mulit-capa [44]. Devlin et. al (2018) [12] presentaron en su trabajo original dos modelos con distinto tamaño: BERT-base que cuenta con 12 capas de un tamaño de 768 y 12 cabezas de auto-atención dando un total de 110M de parámetros, y BERT-grande que cuenta con 24 capas de un tamaño de 1024 y 16 cabezas de auto-atención dando un total de 340M de parámetros.

3.3.2.1. Representación de datos de entrada

Para que el modelo BERT pueda resolver distintas tareas de procesamiento de lenguaje natural, sin necesidad de modificar su arquitectura, se utiliza un formato de datos de entrada que le permite al modelo tratar tanto con oraciones simples como con pares de oraciones. Esto se debe a que algunas tareas del procesamiento de lenguaje natural, como es el caso de la clasificación de oraciones, solo necesitan que se introduzca una oración a la vez al modelo, mientras que otras tareas, como es el caso de la traducción automática, requieren que se introduzcan pares de oraciones.

El formato para los datos de entrada del modelo BERT puede verse en la figura 3.3, donde se puede observar que a las oraciones de entrada se les agregan dos caracteres especiales: [CLS] que indica el inicio de la primera oración y [SEP] que, en caso de necesitar introducir pares de oraciones al modelo, se encarga de separar la primera de la segunda oración. A continuación, la secuencia de caracteres de entrada resultante pasa por tres capas, después de las cuales se genera una representación final que es introducida al modelo. La primera capa convierte cada palabra de la secuencia de entrada a una representación vectorial de varias dimensiones (proceso conocido como Tokenización), utilizando un método llamado "WordPiece Embedding" [49] y haciendo uso de un archivo de vocabulario. La segunda capa agrega información sobre el segmento al que pertenece cada palabra (primera o segunda oración de entrada) y la tercera capa agrega información sobre la posición de cada palabra en la oración de entrada, esto último es necesario ya que las arquitecturas Transformer, a diferencia de las redes LSTM revisadas en la sección 2.3.2, no pueden llevar por si solas un registro de la posición de cada palabra.

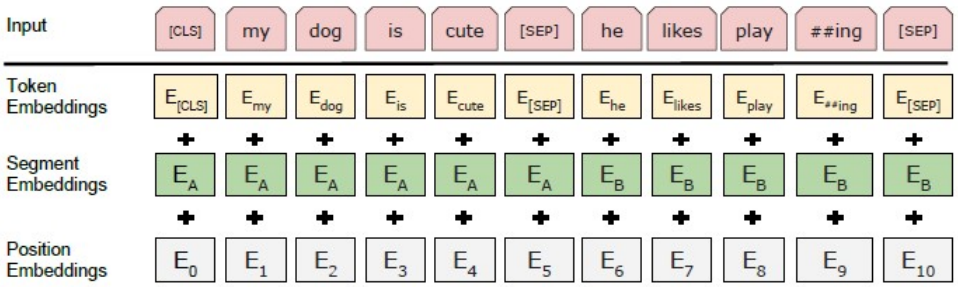


Figura 3.3: Formato de datos de entrada del modelo BERT.

3.3.2.2. Pre-entrenamiento

A diferencia de otros modelos como ELMo [34] en donde el pre-entrenamiento consiste en generar modelos de lenguaje de izquierda a derecha o de derecha a izquierda, BERT se pre-entrena en dos tareas no supervisadas: modelo de lenguaje enmascarado y predicción de la oración siguiente. Para el pre-entrenamiento del primer modelo BERT en inglés se utilizaron como datos de entrenamiento el BookCorpus que contiene aproximadamente 800M de palabras y el extracto del contenido de Wikipedia en inglés que contiene aproximadamente 2,500M de palabras.

- **Modelo de Lenguaje Enmascarado.** La tarea de modelado de lenguaje o "Language Modeling (LM)" en inglés, consiste en desarrollar un modelo estadístico capaz de predecir la siguiente palabra en una oración, dado un conjunto de palabras que la proceden. Los modelos de lenguaje tradicionales solo pueden ser entrenados de izquierda a derecha o de derecha a izquierda dado que un enfoque bidireccional permitiría que cada palabra se "vea" a si misma indirectamente.

  Para poder entrenar una representación bidireccional, el modelo BERT enmascara (oculta) un porcentaje de las palabras de entrada de manera aleatoria y después busca predecir esas palabras ocultas. El proceso de enmascaramiento ocurre después de la tokenización inicial de la oración de entrada descrita en la sección 3.3.2.1, por lo que el modelo no enmascara palabras completas sino los símbolos resultantes. Según el artículo de BERT escrito Devlin et. al (2018) [12], en cada oración de entrada, se enmascaran el 15 % de los símbolos de manera aleatoria, donde el 80 % de las veces se cambia el símbolo de entrada por el símbolo [MASK], el 10 % de las veces se reemplaza el símbolo original por otro aleatorio y el 10 % de las veces restantes se conserva el mismo símbolo de entrada sin cambios. La razón por la cual no se reemplaza el símbolo de entrada por el símbolo [MASK] el 100 % de la veces es para compensar el hecho de que, durante la fase de sintonización fina, el modelo nunca se entrara dicho símbolo y esto afectaría el desempeño general del modelo.

- **Predicción de la Oración Siguiente.** La segunda tarea para la cual es pre-entrenado el modelo BERT se conoce como "Predicción de la Oración Siguiente" o "Next Sentence Prediction (NSP)" en inglés. Muchas tareas específicas del procesamiento de lenguaje natural como la de responder preguntas (Question Answering) o la inferencia de lenguaje natural (NLI) están basadas en el entendimiento de la relación entre dos oraciones, lo cual no es capturado directamente por el modelado del lenguaje. Es por ello que, para poder entrenar un modelo que entienda relaciones entre oraciones, también se pre-entrena el modelo BERT para realizar la tarea de predecir si una oración B es la sucesora de una oración A, esta tarea es binaria (respuesta sí o no) y puede ser entrenada con cualquier texto que este en el mismo idioma.

Es importante mencionar que el pre-entrenamiento del modelo BERT es una tarea computacionalmente muy intensa, para la cual se requiere de un equipo con alto poder

computacional. Según la información reportada en el artículo de BERT [12], el pre-entrenamiento del modelo BERT-base se realizó utilizando 4 TPUs en la nube (16 chips TPUS en total) y el pre-entrenamiento del modelo BERT-grande utilizando 16 TPUs en la nube (64 chips TPUS en total), ambos tardaron 4 días en completarse.

3.3.2.3. Sintonización fina

La sintonización fina de un modelo BERT es un proceso directo en donde solo se introduce un conjunto de datos de entrenamiento con las entradas y salidas específicas de la tarea a realizar y todos los paramentos internos de la red del modelo BERT se modifican para realizarla. Dentro del campo del aprendizaje profundo, se conoce como Aprendizaje Transferido a esta técnica donde se toman los pesos un modelo previamente entrenado y se modifican para realizar una tarea específica.

Es importante señalar que, a diferencia del pre-entrenamiento, esta tarea es computacionalmente menos demandante y no requiere que se proporcione un conjunto de datos tan grande como en el caso del pre-entrenamiento. Según la información presentada por Devlin et. al (2018) [12] en el artículo oficial, un modelo BERT pre-entrenado puede ser sintonizado en menos de una hora usando un TPU en la nube o en unas cuantas horas usando un GPU.

En resumen, la importancia del modelo BERT es que esta arquitectura nos permite descargar un modelo pre-entrenado durante días con grandes cantidades de texto no etiquetado y, haciendo uso del aprendizaje transferido, sintonizarlo finamente para realizar tareas específicas al introducirle al modelo una cantidad mucho menor de texto etiquetado y utilizando una computadora con GPU para actualizar los parámetros del modelo. Los pesos de varios modelos BERT pre-entrenados, así como los scripts de código necesarios para sintonizar un modelo en las diferentes tareas de procesamiento, pueden ser descargados del repositorio de Github del modelo BERT.

3.3.3. ALBERT

Los excelentes resultados obtenidos en el campo del procesamiento de lenguaje natural utilizando modelos de aprendizaje profundo donde se emplean redes neuronales pre-entrenadas, como en es el caso del modelo BERT [12], condujeron el campo de investigación al desarrollo de una variedad de modelos que siguen el mismo principio, pero que buscan mejorar los resultados obtenido realizando optimizaciones o combinaciones de modelos. Un ejemplo de estos nuevos modelos basados en el modelo BERT, es ALBERT [24] cuyo nombre proviene de la frase en ingles “A Lite BERT”, es decir, se trata de un modelo que redujo la cantidad de parámetros totales del modelo BERT, y además logró obtener aún mejores resultados que el modelo original en distintas tareas específicas dentro del área del procesamiento de lenguaje natural.

Al igual que en el caso del modelo BERT, también es posible encontrar el modelo ALBERT en diferentes configuraciones, siendo las principales variantes la cantidad de parámetros totales del modelo y el número de capas ocultas del modelo. Según Devlin

et. al (2018) [12] en el artículo original de ALBERT, la configuración "grande" de ALBERT comparada con la configuración "grande" de BERT tiene aproximadamente 18 veces menos parámetros y puede ser entrenada hasta 1.7 veces más rápido. Una comparación de los parámetros y arquitectura de las configuraciones principales de BERT y ALBERT puede verse en la figura 3.4.

Model		Parameters	Layers	Hidden	Embedding	Parameter-sharing
BERT	base	108M	12	768	768	False
	large	334M	24	1024	1024	False
ALBERT	base	12M	12	768	128	True
	large	18M	24	1024	128	True
	xlarge	60M	24	2048	128	True
	xxlarge	235M	12	4096	128	True

Figura 3.4: Comparativa entre número de parámetros y arquitectura de los modelos BERT y ALBERT.

Para poder lograr esta reducción de parámetros sin afectar significativamente el desempeño, el modelo ALBERT incorpora dos técnicas de reducción de parámetros, cuyos nombres originales en inglés son: "factorized embedding parameterization" y "cross-layer parameter sharing". La primera técnica consiste en descomponer la matriz original de vocabulario del modelo BERT en dos matrices más pequeñas, eliminando así la dependencia que obliga una igualdad de tamaño entre la capas ocultas del modelo y el vocabulario del mismo, esto permitió incrementar el tamaño de las capas ocultas de la red sin necesidad de incrementar también la cantidad de parámetros que contiene el vocabulario. La segunda técnica de reducción de parámetros, "cross-layer parameter sharing", consiste en compartir todos los parámetros del modelo a través de la red neuronal. Esta técnica fue usada para prevenir que los parámetros del modelo crezcan junto con la profundidad de la red.

El modelo ALBERT también cambia la tarea de "predicción de la siguiente oración" (NSP por sus siglas en inglés "Next Sentence Prediction") realizada durante pre-entrenamiento del modelo BERT original, por una tarea llamada "Predicción del orden de la oración" ("Sentence Order Prediction (SOP)" en inglés). La tarea SOP se enfoca principalmente en detectar coherencia entre las oraciones de un texto y está diseñada para mejorar las deficiencias que presentaba la tarea de NSP. Como resultado de implementar este cambio se obtuvo un incremento significativo en el desempeño del modelo al sintonizarlo para resolver tareas específicas.

En el momento en que fue presentado, el modelo ALBERT obtuvo nuevos resultados estado del arte en las referencias GLUE, RACE y SQuAD, teniendo significativamente menos parámetros que el modelo BERT Grande. En específico, Lan et. al (2019) [24] reportan que se impulsó la exactitud del RACE a 89.4 %, el puntaje GLUE a 89.4 y el puntaje F1 en el conjunto de datos SQuAD, utilizado para evaluar tareas de responder a preguntas, a 92.2 %.

3.3.4. Modelos multi-idioma

El éxito de los modelos BERT [12] para la resolución de tareas de procesamiento de lenguaje natural trajo consigo la aparición de distintos modelos pre-entrenados en distintos idiomas diferentes al inglés como es el caso de BERT en español (BETO) [7], BERT en francés [27] y BERT en finlandés [45]. Sin embargo, la necesidad de contar con modelos BERT entrenados en más idiomas y el hecho de que la tarea de pre-entrenamiento es muy costosa computacionalmente, sugiere la alternativa de utilizar modelos multi-idioma, es decir, modelos BERT entrenados con un corpus y vocabulario adaptado para que el modelo aprenda distintos idiomas durante su fase de pre-entrenamiento.

La viabilidad de los modelos multi-idioma basados en BERT ha sido ampliamente estudiada debido la necesidad que se tiene de realizar tareas de procesamiento de lenguaje natural en idiomas distintos al inglés y que cuentan con menos recursos digitales para su entrenamiento. Junto con el modelo original de BERT entrenado en el idioma inglés presentado por Devlin et. al (2018) [12] también se presentó un modelo entrenado en varios idiomas, este modelo BERT multi-idioma fue entrenado utilizando extractos de Wikipedia en 104 idiomas y trajo consigo diversos estudios. Por ejemplo, Pires et. al (2019) [35] presentan un estudio empírico sobre las capacidades de BERT multi-idioma para resolver tareas en un idioma específico después de haber sido sintonizado en otro idioma distinto. Los resultados obtenidos en algunas tareas como reconocimiento de entidad nombrada (NER) y etiquetado de parte del discurso (POS) fueron sorprendentemente buenos y se ven reflejados en la figura 3.5.

Fine-tuning \ Eval	EN	DE	NL	ES
EN	**90.70**	69.74	77.36	73.59
DE	73.83	**82.00**	76.25	70.03
NL	65.46	65.68	**89.86**	72.10
ES	65.38	59.40	64.39	**87.18**

NER F1 results on the CoNLL data.

Fine-tuning \ Eval	EN	DE	ES	IT
EN	**96.82**	89.40	85.91	91.60
DE	83.99	**93.99**	86.32	88.39
ES	81.64	88.87	**96.71**	93.71
IT	86.79	87.82	91.28	**98.11**

POS accuracy on a subset of UD languages.

Figura 3.5: Resultados del modelo BERT multi-idioma en las tareas de NER y POS, entrenado y evaluado en distintos idiomas.

3.3.4.1. XLM

En el año 2019 surgieron diferentes arquitecturas de modelos así como propuestas de optimizaciones con la idea de mejorar el desempeño de los modelos multi-idioma, ejemplo de lo anterior es el trabajo de Lample y Conneau (2019) [23] donde presentan el modelo llamado XLM el cual, al igual que BERT [12], está basado en redes Transformers pero está adaptado para aprender representaciones entre idiomas. El modelo XLM puede ser pre-entrenado de forma no supervisada o supervisada.

Para el caso del pre-entrenamiento no supervisado, el modelo se entrena para resolver la misma tarea de Modelado de Lenguaje Enmascarado (MLM) utilizada en el modelo BERT original, con la diferencia de usar secuencias de texto en lugar de pares de oraciones como entrada del modelo. El pre-entrenamiento supervisado por su parte busca resolver una nueva tarea de entrenamiento propuesta por los autores y conocida como TLM o "Translation Language Modeling" en inglés. En la tarea de TLM se le introducen al modelo pares de oraciones en dos idiomas, para el modelo original de XLM se consideraron el inglés y el francés, y a continuación el modelo enmascara de forma aleatoria palabras en cualquiera de los dos idiomas. Para predecir las palabras enmascaradas, el modelo puede hacer uso de los dos idiomas, es decir, si la palabra enmascarada esta inglés, el modelo puede observar tanto las palabras en inglés que rodean a la palabra enmascarada o su traducción en francés. Las tareas de MLM y TLM del modelo XLM puede verse en la figura 3.6.

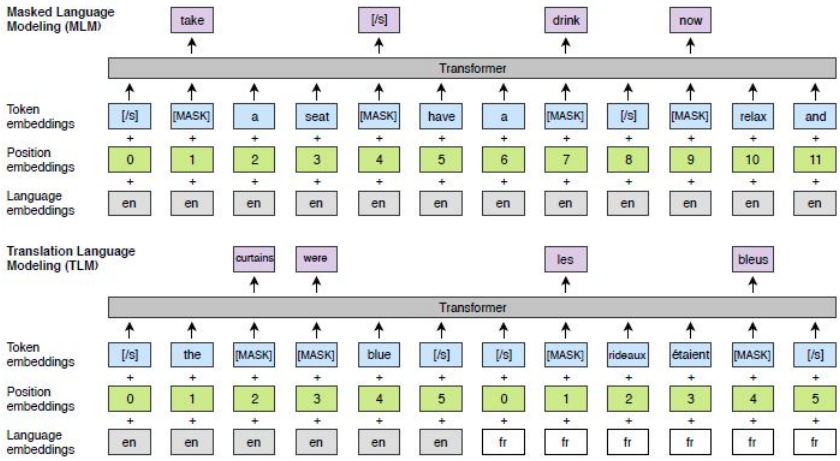


Figura 3.6: Tareas de pre-entrenamiento del modelo XLM.

El modelo XLM fue el primero en demostrar el impacto del modelado entre idiomas al obtener una ganancia en exactitud de 4.9 % en el conjunto de datos XNLI (Cross-lingual Natural Language Inference) usado para evaluar la capacidad de los modelos de aprender múltiples idiomas. Ademas también se obtuvieron ganancias significativas en el puntaje BLUE en diversos conjuntos de datos utilizados para evaluar la tarea de traducción automática.

3.3.4.2. XLM-RoBERTa

XLM-R es uno de los más recientes modelos multi-idioma y es una versión mejorada del modelo XLM [23] desarrollada por Lample y Conneau (2019) del equipo de Facebook AI. La letra R en el nombre del modelo es por que incorpora elementos de otro modelo basado en Transformers llamado RoBERTa[25], el cual es una versión optimizada del modelo BERT. El modelo RoBERTa busca mejorar el desempeño de BERT original al realizar algunas modificaciones, entre ellas, entrenar el modelo durante más tiempo, usando lotes de entrada más grandes y usar más datos de entrada; remover el objetivo de pre-entrenamiento de NSP (Next Sentence Prediction); entrenar en secuencias más largas y cambiar dinámicamente el patrón de enmascaramiento aplicado a los datos de entrenamiento.

XLM-R fue entrenado en 100 idiomas utilizado un vocabulario de 250 mil palabras (pudiendo ser palabras completas o solo partes de la palabra), a diferencia del vocabulario usado en BERT el cual consistía de una longitud de 32 mil palabras. El conjunto de datos de entrenamiento fue un corpus de la página "CommonCrawl" en 100 idiomas cuyo espacio total de almacenamiento se estima estar alrededor de los 2.5 TB. Además el modelo es entrenado al igual que el modelo RoBERTa, removiendo la tarea de NSP como objetivo de pre-entrenamiento y solo entrenando para la tarea de MLM [9].

XLM-RoBERTa supera los puntajes obtenidos por el modelo BERT multi-idioma en tareas como clasificación entre lenguajes por 21 % de exactitud en lenguajes con poca cantidad de recursos de entrenamiento. Además superó el mejor puntaje obtenido hasta esa fecha en el conjunto de datos XNLI por 3.9 % de exactitud y también supero el puntaje F1 en la tarea de "responder-preguntas entre idiomas" (MLQA) por un promedio de 8.4 %.

Capítulo 4

Desarrollo del proyecto

En esta sección se presentaran los detalles del diseño y funcionalidad del proyecto. La plataforma propuesta para apoyar a personas con ligeras pérdidas de memoria sera un asistente virtual, de forma más específica, un sistema de dialogo orientado a tareas. Ejemplos comerciales de este tipo de sistemas son "Assistant" de Google, "Alexa" de Amazon y "Siri" de Apple. La razón por la cual se decidió elaborar el proyecto bajo este esquema es debido a su facilidad y naturalidad de interacción con el usuario. Aunque la idea de recuperar los recuerdos de una personas es una tarea muy compleja, en este proyecto se propone un enfoque que les permitirá a la personas con problemas de perdida de memoria la posibilidad de recuperar información de eventos pasados mediante un sistema de preguntas-respuestas.

Los problemas de pérdida de memoria son cada vez más comunes en la población mundial debido al incremento en la esperanza de vida promedio del ser humano. Como se mencionó previamente, si bien los problemas de memoria no son específicos de personas de la tercera edad, el riesgo de presentar un problema de este tipo si aumenta conforme lo hace la edad, esto se debe a la gran cantidad de enfermedades y lesiones que pueden causar pérdida de memoria. Si bien algunos casos de amnesia son temporales y no afectan la capacidad de las personas para llevar a cabo sus actividades diarias, existen otros problemas, como es el caso de la demencia, que tienen un efecto profundo sobre las capacidades cognitivas de las persona y le impiden llevar una vida plena e independiente, además de afectar también la vida de sus familiares cercanos debido al grado de dependencia que se desarrolla naturalmente.

A lo largo de esta década, ha habido una gran cantidad de avances en el campo de la inteligencia artificial, gracias al uso de algoritmos inspirados en el funcionamiento cerebro humano conocidos como redes neuronales artificiales. Estos nuevos sistemas neuronales han permitido la elaboración de diversos productos con la capacidad de asistir a los seres humanos en diversas tareas complejas, como por ejemplo la detección de enfermedades en radiografías, mejoras significativas en la traducción automática de textos, manejo autónomo de vehículos, entre otras. Por ello surge la cuestión sobre si es posible generar un producto, que utilice técnicas de inteligencia artificial para asistir a personas con problemas de pérdida de memoria, y que a su vez facilite la labor de

sus cuidadores y familiares cercanos.

4.1. Metodología

En este proyecto de tesis se buscó diseñar un sistema de asistencia para personas con problemas de memoria leves, que sea de uso sencillo y que permita utilizar los últimos avances en el campo de inteligencia artificial para asistirlos en la tarea de almacenar nuevos recuerdos, ya que por lo general la amnesia no afecta la memoria a largo plazo sino la capacidad de retener nuevos recuerdos.

Para el diseño de este sistema de asistencia se tomó como inspiración el éxito que han tenido los asistentes virtuales comerciales multitareas creado por grandes empresas tecnológicas como Google y Amazon. Estos asistentes son de fácil uso ya que no hacen uso de una interfaz gráfica de usuario y solo funcionan por comandos de voz, además permiten que los usuarios interactúen con ellos de forma muy intuitiva mediante diálogos, de los cuales el asistente virtual extrae información valiosa para determinar la solicitud del usuario, así como información extra que se requiera para completar dicha solicitud. A este tipo de sistemas también se les conoce como agentes conversacionales o chatbots.

El primer paso del desarrollo fue investigar los elementos que componen a un agente conversacional, así como las diversas herramientas que se encuentran disponibles para la creación e implementación de los mismos. La elección de la herramienta a utilizar para el desarrollo del agente conversacional presentado en este trabajo se realizó principalmente con base en tres criterios: el costo, las limitaciones de la herramienta y por último, que dicha herramienta permitiera crear productos que entiendan y respondan en el idioma español. Este último criterio se consideró de gran importancia ya que, además del hecho de que el desarrollo de este proyecto se realizó en un país de habla hispana, parte importante del trabajo de investigación es adaptar nuevas tecnologías de procesamiento de lenguaje natural al idioma español. Es importante recalcar lo último, ya que gran parte de la investigación en este campo se realiza solamente en el idioma inglés.

Bajo los criterios previamente mencionados, se eligió la herramienta Dialogflow de Google para la creación del agente conversacional. A continuación se procedió a diseñar la interacción deseada entre el agente conversacional y el usuario, para lo cual se definieron los escenarios que el asistente de memoria identificará y con base en ello, le responderá al usuario. Estos escenarios predefinidos tienen el propósito de asistir a los usuarios con problemas leves de memoria a recordar detalles de eventos pasados, y para llevar a cabo esta tarea se emplearon diversas tecnologías y servicios.

Lo que se busca es que el agente conversacional pueda responder correctamente a las preguntas que los usuarios tengan sobre hechos o detalles de eventos pasados en su vida, para ello habría que definir escenarios en el agente conversacional para cada posible pregunta que el usuario tenga, lo cual sería imposible, ya que no hay forma de predecir ni lo que el usuario preguntara, ni las respuestas a dichas preguntas. Para resolver este problema se propuso adaptar el agente conversacional con dos componentes

principales: en primer lugar, una base de datos donde se almacenan los recuerdos del usuario, los cuales podrán ser guardados por un cuidador o incluso por el usuario mismo; en segundo lugar, un modelo de inteligencia artificial que buscara las respuestas a las preguntas del usuario en todos los recuerdos almacenados.

Para la implementación de la base de datos se utilizaron servicios en la nube, debido a las ventajas que ofrecen como lo es el fácil acceso a los mismos y la rápida lectura de datos en tiempo real. Dado que el agente conversacional fue creado con un servicio de la empresa Google (Dialogflow), para la base de datos se decidió utilizar a esta misma empresa como proveedor de servicios y así, mantener en lo posible todo el sistema funcionando bajo el mismo ecosistema de servicios en la nube. El servicio de base de datos utilizado fue Firestore, de la plataforma de desarrollo Firebase.

La tarea de responder a preguntas o QA (Question Answering), es una tarea compleja estudiada en el campo del procesamiento de lenguaje natural, la cual involucra la creación de sistemas computacionales que, dados como entradas un párrafo de contexto y una pregunta, el sistema entregue en su salida una repuesta a dicha pregunta extrayendo la información del párrafo de entrada. La correcta resolución de esta tarea implica que el sistema computacional puede entender el contexto semántico de la pregunta y buscar una respuesta acorde en el párrafo de entrada.

Existen diversos conjuntos de datos disponibles para entrenar y evaluar sistemas en la resolución de la tarea de QA, siendo el conjunto de datos de nombre SQuAD (Stanford Question Answering Dataset) posiblemente el más popular y el más usando. Este conjunto de datos está compuesto de cientos de párrafos extraídos de Wikipedia correspondientes a una amplia variedad de temas, además cada párrafo viene acompañado de preguntas y respuestas sobre el contenido del mismo. Para este proyecto se utilizó el conjunto de datos SQuAD como base para entrenar el modelo inteligente que se incorporó al agente conversacional, ya que además de ser usado comúnmente como referencia para tareas de QA, la estructura del conjunto de datos se adapta perfectamente a las necesidades del proyecto, donde en escenario práctico, el párrafo de contexto estaría conformado por todos los recuerdos almacenados del paciente y la pregunta sería aquella que se le haga al asistente de memoria.

Los últimos avances en el campo del procesamiento de lenguaje natural utilizando técnicas de aprendizaje profundo (subrama de la inteligencia artificial), han permitido la creación de sistemas inteligentes capaces de resolver esta tarea de forma satisfactoria, siendo esta un área de investigación muy activa en los últimos años. En específico, para este proyecto se estudiaron modelos de aprendizaje profundo basados en arquitecturas de redes Transformer (modelos BERT), los cuales revolucionaron la forma de procesar diversas tareas clásicas del procesamiento de lenguaje natural, sin ser los sistemas QA una excepción.

Para este proyecto de tesis se entrenaron diversos modelos BERT para resolver la tarea de QA y se compararon sus puntajes F1, el cual es un indicador del desempeño del modelo. Los principales retos que surgieron en esta parte del desarrollo fueron, el hecho de encontrar modelos que entregaran un buen puntaje F1 en el idioma español, y la necesidad de adaptar herramientas y datos, normalmente utilizados para desarrollar

agentes inteligentes en el idioma inglés, para funcionar en el idioma español.

Finalmente, para la integración total de las partes principales del asistente de memoria se utilizaron otros servicios en la nube, siendo la mayoría de Google Cloud, los cuales permitieron interconectar el modelo de aprendizaje profundo con el agente conversacional y la base de datos. El hecho de vincular todos los componentes del sistema entre si bajo este esquema, permitió la obtención de un producto final capaz de entender y atender correctamente las solicitudes de los usuarios de forma rápida, natural y funcional en el idioma español.

4.2. Diseño general

El objetivo principal de este proyecto es implementar un agente conversacional (Chatbot) que, mediante comandos de voz, interactúe con el usuario y pueda asistirlo almacenando recuerdos, leyéndole recuerdos de días pasados y contestando a preguntas específicas sobre dichos recuerdos. Para el desarrollo del agente conversacional se propone el uso de una plataforma de procesamiento de lenguaje natural capaz de entender las acciones que el usuario desee ejecutar en determinado momento, dichas plataformas fueron introducidas en la sección 3.2. Además, el asistente de memoria necesita contar con un sistema capaz de contestar a las distintas preguntas que pueda tener el usuario relacionadas con sus recuerdos, para ello se propone el uso de modelos de aprendizaje profundo debido a los excelentes resultados que se han obtenido con modelos recientes, esto se explica con detalle en la sección 4.4. Finalmente el agente conversacional necesita de una infraestructura donde residan todos los servicios computacionales que lo componen así como una interfaz de usuario, estos temas será revisado en las secciones 4.5 y 4.6 respectivamente. En la figura 4.1 se representa un ejemplo del funcionamiento esperado del asistente, así como los distintos módulos implicados en dicha interacción. En este ejemplo el usuario hace una pregunta al agente conversacional, el cual, después de revisar una base de datos que contiene todos los recuerdos previamente almacenados del usuario, le da una respuesta basada en la información almacenada.

4.3. Plataforma NLP para creación de Chatbots

Para la elaboración de este proyecto se decidió utilizar la plataforma de desarrollo Dialogflow de Google. Esta herramienta no solo es gratuita y tiene soporte para el idioma español, también tiene varias funcionalidades que facilitan el desarrollo de agentes conversacionales (chatbots) y que otras plataformas no tienen, por ejemplo, el uso de “contexto” para conversaciones más largas. El contexto es una herramienta muy útil para los agentes conversaciones ya que, de forma general, les permite conocer si la conversación con el usuario sigue en torno a un tema específico y así poder concatenar distintas intenciones. Un ejemplo común del uso del contexto es para la implementación de una intención de confirmación, donde se le solicita al usuario verificar si la

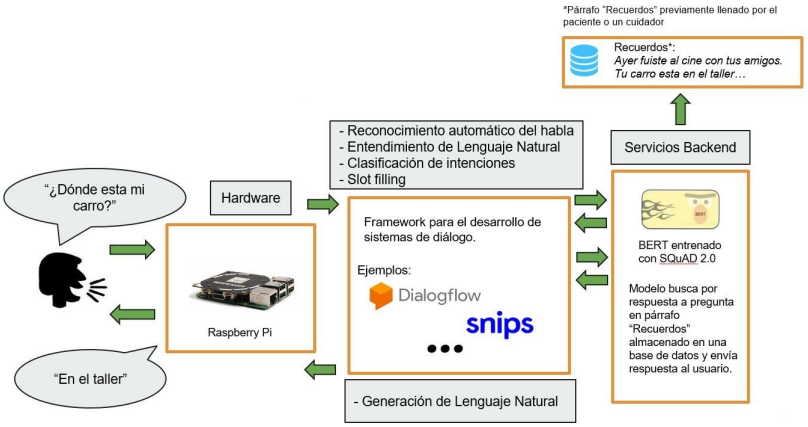


Figura 4.1: Diagrama general representando la interacción deseada entre el asistente y el usuario.

información capturada en otra intención fue correcta.

Dialogflow también cuenta con una interfaz web amigable, la cual permite entrenar agentes conversaciones de forma intuitiva al introducir ejemplos de frases de entrenamiento, las cuales el agente deberá relacionar con intenciones específicas. Además dentro de esta interfaz también es posible etiquetar los *slots* de la intención, especificando que tipo de entidades residieran en dichos *slots*. La plataforma cuenta con varias entidades pre-definidas y utilizadas comúnmente como fechas, lugares, número, entre otras, sin embargo también permite la posibilidad de definir entidades personalizadas proporcionando ejemplos de palabras que caen bajo la categoría de dicha entidad.

Otras de las ventajas que ofrece Dialogflow es la capacidad de integrar los agentes conversacionales creados en otros servicios, permitiendo así una implementación rápida de los mismos. Además, al ser parte de los servicios de Google, se puede realizar integrar fácilmente con otros servicios en la nube como bases de datos y sistemas de analíticas. En general la plataforma de Dialogflow ofrece todas las herramientas necesarias para la elaboración de este proyecto de forma eficiente y gratuita.

El trabajo con Dialogflow se inició definiendo diversos *"intents"* o escenarios predefinidos que serán la base de interacción con el usuario. Estos *intents* estarán orientados a ayudar al usuario a recordar detalles y eventos pasados o a almacenar recuerdos. Para entrenar al agente conversacional y este pueda identificar correctamente cada escenario predefinido, se le deben proporcionar una serie de frases ejemplo, junto con los *slots* (en caso de haber) que debe de llenar para completar la solicitud. Además, se le pueden indicar una serie de respuestas definidas o que debe darle al usuario, o en su defecto indicar qué acciones debe realizar una vez se haya completado la solicitud. Para el desarrollo de este asistente de memoria se definieron definieron *intents* para almacenar recuerdos, escuchar recuerdos, contestar a una pregunta, entre otros. Un ejemplo de las frases de entrenamiento utilizadas para el escenario de leer recuerdos, así como la interfaz de usuarios de Dialogflow, pueden observarse en la figura 4.2.

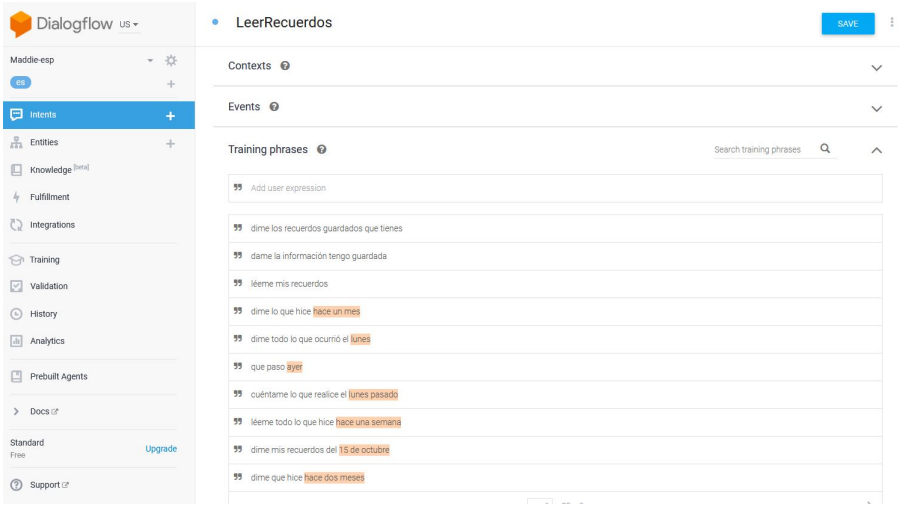


Figura 4.2: Interfaz de usuario de Dialogflow donde se observan ejemplos de frases de entrenamiento para un intent.

4.4. Sistema de Preguntas-Respuestas (QA)

Utilizando como infraestructura del asistente virtual uno de los frameworks mencionados en la sección 3.2, es posible crear un sistema de dialogo que mantenga una conversación con cualquier persona. Sin embargo, para este proyecto se busca hacer un asistente que apoye a personas con problemas leves de memoria por lo que es necesario enfocar las funciones del asistente en ayudar a personas con esta condición. En esta sección se describirá el desarrollo de la función más importante que tendrá el agente conversacional y lo que lo diferenciara de otros productos similares, y es la capacidad de responder a cualquier pregunta que un usuario tenga sobre hechos pasados en su vida diaria.

Lo primero que se realizo fue identificar el tipo de problema a resolver como un problema de responder a preguntas o también llamado "Question Answering" (QA) en inglés. Los últimos avances en aprendizaje profundo han demostrado obtener resultados sobresalientes en distintas tareas de procesamiento de lenguaje natural, ejemplo de ello es el modelo conocido como BERT (Bidirectional Encoder Representations from Transformers). BERT es un modelo desarrollado por Google que presentó resultados sobresalientes en distintas tareas de procesamiento de lenguaje natural como son el análisis de sentimientos, clasificación de oraciones y QA. El modelo BERT es considerado como un hito en el procesamiento de lenguaje natural dado que una sola arquitectura puede ser entrenada en distintos idiomas y para realizar distintas tareas. Una explicación detallada del modelo BERT se encuentra en la sección 3.3.2.

4.4.1. Conjunto de datos SQuAD

La resolución de problemas de QA es una tarea común en el procesamiento de lenguaje natural y para ello existen varios conjuntos de datos (o también llamados "datasets" en inglés) disponibles en internet, sin embargo la mayoría de ellos se encuentran en el idioma inglés. Uno de estos conjuntos de datos que es muy conocido y es considerado como punto de referencia al momento de evaluar un sistema de QA es el SQuAD que significa Stanford Question Answering Dataset [38]. Como su nombre lo indica, este conjunto de datos fue elaborado por la universidad de Stanford y es utilizado en tareas de comprensión de lectura, actualmente existen dos versiones del mismo, la versión 1.1 y la 2.0. La versión 1.1 consta de un conjunto de alrededor de 100,000 preguntas elaboradas por colaboradores de la universidad sobre un conjunto de artículos de Wikipedia, en este conjunto de datos la respuesta a cada pregunta es un segmente de texto dentro del párrafo de lectura correspondiente. La versión 2.0 combina las 100,000 preguntas de la versión 1.1 con más de 50,000 preguntas sin respuesta escritas también por colaboradores de empresa y de una forma muy similar a las preguntas con respuesta [37]. Para que un sistema pueda obtener un buen desempeño en este conjunto de datos no solo debe responder a las preguntas cuando sea posible, también debe ser capaz de determinar cuando no es posible contestar una pregunta con la información del párrafo y abstenerse de contestar.

Un ejemplo de un párrafo con sus respectivas preguntas del conjunto de datos SQuAD 2.0 se puede observar en la figura 4.3

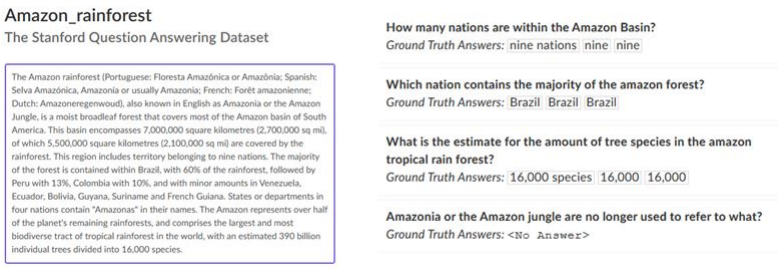
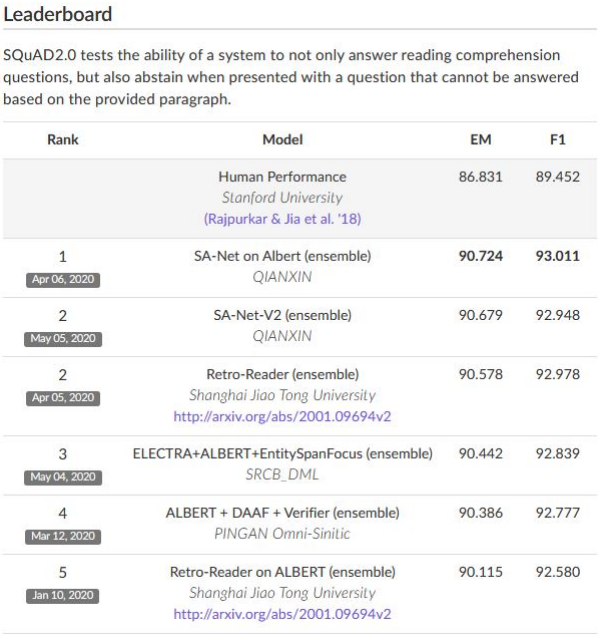


Figura 4.3: Ejemplo de párrafo y preguntas en SQuAD 2.0

Utilizar un modelo que pueda resolver el dataset SQuAD 2.0 de forma eficiente es necesario para la correcta ejecución del asistente virtual por lo que se realizaron distintas pruebas para elegir el modelo que otorgue mejores resultados. Es importante mencionar que, en todas las tareas de procesamiento de lenguaje natural, el idioma del texto a procesar es un factor importante a considerar por lo que, para este proyecto, se iniciaron pruebas en el idioma inglés para verificar la viabilidad del mismo.

En la página oficial del conjunto de datos SQuAD es posible conocer los modelos que han obtenido mejores resultados en la resolución del conjunto de datos tanto para la versión 1.1 como para la versión 2.0, esta información es plasmada en una "tabla de líderes" y una captura de la misma puede verse en la figura 4.4. En dicha tabla

se coloca el nombre del modelo que resolvió el conjunto de datos junto al puntaje F1 obtenido en el mismo, esto servirá de guía en el proyecto para conocer los modelos de los cuales se pueden esperar mejores resultados. Es importante destacar que para el conjunto SQuAD, el desempeño del ser humano no está considerado con un puntaje F1 de 100, para la versión 1.1 se considera al ser humano con un puntaje F1 de 91.22 y para la versión 2.0 un puntaje de 89.45.



Leaderboard

SQuAD2.0 tests the ability of a system to not only answer reading comprehension questions, but also abstain when presented with a question that cannot be answered based on the provided paragraph.

Rank	Model	EM	F1
	Human Performance *Stanford University* (Rajpurkar & Jia et al. '18)	86.831	89.452
1 Apr 06, 2020	SA-Net on Albert (ensemble) *QIANXIN*	**90.724**	**93.011**
2 May 05, 2020	SA-Net-V2 (ensemble) *QIANXIN*	90.679	92.948
2 Apr 05, 2020	Retro-Reader (ensemble) *Shanghai Jiao Tong University* http://arxiv.org/abs/2001.09694v2	90.578	92.978
3 May 04, 2020	ELECTRA+ALBERT+EntitySpanFocus (ensemble) *SRCB_DML*	90.442	92.839
4 Mar 12, 2020	ALBERT + DAAF + Verifier (ensemble) *PINGAN Omni-Sinitic*	90.386	92.777
5 Jan 10, 2020	Retro-Reader on ALBERT (ensemble) *Shanghai Jiao Tong University* http://arxiv.org/abs/2001.09694v2	90.115	92.580

Figura 4.4: Captura de tabla de líderes de conjunto de datos SQuAD 2.0 en Mayo 2020

4.4.2. BERT Inglés

El modelo BERT es una parte importante de la ejecución de esta intención y es por ello que se deben considerar algunos factores. El entrenamiento del modelo BERT consta de dos partes. La primera es un "pre-entrenamiento" en el cual el sistema aprende la tarea de entendimiento de lenguaje natural (NLU) en un idioma específico, la segunda es una sintonización fina del modelo para aprender a realizar una tarea de procesamiento de lenguaje natural en específico.

La fase de pre-entrenamiento de BERT es tardada, según se reporta en la documentación del modelo, la empresa Google tardó tres días utilizando TPUs para pre-entrenar el primer modelo de BERT en inglés. Es por ello que en la documentación oficial del proyecto se encuentran disponibles para descargar varios modelos BERT pre-entrenados. Desafortunadamente no existe un modelo de BERT pre-entrenado solamente en español, sin embargo si existe un modelo pre-entrenado con múltiples idiomas donde se incluye el español. Es por esto y dado que la mayoría de los conjuntos de datos para QA

(incluido el SQuAD) solamente se encuentran disponibles en ingles que se decidió iniciar las primeras pruebas del asistente virtual en inglés y una vez obtenidos resultados, si demuestran ser satisfactorios, se buscara pre-entrenar un modelo BERT en español y con realizar la sintonización fina con un conjunto de datos en el mismo idioma.

Para la fase de sintonización-fina en tareas de QA del modelo BERT es importante considerar que, este modelo fue diseñado considerado el conjunto de datos SQuAD como referencia, dado que es un punto de referencia en tareas de QA, es por ello que si se desea sintonizar el modelo para resolver tareas de QA, el conjunto de datos debe ser el SQuAD o un conjunto de datos que tenga el mismo formato que el SQuAD. Para el caso específico del agente conversacional que se está desarrollando en este proyecto, la idea es introducir al sistema una estructura similar a la del SQuAD donde primero se introduciría un párrafo de contexto con todos los recuerdos del usuario previamente almacenados en una base de datos, y en segundo lugar se introduciría la pregunta que se desea que el agente conteste. Por ello, para este proyecto, el proceso de realizar la sintonización fina del modelo para tareas de QA se simplifica en gran medida.

Con el propósito de probar el diseño de la función principal del asistente, se entrenó primero un modelo BERT para resolver la tarea de QA usando el conjunto de datos SQuAD 1.1 y el modelo pre-entrenado en ingles proporcionado en la documentación oficial del proyecto. En la figura 4.5 se muestran algunos de los resultados obtenidos con el modelo BERT así como su puntaje F1, el cual es la métrica utilizada para evaluar los modelos en el conjunto de datos SQuAD.

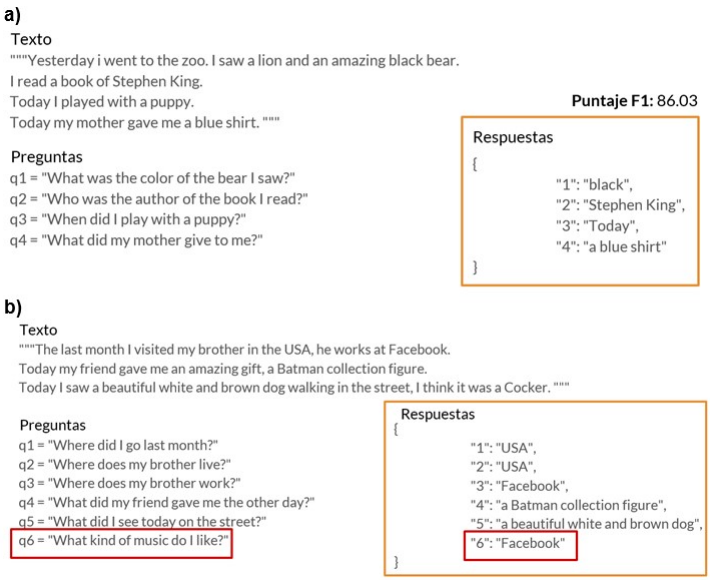


Figura 4.5: Resultados de modelo BERT entrenado en el conjunto de datos SQuAD 1.1

Como se puede observar en la figura 4.5 el modelo, al ser entrenado con el conjunto de datos SQuAD 1.1, no sabe cómo lidiar con preguntas para las que no existe una respuesta explicita dentro del párrafo por lo que simplemente responde con la respuesta

que considera más probable. Para este proyecto es importante que el asistente aprenda a identificar cuando no cuenta con información suficiente para darle respuesta a una pregunta del usuario, ya que es muy posible que la persona que utilice el asistente pregunte algo de lo cual el asistente no tenga información. Es por lo anterior que se entrenó otro modelo BERT con el conjunto de datos SQuAD 2.0. Un ejemplo de los resultados usando dicho modelo así como la puntuación F1 del mismo pueden observarse en la figura 4.6.

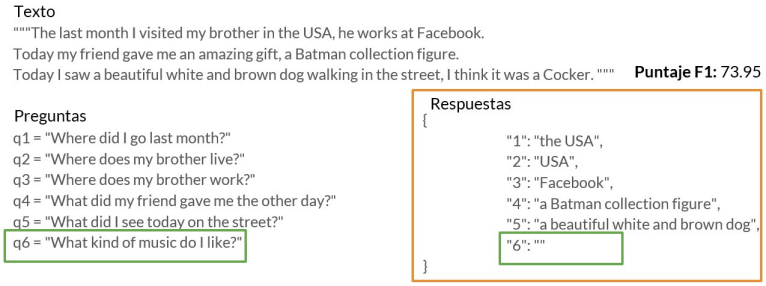


Figura 4.6: Resultados de modelo BERT entrenado en el conjunto de datos SQuAD 2.0

Como se puede observar en la figura 4.6, aunque el puntaje F1 del modelo bajo a 73.95, es decir aproximadamente 13 puntos menos en comparación al puntaje obtenido con el mismo modelo entrenado con el conjunto de datos SQuAD 1.1, el modelo logró responder a las mismas preguntas correctamente e incluso logró identificar que para el caso de la pregunta 6 no contaba con información suficiente en el texto y por ello se abstuvo de proporcionar una respuesta.

4.4.3. ALBERT Inglés

Los resultados obtenidos con el modelo BERT fueron satisfactorios y se consideraron como suficientes para resolver la tarea de QA que debe llevar acabo el asistente virtual, sin embargo, desde el lanzamiento del modelo BERT en el año 2018 comenzaron a surgir otras arquitecturas basadas en el modelo original pero que, debido a sus distintas modificaciones y optimizaciones, permiten alcanzar un mejor desempeño. Para el caso particular del conjunto de datos SQuAD 2.0, es posible visitar la página oficial del conjunto de datos y ahí conocer los modelos que mejores resultados han tenido en la resolución del conjunto de datos.

Con el propósito de mejorar tanto la precisión como la velocidad del sistema de QA del asistente, se decidió implementar un modelo, variación del BERT y cuyos resultados demuestran ser de los mejores obtenidos para la resolución del conjunto de datos SQuAD 2.0, esta variación del modelo se llama ALBERT y la teoría del mismo fue detallada en la sección 3.3.3. De forma general, el nombre ALBERT viene de “A lite BERT” y como su nombre lo implica, este modelo es una versión del modelo BERT con una arquitectura que tiene significativamente menos parámetros que el modelo original pero que presenta un mejor desempeño. Al igual que en el caso del modelo BERT,

también es posible encontrar el modelo ALBERT en diferentes configuraciones, siendo las principales variantes la cantidad de parámetros totales del modelo y el número de capas ocultas del modelo. Según la información del articulo original de ALBERT [24], la configuración "grande" de ALBERT comparada con la configuración "grande" de BERT tiene aproximadamente 18 veces menos parámetros y puede ser entrenada hasta 1.7 veces más rápido.

Es importante recordar que en esta fase del proyecto se continuaron haciendo pruebas de los alcances del sistema en el idioma inglés, ya que es un idioma mejor documentado y con una mayor cantidad de conjuntos de datos que el español, y de no alcanzar resultados satisfactorios en inglés no se puede esperar tener mejores resultados en español. Similar al modelo BERT, el modelo ALBERT tiene tanto una fase de pre-entrenamiento, donde el modelo aprende la tarea de modelado de lenguaje en un idioma específico, como una fase de sintonización-fina donde el modelo se entrena para una tarea específica.

Para la fase de pre-entrenamiento es posible descargar un modelo ALBERT-base pre-entrenado en el idioma inglés desde el repositorio oficial del modelo de Github. Para la fase de sintonización-fina se utilizará nuevamente el conjunto de datos SQuAD 2.0, ya que como se explicó previamente, es necesario que el sistema final pueda identificar las preguntas para las cuales no tiene información suficiente para dar una respuesta. Los resultados obtenidos del entrenamiento utilizando la librería de aprendizaje profundo TensorFlow, después de dos repeticiones (también conocidas como epochs), pueden ser observados en la figura 4.7, donde es importante notar que el puntaje F1 incremento a 78.91, es decir casi 5 puntos más que en el modelo BERT normal. Además el archivo resultante que contiene los pesos del modelo resulto ser mucho más ligero que el obtenido entrenando el modelo BERT original, en el modelo ALBERT el archivo pesó 156 MB y en el caso del modelo BERT original pesó 1.98 GB.

```
=================================================
exact = 76.24863134843763
f1 = 78.91188093037756
null_score_diff_threshold = -0.4245270788669586
total = 11873
best perf happened at step: 10000
```

Figura 4.7: Resultados de modelo ALBERT entrenado en el conjunto de datos SQuAD 2.0

4.4.4. BERT multi-idioma

Los resultados obtenidos con los modelos en el idioma inglés fueron considerados como satisfactorios entonces se procedió a implementar el modelo en el idioma español, para ello se decidió primero implementar el modelo en BERT original, sin embargo surgen dos necesidades. La primera es contar un modelo BERT pre-entrenado en el

idioma español y la segunda es contar con un conjunto de datos similar al SQuAD pero en idioma español.

Debido al éxito de los modelos BERT, comenzaron a surgir diversas propuestas para poder utilizar estos modelos en idiomas distintos al inglés. Pre-entrenar un modelo BERT en un idioma específico no es una tarea sencilla, y principalmente se necesitan dos cosas: un conjunto de datos y un equipo computacional capaz de realizar tareas muy demandantes. El conjunto de datos debe ser una gran cantidad de texto no etiquetado en el idioma objetivo, para lo cual puede usarse el corpus de Wikipedia en español que contiene aproximadamente 120 millones de palabras y que cuya descarga es gratuita. Sin embargo el verdadero problema radica en el pre-entrenamiento del modelo en sí, ya que según el artículo oficial de BERT, el equipo de Google entrenó el modelo durante 4 días utilizando 4 TPUs para el modelo base y 16 para el modelo grande. Es por lo anterior que, como alternativa para implementar un modelo BERT en idiomas distintos al inglés, BERT multi-idioma fue lanzado junto con el modelo BERT original en inglés. Este modelo fue entrenado en 104 idiomas, incluido el español, y es la forma más rápida de implementar un modelo BERT en idiomas distintos al inglés.

Para la realizar la sintonización fina del modelo BERT en la tarea de QA en español, es necesario utilizar un conjunto de datos que este en dicho idioma y además que este en el mismo formato que el conjunto de datos SQuAD, ya que es el tipo de estructura que el modelo espera. Dado que fabricar un conjunto de datos desde el principio sería una tarea muy laboriosa y además no se necesita un conjunto de datos con un dominio específico (dado que el asistente virtual solo busca responder cualquier tipo de pregunta dado un párrafo de contexto), la opción que se consideró más eficiente fue traducir el conjunto de datos SQuAD 2.0 con ayuda de una herramienta de traducción automática.

Pio Carrino et. al (2019) [8] presentan en su artículo un método de traducción automática para el conjunto de datos SQuAD al idioma español. Para este proyecto resultó de gran ayuda el resultado obtenido, ya que independiente del método de traducción, los autores del artículo publican un versión traducida del conjunto de datos SQuAD que puede ser utilizada para sintonizar el modelo BERT en tareas de QA en español.

Una vez que se han obtenido tanto el modelo pre-entrenado como el conjunto de datos para la tarea de sintonización fina, es posible continuar con el entrenamiento del modelo. Para ello se utilizaron las librerías PyTorch y Transformers de Huggingface [47] debido a las facilidades que ofrece tanto para el entrenamiento de modelos basados en BERT como para la utilización de dichos modelos una vez entrenados. El modelo se entrenó inicialmente durante dos epochs (misma cantidad que se utilizó en el caso de los modelos BERT y ALBERT in inglés) y usando la versión "case", es decir respetando mayúsculas y signos de puntuación, ya que es la versión recomendada por Google. Los resultados finales del pre-entrenamiento incluido el puntaje F1 del modelo se pueden observar en la figura 4.8.

Como se puede observar, el puntaje F1 fue de 66.94 lo cual se consideró como normal dado que no se esperaba obtener mejores resultados que en el caso del modelo entrenado solamente en el idioma inglés. Por ello, se decidió incrementar la cantidad

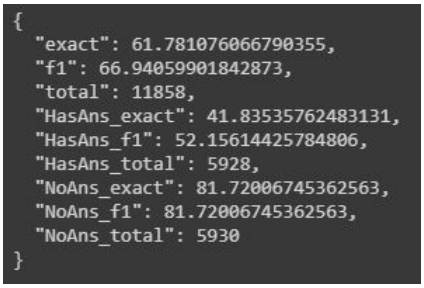


Figura 4.8: Resultados del modelo BERT multi-idioma entrenado en el conjunto de datos SQuAD 2.0 por 2 epochs

de epochs de entrenamiento a cinco y verificar si existía una mejora en el desempeño del modelo. Los resultados del entrenamiento con 5 epochs pueden verse en la figura 4.9. Analizando las figuras se puede observar es que sucedió lo contrario a lo que se esperaba y el puntaje F1 disminuyo. Este comportamiento es indicador de un posible sobre ajuste del modelo.

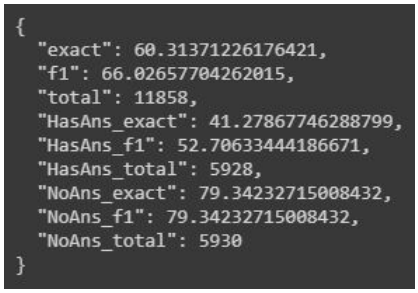


Figura 4.9: Resultados del modelo BERT multi-idioma entrenado en el conjunto de datos SQuAD 2.0 por 5 epochs

4.4.5. XLM-RoBERTa

Los modelos multi-idioma son ampliamente estudiados actualmente en el área del aprendizaje profundo y por ello, se tienen registros de modelos multi-lenguaje que, para algunas tareas de procesamiento de lenguaje natural en idiomas diferentes al inglés, han obtenido mejores resultados que modelos entrenados en un solo idioma, tal es el caso del modelo XLM-RoBERTa.

El modelo XLM-RoBERTa [11] es una variación del modelo BERT original pero con la variante de haber sido entrenado con mucho mayores cantidades de texto, se estima que el almacenamiento necesario para guardar todo el corpus de entrenamiento fue de 2.5 TB. Además, el modelo XLM-RoBERTa fue pre-entrenado en más de 100 idiomas y solo para la tarea de MLM (modelado de lenguaje enmascarado o “Masked

Language Model" en inglés) siendo esta una diferencia significativa hacia el modelo BERT original donde además se pre-entrena el modelo para resolver la tarea de NSP (predicción de oración siguiente o "Next Sentence Prediction" en inglés).

De manera general el modelo XLM-RoBERTa reporta un incremento en los puntajes de distintas tareas de procesamiento de lenguaje natural comparado con el modelo BERT multi-idioma, así como un incremento en puntajes para tareas en lenguajes con pocos recursos. Por ello se decidió verificar de manera empírica si para el caso del idioma español y para la tarea específica de QA se logra tener un mejor desempeño que un modelo entrenado exclusivamente en el idioma español.

Para la sintonización-fina del modelo se utilizaron nuevamente las librerías PyTorch y Transformers. La librería Transformers de Huggingface no solo ofrece una interfaz sencilla para entrenar y utilizar modelos basados en BERT, además ofrece la posibilidad de descargar directamente los pesos de varios modelos pre-entrenados estando el modelo XLM-RoBERTa entre ellos. Al igual que en los casos anteriores, se realizó la sintonización-fina del modelo durante 2 epochs y 5 epochs para comprar los resultados contra los modelos previamente entrenados. Los resultados pueden verse en las figuras 4.10 y 4.11 respectivamente, donde puede observarse que el modelo no presenta una mejora significativa en su desempeño al incrementar la cantidad de epochs de entrenamiento de 2 a 5, además el puntaje F1 alcanzado es muy similar al obtenido en la sección 4.4.4.


```
{
  "exact": 59.87518974531962,
  "f1": 64.75717251383817,
  "total": 11858,
  "HasAns_exact": 40.587044534412954,
  "HasAns_f1": 50.352657164151516,
  "HasAns_total": 5928,
  "NoAns_exact": 79.15682967959528,
  "NoAns_f1": 79.15682967959528,
  "NoAns_total": 5930
}
```


Figura 4.10: Resultados del modelo XLM-RoBERTa entrenado en el conjunto de datos SQuAD 2.0 por 2 epochs

4.4.6. BERT Español (BETO)

Se tiene el objetivo de implementar un modelo BERT para resolver la tarea de QA en el idioma español con el objetivo de integrar esta funcionalidad al asistente de memoria, para ello primero se realizaron pruebas del desempeño de un modelo BERT en el idioma inglés y a continuación se buscó mejorar el puntaje F1 en la tarea de QA al cambiar el modelo BERT a un modelo ALBERT, el cual demostró tener un desempeño muy superior al modelo BERT original. Sin embargo, dado que el asistente de memoria que se presenta en este proyecto se desea funcione en el idioma español, se buscaron formas de implementar un modelo de QA que funcione en dicho idioma. Dado que el pre-entrenamiento de un modelo BERT es una tarea compleja que requiere de un equipo

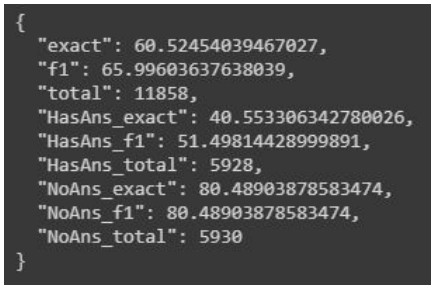


Figura 4.11: Resultados del modelo XLM-RoBERTa entrenado en el conjunto de datos SQuAD 2.0 por 5 epochs

con un alto poder computacional, se buscó la alternativa de utilizar modelos multi-idioma. Para ello, utilizando una versión traducida al español del conjunto de datos SQuAD 2.0, se realizaron pruebas utilizando el modelo BERT multi-idioma y después se hicieron pruebas utilizando el modelo XLM-RoBERTa, sin embargo ambos modelos presentaron resultados muy similares donde el puntaje F1 es de aproximadamente 66. Desafortunadamente, al momento de escribir este trabajo, el modelo ALBERT aún no cuenta con versión multi-idioma que se puedan sintonizar para realizar la tarea de QA en español, por ello se decidió realizar como prueba final la sintonización de un modelo BERT pre-entrenado solamente en el idioma español.

Cañete et. al (2020) [7] de la universidad de Chile, realizaron la labor de pre-entrenar un modelo BERT en español debido a la necesidad contar con un modelo en dicho idioma dado que es uno de los cinco idiomas más hablados en el mundo. La arquitectura de BERT en español, también conocido como BETO, es similar a la del modelo BERT original. Cuenta con 12 capas de auto-atención, 16 cabezas de atención y 1024 capas ocultas, teniendo un total de 110M de parámetros.

BETO fue pre-entrenado utilizando varios corpus de texto no etiquetado entre los cuales estaban el de Wikipedia y textos del proyecto OPUS (Open Parallel Corpus), el cual es una larga colección gratuita de textos traducidos de la web entre los que se encuentran artículos de las naciones unidas, platicas de TED, subtítulos, etc. El tamaño total del corpus utilizado para el entrenamiento de BETO se compara con el de BERT original alcanzado aproximadamente 3 billones de palabras. Similar al modelo BERT original, existen dos versiones de BETO, una versión donde el modelo fue pre-entrenado respetando mayúsculas y acentos también conocida como versión “cased” en inglés, y una versión en donde se removieron acentos y todo el texto se encuentra en minúsculas también llamada “uncased”. Además, también se construyó un vocabulario para el proceso de tokenización con aproximadamente 32 mil símbolos.

Los pesos pre-entrenados del modelo BERT en español están disponibles tanto para ser usados tanto en Tensorflow como en PyTorch. Al igual que en el caso de los modelos multi-idioma, se decidió utilizar los pesos preparados para PyTorch junto con la librería Transformers de Huggingface debido a las facilidades que ofrece tanto para el entrenamiento de modelos basados en BERT como para la utilización de dichos modelos una

vez entrenados. Así mismo, para realizar la sintonización fina del modelo BETO en la tarea de QA en español, se utilizara el conjunto de datos SQuAD traducido al español por Pio Carrino et. al (2019) [8]. Los resultados finales del entrenamiento por 2 epochs, incluido el puntaje F1 del modelo se pueden observar en la figura 4.12.


```
{
  "exact": 62.37139483892731,
  "f1": 68.2312064054464,
  "total": 11858,
  "HasAns_exact": 44.095816464237515,
  "HasAns_f1": 55.81741659173052,
  "HasAns_total": 5928,
  "NoAns_exact": 80.64080944350759,
  "NoAns_f1": 80.64080944350759,
  "NoAns_total": 5930
}
```


Figura 4.12: Resultados de modelo BERT-Español (BETO) entrenado en el conjunto de datos SQuAD 2.0 por 2 epochs

Como se puede observar, el puntaje F1 fue de 68.23 lo cual se consideró como satisfactorio ya que es el puntaje más alto obtenido en la tarea de QA en español a lo largo de este proyecto. Por ello, se decidió incrementar la cantidad de epochs de entrenamiento a cinco y verificar si existía una mejora significativa en el puntaje F1 del modelo. Los resultados de dicho entrenamiento puede verse en la figura 4.13 donde se puede observar que el puntaje F1 fue de 67.45, es decir, menor que el en entrenamiento con solo 2 epochs. Al igual que en el caso del entrenamiento del modelo BERT multi-idioma, este comportamiento nos indica un posible sobre ajuste del modelo.


```
{
  "exact": 60.94619666048237,
  "f1": 67.45181369772293,
  "total": 11858,
  "HasAns_exact": 43.16801619433198,
  "HasAns_f1": 56.1814451463553,
  "HasAns_total": 5928,
  "NoAns_exact": 78.71838111298483,
  "NoAns_f1": 78.71838111298483,
  "NoAns_total": 5930
}
```


Figura 4.13: Resultados de modelo BERT-Español (BETO) entrenado en el conjunto de datos SQuAD 2.0 por 5 epochs

4.5. Servicios de Computación en la Nube

En la sección anterior se describió el proceso para desarrollar un modelo capaz de resolver la tarea de QA en el idioma español con el mejor desempeño posible. En

esta sección se presentara la forma de implementar dicho modelo para que el asistente virtual pueda hacer uso del mismo, además se implementaran otros servicios necesario para este proyecto, como es el caso de la base de datos donde se almacenaran todos los recuerdos del usuario y que después serán utilizados por el agente conversacional para responder preguntas.

4.5.1. Google Cloud - App Engine

Para la elaboración de este proyecto lo que se busca es servir el modelo en una plataforma como un servicio REST, y así poder acceder a él por medio del agente conversacional creado con la herramienta Dialogflow. Los servicios que tienen una arquitectura REST también son conocidos como RESTful API y son un tipo de servicios web que permiten la comunicación entre servicios computacionales por el internet. Las interfaces RESTful utilizan el protocolo HTTP para obtener, publicar, actualizar y eliminar información mediante métodos con los nombres de GET, POST, PUT y DELETE respectivamente.

Existen diferentes formas de implementar un modelo de aprendizaje profundo para realizar predicciones en tiempo real. La forma tradicional seria utilizando una librería como Flask de Python y servir el modelo un servidor, este enfoque es perfectamente válido, sin embargo implica la necesidad de adquirir un servidor dedicado para servir el modelo, brindarle servicios de mantenimiento, configurar la red local para que permita accesos desde el Internet, entre otras cosas. Los servicios en la nube surgen como una solución para simplificar este tipo de situaciones. Actualmente existen una gran variedad de servicios en la nube siendo los principales proveedores grandes empresas como Google y Amazon. Para el desarrollo de este proyecto se propuso mantener, en la medida de lo posible, todos los servicios de la nube con el mismo proveedor, ya que esto facilitaría la integración de los mismos. Dado que la plataforma a utilizar para crear el agente conversacional se decidió que sería Dialogflow de la empresa Google, entonces se decidió utilizar los servicios de la nube de Google que están agrupados bajo la marca de Google Cloud.

Para poder servir el modelo de aprendizaje profundo previamente entrenado se decidió utilizar el servicio de App Engine de Google Cloud [18]. App Engine es una plataforma de cómputo en la nube que permite el desarrollo y almacenamiento (“hosting”) de aplicaciones web. Una de las principales ventajas de la plataforma App Engine es que ofrece la posibilidad de auto-escalamiento, es decir, que la cantidad de recursos computacionales que se le asignarán a la aplicación para su ejecución serán proporcionales a la cantidad de usuarios que la estén haciendo uso de la misma. El auto-escalamiento es de mucha utilidad si se desea poner el sistema en un habiente de producción, ya que permite reducir los costos del uso del servició en la nube y al mismo tiempo, se cuida que los usuarios siempre tengan una experiencia de uso cómoda al no experimentar retrasos durante la comunicación con el servidor.

El servicio de App Engine ofrece dos posibles opciones de ambiente: estándar y flexible. El ambiente estándar se basa en uso de instancias pre-definidas, donde el usua-

rio puede elegir una de varias posibles configuraciones pero no puede modificar dicha configuración. Las instancias se dividen en clases, donde cada clase determina la cantidad de memoria RAM y CPU disponibles para cada instancia, así como los costos por usar el servicio (se incluyen tanto costos por hora como una cantidad definida de horas consideradas como cuota gratuita donde no se cobraran costos por usar el servicio). El ambiente estándar de App Engine tiene soporte para los siguientes lenguajes de programación: Python, Java, Node.js, PHP, Ruby y Go.

El ambiente flexible de App Engine, a diferencia del ambiente estándar, permite a los desarrolladores personalizar el tipo de instancia bajo la cual se ejecutara su servicio o aplicación web. Entre las opciones de configuración se encuentra la cantidad de memoria y CPU que la aplicación necesita, además es posible acceder a la instancia mediante protocolo SSH y tiene soporte nativo para integrar en la instancia micro servicios, autorización, bases de datos relacionales y no relaciones, entre otros. Los lenguajes de programación que soporta el ambiente flexible de App Engine de forma nativa son: Go, Java 8, PHP 5/7, Python 2.7/3.6, .NET, Node.js, Ruby.

Para este proyecto se utilizó el ambiente flexible dado que la cantidad de memoria máxima que ofrece el entorno estándar (2048 MB), no fue suficiente para servir el modelo de aprendizaje profundo. Para desplegar un servicio web en App Engine es necesario la creación de principalmente tres archivos: el archivo de configuración del entorno de ejecución, el archivo de requerimientos del servicio web y el archivo donde se ejecuta el programa principal.

En el archivo de configuración se agregan los requerimientos del entorno de ejecución, ahí se deben especificar datos como cantidad de núcleos a utilizar, cantidad de memoria RAM, espacio de disco duro, entre otros. El archivo debe guardarse en la carpeta principal del proyecto bajo el nombre de "app.yaml". En la figura 4.14 se muestra el archivo de configuración utilizado para servir el modelo BERT utilizado en este proyecto, donde se especifica que se utilizara un ambiente flexible, donde además se utilizara el lenguaje de programación Python en su versión 3.6 (solo es necesario especificar el número de la versión mayor a utilizar ya sea 2 para la versión 2.7 o 3 para la versión 3.6), el entorno deberá contar con 4GB de memoria RAM, 15GB de espacio de almacenamiento y se deberán usar 2 núcleos de procesamiento, además se especifica que la clase de instancia será tipo F (lo cual habilita el auto-escalamiento) y que se utilizará un servidor Gunicorn para administrar y ejecutar la aplicación web.

El siguiente archivo a configurar para desplegar una aplicación web en App Engine es el archivo de requerimientos, en dicho archivo se listan las librerías necesarias para que la aplicación funcione correctamente, junto con el número de versión a utilizar, y el servidor de App Engine se encarga de instalar dichas dependencias, este archivo debe guardarse bajo el nombre de "requirements.txt" En la figura 4.15 se muestra el archivo de configuración utilizada para implementar el modelo BERT en App Engine, donde además de listar las librerías utilizadas para servir aplicaciones web en Python (Flask y Gunicorn), se especifica que deben instalarse las librerías PyTorch y Transformes de Huggingface.

El último archivo se compone del programa principal, este archivo deberá guardarse

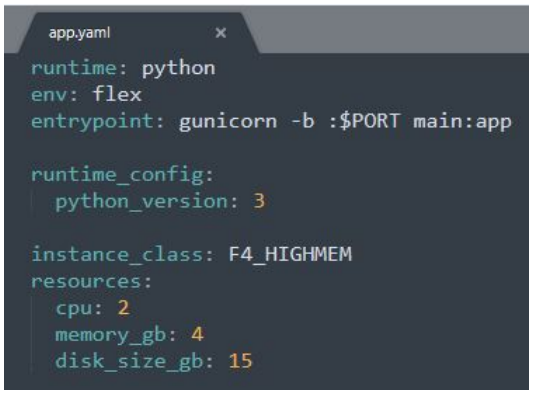


Figura 4.14: Archivo de configuración "app.yaml" utilizado en el proyecto.

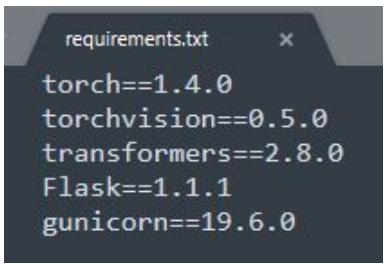


Figura 4.15: Archivo de requerimientos "requirement.txt" utilizado en el proyecto.

bajo el nombre de "main.py" y debera estar estructurado bajo el esquema del framework Flask, es decir, se deberán especificar las rutas y los métodos HTTP bajo los cuales la aplicación responderá a los usuarios. Una vez configurados los tres archivos previamente mencionados, para desplegar la aplicación en los servidores de Google es necesario descargar la CLI (Command Line Interface) de Google Cloud en cualquier computadora, posicionarse en la carpeta principal del proyecto donde residen los archivos "main.py", "requirements.txt" y "app.yaml", y ejecutar en la consola de comandos la instrucción de despliegue indicada en la documentación de App Engine (al momento de realizar este proyecto fue "gcloud app deploy") [18].

4.5.2. Firebase - Firestore

En las secciones anteriores, se explicó el procedimiento para generar un modelo de aprendizaje profundo capaz de resolver la tarea de QA en el idioma español y además, se explicó la propuesta para implementar dicho modelo como una interfaz RESTful utilizando el servicio en la nube App Engine de Google Cloud y así, poder realizar inferencias en tiempo real. Es importante recordar que el modelo de QA previamente mencionado necesita como entradas un párrafo de contexto y una pregunta a contestar, una vez proporcionados dichos datos, el modelo tratará de responder la pregunta con

base a la información entregada en el párrafo de contexto. La propuesta en este proyecto es que la pregunta provenga del usuario y el párrafo de contexto este compuesto por todos los recuerdos del usuario previamente almacenados en una base de datos. En esta sección se explicarán las características de la base de datos a utilizar así como su integración en el proyecto.

Actualmente existen principalmente dos tipos de bases datos: las bases de datos relacionales, también conocidas como bases de datos SQL, y las no relaciones, también conocidas como bases de datos NoSQL. Las bases de datos relacionales hacen uso de tablas y esquemas, es decir la información es almacenada de una forma estructurada y se utiliza el lenguaje SQL (Structured Query Language) para realizar consultas sobre dicha información. Dado que la información almacenada dentro de la base de datos debe tener una estructura específica, es una práctica común tener varias tablas con distinta estructura dentro de una misma bases de datos y, en caso de necesitar información que esté presente en ambas tablas, unirlas mediante comandos SQL.

Las bases de datos no relacionales o NoSQL difieren de las bases de datos relacionales en varios aspecto, siendo el principal que estas no utilizan el lenguaje SQL como método principal para realizar consultas, sin embargo, también pueden soportar dicho lenguaje. Las bases de datos NoSQL utilizan un esquema fijo para almacenar información, es decir, no almacenan la información en tablas con una estructura definida como en el caso de las bases de datos SQL y por ello normalmente no soportan las operaciones de unión de tablas y tampoco garantizan que la información no se repita ni que sea consistente. Sin embargo, a pesar de los inconvenientes que puede presentar una base de datos NoSQL, también ofrece muchas ventajas que han sido la causa de su popularidad en empresas que manejan grandes cantidades de datos. Las principales ventajas de las bases de datos no relacionales son que, al no ser estructuradas permiten mayor flexibilidad al almacenar la información, además no son centralizadas por lo que también permiten una mayor escalabilidad horizontal, es decir es posible mejorar el rendimiento y espacio de almacenamiento del sistema si se agregan más nodos. Por lo anterior las bases de datos no relacionales son muy efectivas para manejar grandes cantidades de datos, además son más rápidas que las bases de datos SQL al realizar operaciones de lectura o escritura en un solo campo.

Al igual que para el caso de servir un modelo de aprendizaje profundo detallado en la sección 4.5.1, para implementar una base de datos, ya sea relacional o no relacional, existen dos principales opciones, la primera opciones es donde se adquiere y configura un servidor con el software necesario para administrar la base de datos, además de realizar las configuraciones en la red local necesarias para su funcionamiento y seguridad; y la segunda utilizar un servicio en la nube.

Actualmente existen varios sistemas para administrar bases de datos, tanto relacionales como no relacionales, y varios de ellos cuentan con la posibilidad de usar sus servicios en la nube, sin embargo es importante recordar que una de las premisas de este proyecto es la de usar, en la medida de lo posible, los servicios de un mismo proveedor, en este caso Google Cloud. Por ello para integrar en el proyecto una base de datos se decidió utilizar Cloud Firestore [20].

Cloud Firestore es un servicio incluido en Google Firebase, una plataforma para el desarrollo de aplicaciones web y móviles que pertenece al conjunto de servicios que ofrece Google para desarrolladores. La plataforma Firebase es un conjunto de herramientas usadas para construir y mejorar aplicaciones, estas herramientas son varios servicios que son comúnmente parte de la infraestructura de la aplicación y que, al ser facilitados, permiten a los desarrolladores enfocarse directamente en el desarrollo de la aplicación en sí, y no en la implementación de dichos servicios. Entre los servicios que incluye Firebase se encuentran los servicios de analíticas, autenticación de usuarios, base de datos, almacenamiento de archivos, web hosting, entre otros.

Firestore es el servicio de base de datos de Firebase y es una base de datos NoSQL, flexible y escalable adaptada para el desarrollo de aplicaciones web y móviles. Entre los principales beneficios de utilizar Firebase está la posibilidad de mantener los datos sincronizados a través de todos los dispositivos conectados a la base de datos por medio de "listeners", la capacidad de mantener las aplicaciones funcionales sin importar si presentan problemas de conectividad a Internet gracias a que ofrece soporte offline, la posibilidad de realizar lecturas y escrituras rápidas en la base de datos y además ofrece una integración sencilla con otros servicios de Firebase y Google Cloud, siendo esto último de gran importancia ya que la base de datos debe funcionar en conjunto con el agente elaborado en Dialogflow y el modelo de aprendizaje profundo implementado en App Engine.

Al ser un base de datos NoSQL, la información almacenada en Firestore no se guarda en tablas, como en el caso de la bases de datos SQL, sino en documentos. Cada documento puede contener distintos tipos de datos, los cuales se almacenan en el formato llave-valor. Los documentos a su vez se almacenan en contenedores llamados colecciones, las cuales permiten ordenar los documentos de forma más eficiente. Entre los distintos tipos de datos que pueden contener los documentos también están las colecciones mismas, lo cual permite que la base de datos crezca tanto como la aplicación lo necesite al crear una estructura jerárquica en forma de árbol.

Para este proyecto, la base de datos en Firestore inicia con una colección de todos los usuarios de la aplicación, donde cada usuario es un documento que entre sus valores tiene una colección de recuerdos, cada recuerdo es a su vez un documento con información como la fecha de registro del mismo y su texto correspondiente. En la figura 4.16 se muestra un ejemplo de la estructura de la base datos para el proyecto, la cual es alimentada por el asistente de memoria y de la cual se obtiene la información necesaria para contestar a las preguntas del usuario.

4.5.3. Firebase - Functions

En este punto del proyecto se tienen tres sistemas diferentes funcionando de manera independiente: el agente conversacional de Dialogflow, la base de datos en Firestore, y el modelo BERT desplegado en App Engine; sin embargo es necesario conectar estos sistemas para que funcionen en conjunto y se puedan completar las solicitudes del usuario. Para poder llevar a cabo esta tarea se hace uso de otro servicio en la nube

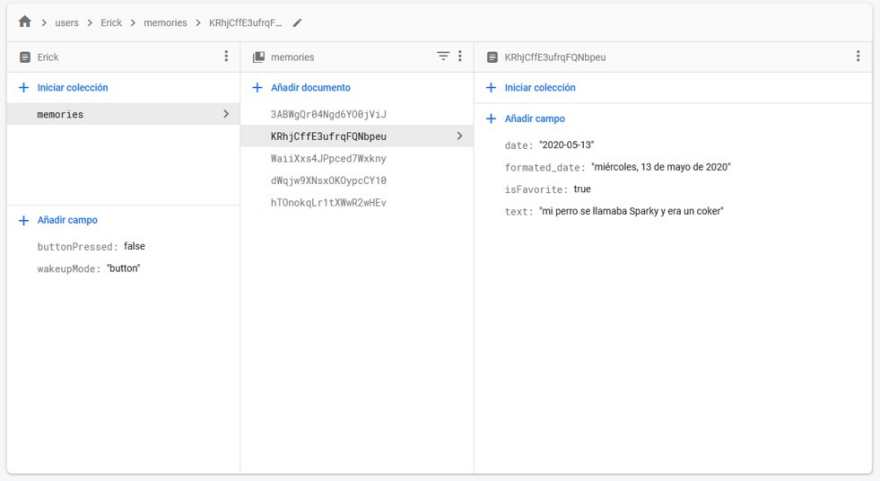


Figura 4.16: Información del asistente de memoria almacenada en la base de datos Firestore.

llamado Cloud Functions [21]. El servicio de Cloud Functions es otro servicio más que ofrece la empresa Google y que, junto con la base de datos utilizada para este proyecto, forma parte de los servicios de la plataforma Firebase. Este servicio permite a los desarrolladores almacenar y ejecutar fragmentos de código dentro de servidores en la nube, permitiendo así la creación de aplicaciones complejas al interconectar varios servicios de la nube tanto de Google Cloud como de otros proveedores.

Dentro de la ingeniería de software es común el hecho de separar un proyecto en dos partes: el frontend y el backend. De manera general, el frontend es la interfaz de usuario con la que interactúa el cliente y el backend es servidor donde reside la aplicación y los servicios que hacen funcionar a la misma como por ejemplo, una base de datos. Generalmente, cuando no se utilizan servicios en la nube, el desarrollador adquiere un servidor donde configura sus servicios de backend individualmente y además, con ayuda de un framework de desarrollo para backend, interconecta todos sus servicios. Es importante separar la lógica y el código backend del frontend, ya que normalmente mediante el código backend se accede a servicios que pueden tener información privada, como acceso a una base de datos, y se debe cuidar que los usuarios no tengan acceso a dicha información.

Para este proyecto se están utilizando servicios en la nube, es decir, no existe un solo servidor en donde estén ubicados todos los servicios de la aplicación (a este tipo de aplicaciones se conocen como “serverless”), y por ello no es posible implementar código de backend de la forma tradicional. Para solucionar dicha necesidad se utilizó Google Cloud Functions de Firebase, un framework “serverless” que permite ejecutar código de backend dentro de un servidor de la nube, en respuesta a eventos accionados por solicitudes HTTP o por algún otro servicio de la plataforma Firebase. Cloud Functions utiliza el lenguaje programacion Javascript o Typescript (una superclase de Javascript)

y utiliza como entorno de ejecución el ambiente Node.js, un popular ambiente de ejecución que permite correr código Javascript fuera de un navegador y que es comúnmente utilizado para programar código backend.

Para el caso específico de este proyecto, se utilizó Cloud Functions como medio para interconectar los servicios de base datos, el modelo de aprendizaje profundo servido en App Engine y al agente conversacional creado en Dialogflow. Dado que Dialogflow es el medio por el cual se va a interactuar con el usuario, se definieron dos funciones principales como respuesta a eventos accionados por Dialogflow. La primera función es una puerta de enlace por la cual se enviaran solicitudes HTTP cada vez que se desee conversar con el asistente de memoria, estas solicitudes pueden venir de cualquier medio definido en el frontend (por ejemplo una página web o una aplicación de Android) una vez se le hayan otorgado permisos para realizar dicha acción. La segunda función principal definida en Cloud Functions es la que se ejecuta cuando el asistente virtual necesita comunicarse con otros servicios para completar una acción (intención) solicitada por el usuario, este proceso es conocido como Fulfillment y es aquí donde un agente conversacional accede a otros servicios externos que puede ser APIs, otros servicios en la nube, bases de datos, entre otros. Para el caso específico de este proyecto es en la función de Fulfillment donde el agente conversacional accederá a la base de datos para leer o escribir recuerdos y también donde accede al modelo de aprendizaje profundo para responder a las preguntas que le hagan los usuarios. En la figura 4.17 se puede ver un diagrama de la interacción de los servicios backend en este proyecto.

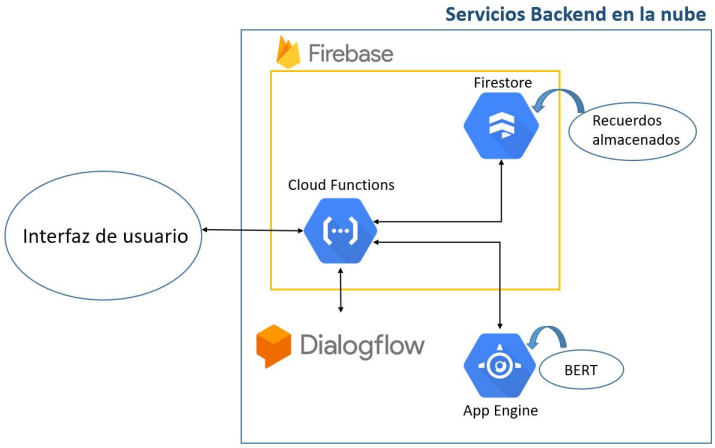


Figura 4.17: Diagrama de los servicios backend en la nube del asistente de memoria.

4.6. Interfaz de usuario

En la sección 4.4 se presentó el diseño y desarrollo del backend del proyecto, es decir de la parte de la aplicación que reside en servidores y donde se incluyen servicios

responsables del correcto funcionamiento de la aplicación como es el caso del acceso a la información que reside en una base de datos. En esta sección se explicarán los detalles sobre el diseño y desarrollo del frontend de la aplicación, es decir de la parte de la aplicación encargada de interactuar con el usuario.

Para este proyecto se proponen dos formas de interacción con el agente conversacional, la primera es mediante comandos de voz y con ayuda de un dispositivo compuesto de bocina y micrófono como interfaz, esta propuesta es motivada por el éxito de otros productos comerciales como es la caso de Amazon Echo y Google Home, dispositivos utilizados para interactuar con los asistentes virtuales Alexa de Amazon y Google Assistant de Google respectivamente. Dado que el público objetivo del agente conversacional presentado en este trabajo será, en su mayoría, personas de edad avanzada, es necesario implementar una forma de interacción sencilla y natural, por lo que se decidió el uso de comandos de voz como principal forma de interacción con el asistente de memoria.

La interacción por voz ofrece la posibilidad de interactuar con el agente conversacional de forma más natural, sin embargo, esta forma de interacción no siempre es óptima para realizar otras tareas y no siempre es la forma más rápida y eficiente de obtener información, por ejemplo, en el caso específico del asistente de memoria que se presenta en este trabajo, el sistema de comandos de voz es la forma más sencilla que tiene el paciente con problemas de memoria para comunicarse con el asistente sin embargo, el cuidador también tiene que comunicarse con el asistente de memoria para realizar acciones como registrar nuevos recuerdos o eliminar recuerdos de eventos simples que ya no son importantes para el paciente, para ello es más eficiente realizar dichas actividades desde un dispositivo móvil o una página web. Es por lo anterior que, además del dispositivo compuesto de bocina y micrófono, también se propone el desarrollo de una aplicación móvil para sistemas Android como interfaz con el agente conversacional. Es importante mencionar que tanto la aplicación para Android como el sistema micrófono/bocina estarán conectados al mismo backend del asistente de memoria en la nube, por lo que la información de los recuerdos almacenados será consistente, siendo la misma para ambos sistemas en todo momento.

4.6.1. Hardware

En esta sección del proyecto se describe el desarrollo de la interfaz de usuario del asistente de memoria, es decir, el medio por el cual los usuarios podrán comunicarse con el agente conversacional. Como se mencionó anteriormente, la principal forma de interacción será por medio de comandos de voz, emulando la forma de interacción de otros asistentes virtuales comerciales que existen actualmente. Para ello se implementará un dispositivo que sea capaz de capturar las solicitudes del usuario en forma de audio por medio de un micrófono, después, el dispositivo deberá convertir dichas solicitudes de audio a texto, enviar las solicitudes al servicio de backend del asistente de memoria descrito en la sección 4.5, y por último convertir las respuestas recibidas por el servicio backend de texto a voz.

Para poder implementar correctamente esta interfaz de usuario se decidió utilizar

una Raspberry Pi versión 3B+. La Raspberry Pi es una computadora de bajo costo y de tamaño reducido (aproximadamente el tamaño de una tarjeta de crédito), utilizada ampliamente en el desarrollo de proyectos en las áreas de electrónica, mecatrónica y computación. Existen una gran cantidad de periféricos que pueden conectarse a la Raspberry Pi, como pueden ser cámaras, pantallas, sensores, teclados, micrófonos, entre otros. Para este proyecto se utilizó una tarjeta Raspberry Pi modelo 3B+, la cual puede verse en la figura 4.18 y tiene las siguientes características:

- Procesador Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz
- 1GB LPDDR2 SDRAM
- 2.4GHz y 5GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2, BLE
- Gigabit Ethernet sobre USB 2.0 (max. 300 Mbps)
- 40-pin GPIO
- Full-size HDMI
- 4 puertos USB 2.0
- Puerto para Cámara CSI
- Puerto para Display DSI
- Salida 4-pole stereo
- Puerto Micro SD
- Entrada de alimentación de 5V/2.5A DC
- Soporte Power-over-Ethernet (PoE)

Es importante mencionar que, como cualquier computadora de uso normal, la Raspberry Pi necesita contar con un sistema operativo, el cual debe ser instalado en la tarjeta Micro SD. Aunque existen otros sistemas operativos creados por terceros, para este proyecto se utilizó el sistema operativo oficial de la tarjeta, el cual es se conoce con el nombre de Raspberry Pi OS (anteriormente conocido como Raspbian).

Además de la tarjeta Raspberry es necesario utilizar dos periféricos: un micrófono y una bocina. Como micrófono se decidió utilizar el ReSpeaker 4-Mic Array de la marca “Seeed”. Este arreglo de micrófonos está diseñado para ser utilizado en conjunto con la tarjeta Raspberry Pi y está pensando para el diseño de aplicaciones que dependan de comandos de voz, en la figura 4.19 pueden observarse las características del arreglo de micrófonos y en la figura 4.20 puede verse montado en la una tarjeta Raspberry Pi. El arreglo ReSpeaker cuenta con 4 micrófonos que le permiten capturar voces en un radio de hasta 3 metros, además cuenta con 12 LEDS programables para mejorar la interacción con los usuarios y emular el comportamiento de otros productos comerciales

Figura 4.18: Raspberry Pi Modelo 3B+.

como Amazon Echo y el Google Home. La instalación del ReSpeaker en la tarjeta Raspberry Pi está detallada en la documentación oficial en la página de Seeed [40], donde además se pueden encontrar ejemplos de código tanto para la captura de audio, como para la utilización de los LEDs.

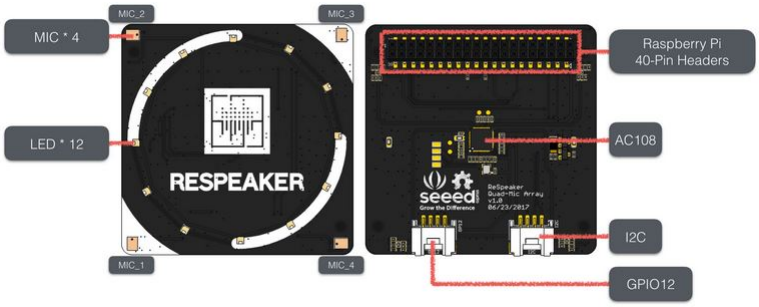


Figura 4.19: ReSpeaker 4-Mic Array.

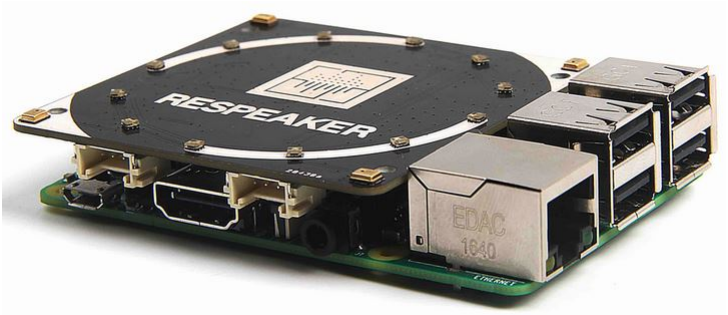


Figura 4.20: ReSpeaker 4-Mic Array montado en una tarjeta Raspberry Pi.

La bocina que se decidió utilizar para el proyecto es un altavoz portátil "Hamburger Mini Speaker" mostrado en la figura 4.21, el cual a pesar de ser pequeño ofrece una

buen volumen de audio y un aspecto elegante. El altavoz Hamburger Mini Speaker cuenta con una batería de litio recargable y ofrece un sonido de 360 grados.

Figura 4.21: Altavoz Hamburger Mini Speaker.

4.6.2. Interfaz de voz

En esta sección se explicará la implementación del sistema de voz del asistente de memoria, esto incluye el reconocimiento y procesamiento de voz proveniente del usuario, la comunicación entre frontend y backend de la aplicación, y por último la conversión de las respuestas del agente conversacional de texto a voz para ser escuchadas por el usuario.

La tarjeta Raspberry Pi permite la ejecución de scripts de código de la misma forma en como se realizaria en una computadora de escritorio, es por ello que se decidió elaborar un script en el lenguaje Python para realizar la conexión entre la Raspberry Pi y el agente conversacional implementado usando servicios en la nube (backend). Se decidió utilizar Python para desarrollar esta interfaz ya que es un lenguaje que ha obtenido mucha popularidad en los últimos años y por ende cuenta con una gran cantidad de librerías y un amplio soporte de la comunidad, lo cual facilita en gran medida el desarrollo de aplicaciones.

Existen varios servicios en la nube encargados de realizar la conversión de voz a texto (Speech Recognition), ejemplos de ello son el servicios Speech-to-Text de Google Cloud y Amazon Transcribe de Amazon Web Services. La utilización de cualquiera de estos servicios es un enfoque válido para implementar reconocimiento de voz en la aplicación, sin embargo, con el fin de reducir los tiempos de respuesta entre la interacción humano-computadora, se decidió no utilizar un servicio en la nube para realizar esta tarea y en su lugar utilizar la librería SpeechRecognition de Python. SpeechRecognition es una librería utilizada para realizar tareas de reconocimiento de voz que tiene soporte para

diferentes idiomas y diferentes APIs, tanto online como offline, algunas de las APIs que soporta son Google Speech Recognition, Google Cloud Speech API, Wit.ai, Microsoft Bing Voice Recognition, IBM Speech to Text, entre otras. Para este proyecto se decidió utilizar la implementación de la API de Google Speech Recognition ya que, a diferencia de la variante en la nube Google Cloud Speech API, esta no necesita ningún tipo de autentificación y no tiene ningún costo.

Una vez implementado el sistema de reconocimiento de voz que convertirá las palabras del usuario en texto es necesario procesar dicho texto y enviar una solicitud al servicio backend. Como se vio en la sección 4.5, para poder comunicar el frontend con el agente conversacional de Dialogflow, se implementó en Cloud Functions una función que sirve como puerta de enlace, para llamar dicha función se debe realizar una solicitud POST por medio del protocolo HTTP a la URL indicada en la consola de Cloud Functions, en el cuerpo de la solicitud es importante incluir el código ISO del lenguaje bajo el cual se está interactuando con el agente conversacional, en este caso "es-ES", y el texto que se le desea enviar, en este caso la transcripción de la pregunta hecha por el usuario.

En caso de que la solicitud HTTP a la función implementada en Cloud Functions haya sido exitosa, se obtendrá una respuesta del agente de Dialogflow, esta respuesta está en formato de texto pero debe convertirse a voz para ser escuchada por el usuario. Para ello se propone el uso de un servicio en la nube de conversión de texto a voz, al igual que para el caso de conversión de voz a texto, existen varias opciones posibles en el mercado por ejemplo Speech-to-Text de Google Cloud y Amazon Polly de AWS. Cada servicio cuenta con diferentes voces en diferentes idiomas, algunas de las cuales están mejoradas con aprendizaje profundo para escucharse más naturales, estas voces son conocidas como "Voces Neuronales" y su uso es más costoso. Para este proyecto se decidió utilizar la voz "Mia" del servicio Amazon Polly, ya que es una voz en español con acento de México y, a gusto del autor, fue las más agradable de escuchar entre las otras opciones.

En este punto del desarrollo del proyecto se realiza la conversión de la solicitud del usuario de voz a texto, se envía la solicitud a Dialogflow por medio de protocolo HTTP y finalmente se reproduce la respuesta de Dialogflow al usuario en formato de voz utilizando un sistema de conversión de texto a voz. Si bien el proceso anterior representa casi en su totalidad la interacción humano-computadora, aún falta una forma de activar el sistema ya que no es eficiente que el sistema reaccione a todas las voces en su entorno sino sólo a las instrucciones que van explícitamente dirigidas a él. Para incorporar este tipo de comportamiento al asistente virtual sin perder la naturalidad de la interacción, es una práctica común el implementar una palabra de activación (también llamada "Wakeword" o "Hotword" en inglés), es decir, una palabra que sirve como interruptor y que al ser escuchada por el asistente, este comienza a interactuar con el usuario hasta completar una solicitud, una vez terminada la conversación el asistente regresa a un estado de reposo en donde solo se activará nuevamente si escucha la palabra de activación.

Todos los asistentes virtuales disponibles en el mercado, y que han sido creados por

grandes empresas como Google, Amazon, Apple y Microsoft, implementan el uso de una palabra de activación para llamar al asistente virtual y hacer uso de sus funciones. Esta palabra de activación suele un nombre dado al asistente virtual, por ejemplo el asistente virtual de la empresa Amazon reacciona al nombre "Alexa", el de la empresa Apple reacciona al nombre "Siri" y el de Microsoft a "Cortana", una excepción a esto es el asistente de Google que no reacciona a un nombre propio sino a la frase "Ok, Google" o "Hey, Google". Para la elaboración de este proyecto se decidió darle al asistente de memoria el nombre de "Maddie" por la forma de pronunciar la palabra que surge al juntar las primeras letras de la siguiente frase en inglés que engloba el concepto principal del proyecto: "Memory-aid Assistant using Deep learning" (MADee).

La interfaz de Dialogflow actualmente no ofrece ninguna forma de implementar una palabra de activación en el asistente, además, sería muy costoso (dado que se están utilizando servicios en la nube) e ineficiente el realizar la detección de la palabra de activación desde el backend de la aplicación, ya que esto implicaría enviar toda la información que recibe el micrófono al backend. Es por lo anterior que se decidió implementar la detección de la palabra de activación desde el frontend de la aplicación, es decir, desde la tarjeta Raspberry Pi.

La tarea de detectar una palabra de activación no es trivial, ya que es necesario enseñarle a un sistema a detectar una palabra en específico sin importar al acento o la persona que lo pronuncia. Es por ello que, para realizar esta tarea se utilizó la librería Snowboy de Python. Snowboy es un detector de "Wakeword" altamente personalizable, de rápida respuesta, que no requiere de conexión a internet para funcionar y que es compatible con la tarjeta Raspberry Pi. Snowboy además ofrece una interfaz web por medio de la cual se puede entrenar un modelo al repetir tres veces la palabra de activación para la cual se desea entrenar, esta tarea debe ser realizada por varias personas para así enseñar al modelo diferentes formas de pronunciación así como y diferentes entonaciones (la documentación oficial sugiere 500 personas para tener un modelo universal para ser usado por cualquier persona, sin embargo el modelo puede ser usado desde la primera grabación). La librería Snowboy cuenta también con código ejemplo para implementar el reconocimiento de Wakeword en el lenguaje Python, lo cual permitió una rápida integración con el proyecto.

De manera general, la interacción completa del usuario con el asistente de memoria puede verse reflejada en la figura 4.22, en donde además se ven representados todos los módulos del asistente de memoria que interactúan entre sí para responder a las solicitudes del usuario.

4.6.3. Aplicación móvil para Android

La comunicación con el asistente de memoria por medio de comandos de voz permite una interacción fluida con los usuarios, sin embargo la misma naturaleza de la interacción presenta limitantes para realizar algunas tareas. Esto se debe a que el usuario no cuenta con una interfaz gráfica donde pueda visualizar varios elementos a la vez, es así que tareas donde se requiera eliminar o editar elementos pueden ser muy

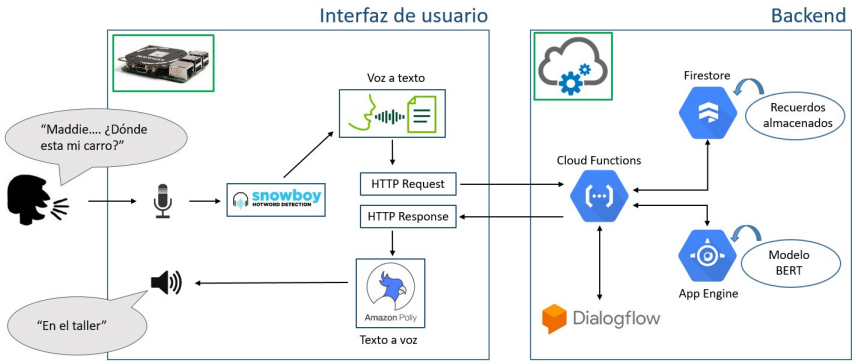


Figura 4.22: Diagrama donde se muestra los módulos involucrados durante la interacción del usuario con el asistente de memoria.

complejas de implementar y confusas de utilizar para los usuarios. Es por ello que, se decidió implementar una aplicación móvil que permita a los usuarios tener otra forma de interactuar con el asistente de memoria.

El sistema para el cual se decidió desarrollar la aplicación es el sistema Android, ya que además de ser un popular sistema operativo para dispositivos móviles, también pertenece a la empresa Google, lo cual permite una fácil integración con los servicios backend del asistente de memoria. El desarrollo de aplicaciones para Android se realiza en la IDE Android Studio, la cual admite el uso de los lenguajes de programación Java y Kotlin. El principal objetivo de la aplicación móvil es el de permitir a los usuarios visualizar y eliminar los recuerdos que han sido previamente almacenados. Esta aplicación está diseñada para ser usada principalmente por los cuidadores de los pacientes que utilizan el asistente de memoria, por lo que se puede aprovechar al máximo las ventajas que ofrece el sistema Android.

Para poder visualizar los recuerdos del usuario almacenados utilizando el asistente de memoria, es necesario acceder a la base de datos del sistema en Firestore 4.5.2. Para ello, Firebase ofrece rápidas opciones de integración tanto para aplicaciones web como aplicaciones móviles, por lo que el proceso de vincular la aplicación de Android con la infraestructura del sistema en la nube viene descrito en la documentación de Google Firebase.

En la figura 4.23 se muestra una captura de la interfaz de Android diseñada, en la cual observa una lista con todos los recuerdos almacenados en el asistente virtual, obtenidos de la base de datos en la nube. Como se muestra en la figura 4.24, es posible seleccionar cada recuerdo de la lista y visualizar su contenido, además, junto con cada recuerdo se muestra la fecha en la cual fue almacenado y dos botones, uno en forma de estrella con el cual se marca el recuerdo como "favorito", y otro en forma de bote de basura que, al ser presionado, muestra una alerta al usuario y le pregunta si desea eliminar permanentemente el recuerdo, en caso de que la acción sea confirmada el recuerdo se eliminara completamente de la base de datos. Este comportamiento se

Figura 4.23: Visualización de los recuerdos almacenados en la aplicación móvil.

muestra en la figura 4.25.

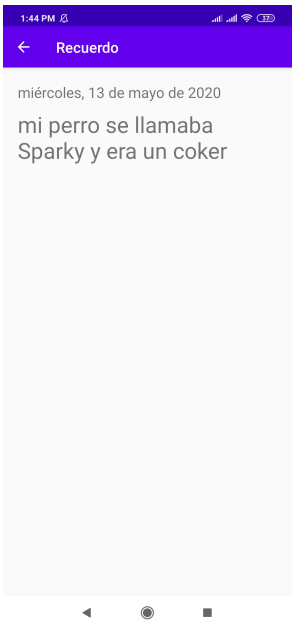


Figura 4.24: Visualización de los detalles de un recuerdo en la aplicación móvil.

Además de la posibilidad de administrar los recuerdos almacenados, se decidió uti-

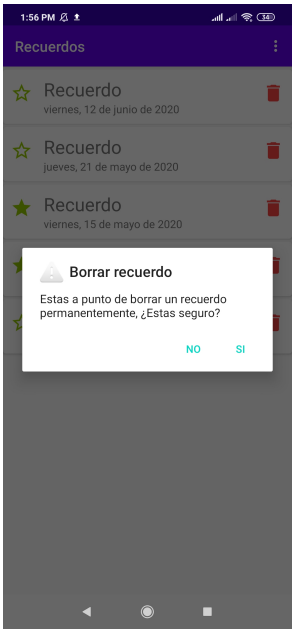


Figura 4.25: Eliminación de un recuerdo en la aplicación móvil.

lizar la aplicación de Android como otra interfaz para conversar con el asistente de memoria. A diferencia de la interfaz por comandos de voz presentada previamente en la sección 4.6.2, donde la prioridad en el diseño fue su uso sencillo para simplificar la interacción con los pacientes con problemas de perdida de memoria, esta interfaz está diseñada para ser utilizada mayormente por sus cuidadores, por lo que se decidió priorizar el hecho de poder acceder al asistente de memoria desde cualquier punto y poder registrar nuevos recuerdos de forma más rápida. Es por ello que se decidió utilizar solamente texto como medio de comunicación y no utilizar comandos de voz como, por lo que se decidió hacer una interfaz estilo chat, la cual puede verse en la figura 4.26.

Es importante mencionar que tanto la interfaz de voz como la interfaz por chat se comunican al mismo asistente de memoria en la nube, por lo que si se almacena un recuerdo usando cualquiera de los dos medios se verá reflejado también en el otro. Además la aplicación móvil fue programada con un "oyente" a la base de datos, esto quiere decir que al presentarse cualquier cambio, como inserción o eliminación de recuerdos, se actualizara en tiempo real la información que se despliega en la aplicación.

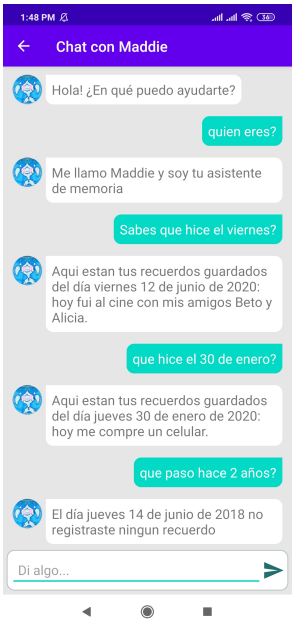


Figura 4.26: Comunicación con el asistente de memoria por medio de la interfaz de chat.

Capítulo 5

Análisis de Resultados

En su mayoría, los sistemas disponibles actualmente para asistir a personas con trastornos de memoria como la demencia y el Alzheimer, están diseñados para monitorear las condiciones de vida las personas y así evitar accidentes o detectar enfermedades en fases tempranas. El sistema propuesto seria de los primeros en enfocarse en mejorar la calidad de vida tanto de las personas que padecen trastornos de memoria como de sus cuidadores.

Para el desarrollo del sistema de asistencia presentado en este trabajo se utilizaron técnicas avanzadas de inteligencia artificial junto con servicios de computación en la nube, lo cual permitió obtener un sistema funcional, robusto, de uso fácil e intuitivo. El asistente de memoria desarrollado es capaz de almacenar información relevante para después asistir a los usuarios respondiendo a preguntas que estos le hagan mediante un sistema de preguntas respuestas y por medio de comandos de voz. Además, gracias a los últimos avances en técnicas de aprendizaje profundo y procesamiento de lenguaje natural, fue posible adaptar el asistente de memoria para que funciona correctamente en el idioma español.

El asistente de memoria presentado en este trabajo fue diseñado teniendo en mente que la mayoría de sus usuarios serian gente de edad avanzada, esto debido a la naturaleza de los problemas de memoria. Es por ello que elaboró una interfaz sencilla de utilizar e inspirada en otros asistentes virtuales comerciales como Alexa de Amazon y Google Assistant que, si bien están orientados para ser utilizados por otro tipo de usuarios y buscan resolver otras tareas, han demostrado que la comunicación por comandos de voz y el uso de un sistema de dialogo para resolver tareas, son una forma efectiva de comunicación con todo tipo de usuarios. En la figura 5.1 se muestra una foto del dispositivo diseñado para que el usuario interactúe con el asistente de memoria, utilizando el hardware indicado en la sección 4.6.1.

Para el desarrollo de este proyecto se utilizaron como infraestructura servicios en la nube, lo cual ofrece diversas ventajas sobre otros enfoques de desarrollo tradicional donde la aplicación solo reside en un servidor físico que tiene que ser configurado y mantenido. Además de contar con una infraestructura constantemente actualizada y asegurada por el proveedor de servicios en nube, el hecho de que el asistente de memoria

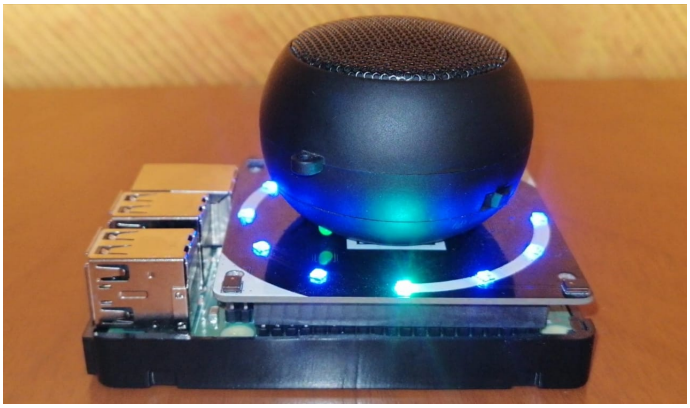


Figura 5.1: Dispositivo diseñado como interfaz para establecer comunicación con el asistente de memoria.

resida en la nube permitió tener la flexibilidad de comunicarlo con más de una interfaz de usuario, con el propósito de atender distintas necesidades, conservando su funcionalidad y teniendo consistencia en la información almacenada.

Como se explicó en la sección 2.4, una de las principales características de los sistemas de dialogo orientados a tareas es su capacidad para identificar las solicitudes del usuario tomando como referencia una lista de escenarios previamente definidos lo cual se conoce como "clasificación de intenciones", donde además cada intención puede tener "slots", es decir, información específica que el usuario debe proporcionar para que el sistema de dialogo pueda completar correctamente la intención. Es importante mencionar que, si bien técnicamente no existe un límite para la cantidad de intenciones que puede tener un asistente, con el propósito de delimitar este proyecto de tesis solo se integraron en el asistente intenciones que buscan atender el problema de pérdida de memoria.

Considerando el diseño presentado en la sección 4.2 se definieron tres principales escenarios de interacción entre el usuario y el asistente de memoria. El primero es el de registrar un recuerdo, para ello se definió una intención que identifica la necesidad del usuario de registrar un nuevo recuerdo en la base de datos, para ello el usuario debe utilizar palabras como "quiero almacenar que..." o "registra que...", además se definió un slot que debe ser llenado para terminar la intención exitosamente y este es el contenido del recuerdo. De esta forma si el usuario solo expresa que desea registrar un recuerdo pero no indica cual es dicho recuerdo, el asistente le preguntará de forma explícita sobre cuál es el contenido que desea almacenar, de la misma forma si el usuario expresa en una misma oración que desea almacenar un recuerdo y en seguida indica el contenido de dicho recuerdo, el sistema llenara el slot y procederá con la conversación. Para este escenario además se agregó una fase de confirmación, es decir, después de que el asistente identifique tanto la intención de registrar un recuerdo como el contenido del mismo, se le leerá al usuario la información capturada y se le pedirá una confirmación

para almacenar el recuerdo en la base de datos. Esto último se implementó para que el usuario valide la información capturada, ya que pudo surgir un error en la conversación de voz a texto (debido a situaciones externas como un exceso de ruido en el ambiente) y la información capturada por el asistente no sea exactamente la que el usuario expresó. En caso de que la confirmación sea positiva, el recuerdo se almacenara en la base de datos del asistente de memoria, donde junto con el contenido del recuerdo también se registrara la fecha en la que fue almacenado.

El segundo de los tres principales escenarios definidos de interacción entre el usuario y el asistente de memoria, es el de escuchar los recuerdos almacenados en un día específico. Similar al escenario de registrar recuerdos, el usuario debe manifestar su necesidad de escuchar los recuerdos registrados utilizando frases como "dime que fue lo que paso el día...", "léeme mis recuerdos del día..." o "que fue lo que hice ayer". Para esta intención, el asistente solo tiene un slot que es considerado como opcional y es llenado con la fecha que corresponde al día del cual se desean escuchar los recuerdos almacenados. En caso de que el usuario no proporcione ninguna fecha el asistente le dirá los recuerdos almacenados en el día actual, y en caso de no tener registrado ningún recuerdo el día indicado, el asistente le indicará al usuario que ese día no hubo ningún registro. Es importante indicar que, las capacidades de Dialogflow le permiten al asistente virtual identificar el día exacto (número de día, mes y año) al que se refiere el usuario aun cuando este último no lo diga de forma explícita y diga frases como "dime los recuerdo que registre ayer", es decir, de manera interna el asistente convertirá la palabra "ayer" en el día, mes y año correspondiente. El mismo caso ocurre para otras frases similares como por ejemplo "¿Qué hice hace un año?" o "dime lo que registre hace cinco meses". En caso de que el usuario o el cuidador deseen conocer de una forma más interactiva y visual todos los recuerdos almacenados, es posible visualizar una lista de los mismos usando la aplicación para Android desarrollada para este proyecto de tesis y cuyos detalles fueron descritos en la sección 4.6.3. Es importante recordar que la única forma en la cual se pueden eliminar recuerdos es por medio de esta aplicación.

El tercero de los tres principales escenarios de interacción definidos, corresponde al evento donde el usuario le hace cualquier pregunta al asistente de memoria y este último le responde con base en la información contenida en los recuerdos previamente almacenados. Este escenario fue el más complejo de implementar pero es posiblemente también el más relevante, ya que a diferencia del escenario donde solo se le comunican al usuario los recuerdos almacenados en un día definido, en este escenario se le puede responder al usuario preguntas muy específicas que este tenga sobre su vida personal o sobre recientes eventos pasados.

Para la resolución de esta tarea se utilizaron técnicas de estado del arte en inteligencia artificial, donde básicamente se entrenó un modelo matemático capaz de responder preguntas dado un texto del cual obtendrá las respuestas, para esta aplicación específica, este texto está compuesto por los recuerdos del usuario almacenados en la base de datos del asistente de memoria. De esta forma se garantiza que las respuestas otorgadas por el asistente son completamente basadas en eventos registrados por el usuario o su cuidador. Además el modelo matemático entrenado es capaz de trabajar en el idioma

español e identificar cuando la respuesta a la pregunta no se encuentra en el texto de entrada, en cuyo caso no dará una respuesta.

El entendimiento del funcionamiento general del asistente es importante para su correcta utilización, ya que la calidad de las respuestas depende tanto de la cantidad de recuerdos registrados, como de la claridad con la cual fueron registrados los mismos. Por ello es importante aclarar, que el asistente de memoria solo dará como respuesta información que se encuentre escrita de forma explícita en los recuerdos, es decir, no inferirá y generará texto que no este presente en los mismos. Esto implica, que es responsabilidad de los cuidadores verificar que los recuerdos almacenados por el asistente contengan correctamente la información que se desea el usuario escuche, de lo contrario el asistente de memoria puede no encontrar la respuesta a la pregunta o decir información incorrecta. En la figura 5.2 se muestran tres ejemplos donde se realizan preguntas con información ambigua o que no está especificada de forma explícita en los recuerdos almacenados, y por ello, el asistente virtual podría responder con información errónea.

En los siguientes ejemplos el asistente de memoria entregará una respuesta incorrecta o indicará que no tiene la suficiente información para contestar.		
Limitante	**Ejemplos de recuerdos almacenado**	**Pregunta**
No realiza sumas, restas o cualquier otra operación matemática.	"Ayer fui al cine. Antier fui al cine. Hace 5 días fui al cine."	"¿Cuantas veces he ido al cine?"
No contesta "si" o "no".	"Ayer lleve mi carro al taller mecánico."	"¿Sabes si ayer lleve mi carro al taller mecánico?"
No puede lidiar con ambigüedades.	"Ayer fui a la fiesta de mi amigo Alex con mi amigo Víctor."	"¿Cómo se llama mi amigo?"

Figura 5.2: Ejemplos de casos de mal uso del asistente de memoria.

Como se detalló en la sección 4.4, se desarrollo exitosamente el sistema de preguntas y respuestas en el idioma español, sin embargo, este sistema tiene un menor desempeño que otras otras versiones del mismo en el idioma inglés, lo se ve reflejado en su puntaje F1. Para la versión en español del sistema se obtuvo un puntaje F1 máximo de 68.23 utilizando en modelo BERT pre-entrenado en el idioma español [7] y una versión del conjunto de datos SQuAD 2.0 traducida el español [8]. En contraste, durante el desarrollo del proyecto se hicieron las primeras pruebas con modelos pre-entrenados en el idioma inglés y se alcanzó un puntaje F1 máximo de 78.91. Si bien el puntaje obtenido con el modelo en español esta casi 10 puntos por debajo del modelo en inglés, es importante aclarar que esto no significa precisamente que el modelo vaya a cometer más errores al momento de responder preguntas, sino que el usuario debe guardar los recuerdos de forma más clara y especificando de forma más explícita la respuesta que espera obtener. Además también es importante aclarar que la versión original del conjunto de datos SQuAD 2.0 no coloca el desempeño del ser humano con un puntaje F1 de 100 sino de 89.45.

Además de los tres principales escenarios de interacción entre el usuario y el asistente de memoria definidos previamente, se le incorporaron al asistente otras capacidades como la posibilidad de proporcionar la hora y fecha actual (incluyendo el día de la

semana). Esto fue agregado considerando el hecho de que las personas que sufren de problemas de pérdida de memoria comúnmente pierden la noción del día en el que viven.

En general, las respuestas de interacción definidas en el asistente, junto con el corto tiempo de respuesta del sistema gracias a la optimización obtenida por el uso de servicios en la nube, ofrecen una interacción natural y fluida con los usuarios. Además, el hecho de que el sistema funcione por medio de comandos de voz y no requiera que el usuario aprenda a utilizar un hardware específico, facilita si uso y permite que sea un sistema accesible para una población de edad más avanzada y que, en la mayoría de los casos, no esta tan acostumbrada al uso de dispositivos tecnológicos.

Capítulo 6

Conclusiones

El aumento en la expectativa de vida de las personas trae consigo la necesidad de desarrollar nuevos sistemas de asistencia que permitan enfrentar las nuevas enfermedades y retos que trae consigo el envejecimiento. Este proyecto de tesis tuvo como propósito el desarrollo de una plataforma de apoyo para personas con problemas leves de memoria. Se comprobó que mediante el uso de inteligencia artificial y servicios de computación en la nube, se puede crear una herramienta capaz de almacenar recuerdos y asistir a las personas en recordar detalles de eventos pasados. El uso de nuevas tecnologías permitió el desarrollo de un sistema de asistencia con altas capacidades, que además es de uso sencillo y compatible en el idioma español.

La aparición de nuevas tecnologías, así como la mejora de otras ya existentes, ha permitido la creación de nuevos productos y servicios diseñados para asistir al ser humano y mejorar su calidad de vida. Los últimos avances en el campo de la inteligencia artificial jugaron un papel importante en el desarrollo de este proyecto, ya que gracias a la gran cantidad de datos disponibles públicamente y a la mejora en la capacidades de cómputo de los sistema actuales, es posible crear sistemas inteligentes capaces de realizar tareas complejas, como es el caso de la tarea de responder preguntas (QA), la cual fue clave para la elaboración del sistema de asistencia de memoria.

En este trabajo se presentó el desarrollo de un asistente virtual para personas con pérdida de memoria en la forma de un agente conversacional (chatbot). La comunicación entre el asistente de memoria y el usuario es por medio de comandos de voz, emulando a otros asistentes virtuales comerciales y simplificando en gran medida la interacción entre humano y computadora.

Para la elaboración de este agente conversacional se utilizó la herramienta de Procesamiento de Lenguaje Natural llamada Dialogflow, la cual permitió la creación de un sistema capaz tanto de entender e identificar las solicitudes del usuario, como de coordinar las acciones necesarias entre otros servicios para completar correctamente la solicitud. Así mismo, parte importante del desarrollo de este asistente de memoria fue la creación de un sistema de respuesta a preguntas, para lo cual se utilizaron las herramientas más recientes en el campo del Aprendizaje Profundo y el Procesamiento de Lenguaje Natural, en específico los modelos BERT, los cuales permitieron crear un

modelo capaz de responder a las preguntas del usuario en el idioma español y con un buen desempeño.

Como infraestructura del proyecto se utilizaron servicios en la nube, lo cual permitió tener un enfoque "sin servidores" que facilitó en gran medida la implementación y despliegue de otros servicios necesarios para el correcto funcionamiento del asistente, como es el caso de una base datos. La computación en la nube ha demostrado ser una tecnología muy prometedora, la cual ha ganado popularidad en los últimos años gracias a las facilidades que ofrece para el desarrollo e implementación de nuevos productos y servicios.

Como medio de interacción entre usuario y asistente de memoria se utilizó una tarjeta Raspberry Pi adaptada con un micrófono y una bocina, en este dispositivo se implementaron librerías y llamadas a otros servicios en la nube para convertir los comandos de voz del usuario a texto y convertir las respuestas del asistente de texto a voz. Además, se desarrolló una aplicación móvil para el sistema Android, la cual fue diseñada como herramienta de apoyo para los cuidadores de las personas con problemas de pérdida de memoria, y que ofrece, junto con algunas opciones de configuración del sistema, la posibilidad de visualizar y validar los recuerdos almacenados por el asistente virtual.

De forma general, se desarrolló un asistente de memoria funcional, en idioma español, capaz de identificar y atender las solicitudes de los usuarios que por el momento pueden ser guardar recuerdos, leer los recuerdos almacenados, contestar a cualquier pregunta del usuario con base a los recuerdos almacenados y abstenerse de contestar si no se cuenta con información suficiente. Además, es importante mencionar que la arquitectura del sistema, así como las tecnologías empleadas en su desarrollo permiten que se puedan agregar más funciones, o mejorar las ya existentes, de forma rápida y eficiente.

Es importante recalcar que gran parte del correcto funcionamiento del asistente de memoria depende del uso de modelos de inteligencia artificial, en específico de modelos basados en aprendizaje profundo. Aunque en este proyecto de tesis solo se presentó el desarrollo del sistema QA, en general el asistente de memoria hace uso de varios modelos dependientes de esta tecnología. Por ejemplo, la herramienta Dialogflow hace uso de técnicas de aprendizaje profundo para crear un modelo capaz de entender el lenguaje e identificar las solicitudes del usuario, es por ello que, durante el proceso de creación de un agente conversacional, Dialogflow le solicita al desarrollador que le proporcione algunas frases ejemplo con las cuales ajustará los parámetros de su modelo e identificara correctamente las solicitudes de los usuarios. Otros elementos del asistente de memoria que utilizan técnicas de inteligencia artificial son los módulos de conversión de voz a texto y texto a voz, aquí se utilizan modelos de aprendizaje profundo para capturar mejor las palabras dichas por el usuario y para producir una voz que suene de forma más natural, respectivamente.

El aprendizaje profundo es una subrama de la inteligencia artificial que ha revolucionado la forma de procesar datos en diversas áreas de la computación. En este trabajo se investigó su impacto en el área del procesamiento de lenguaje natural, donde se desa-

rrollaron modelos pre-entrenados que hacen uso de un mecanismo de atención para comprender de forma más efectiva las oraciones de entrada (modelos BERT), lo cuales ha permitido mejorar significativamente el desempeño de los sistemas computacionales en diversas tareas de procesamiento de lenguaje natural. Una de las principales contribuciones de este trabajo es el estudio e implementación de dichos modelos en el idioma español, ya que en su mayoría, la investigación realizada en este campo se hace en torno al idioma inglés.

En este proyecto se presentaron y evaluaron los resultados de entrenar diversos modelos BERT para resolver la tarea de responder a preguntas en el idioma español. Para esto se utilizaron tanto modelos pre-entrenados solamente con datos en español, como modelos multi-idioma, es decir, modelos pre-entrenados con datos en diversos idiomas (incluido el español). La viabilidad de los modelos multi-idioma se consideró de gran importancia ya que, en caso de no contar suficientes datos de entrenamiento en un idioma específico para resolver una tarea de procesamiento de lenguaje natural definida, sería de gran utilidad el poder reutilizar modelos pre-entrenados por otras personas, sin ser necesario que estos hayan sido entrenados con datos en el idioma objetivo en un principio.

Con base a los datos obtenidos de forma empírica, se concluye que si bien los modelos entrenados solamente en idioma español presentaron un mejor puntaje F1 para la resolución de la tarea de QA, el puntaje de los modelos multi-idioma fue sorprendentemente similar, siendo sólo ligeramente menor. Esto implica que, si en un caso futuro se desea resolver una tarea de NLP pero no se cuenta con suficientes datos de entrenamiento en el idioma deseado, es viable considerar el uso de modelos BERT multi-idioma para resolver dicha tarea. Cabe recalcar también, que en el caso del idioma inglés se obtuvó el mejor puntaje F1 en la tarea de QA utilizando el modelo llamado ALBERT (A Lite BERT), el cual, al momento de redactar este documento, aún no cuenta con una versión multi-idioma, pero que según el repositorio oficial del proyecto ya se encuentra en desarrollo, por lo que es una opción que se debe considerar para trabajos futuros.

Los campos de la inteligencia artificial y el procesamiento de lenguaje están actualmente en constante desarrollo y crecimiento. Grandes empresas de tecnología como Google y Facebook se encuentran desarrollando nuevos y más eficientes modelos de aprendizaje profundo, descubriendo mejores arquitecturas de redes neuronales y haciendo uso de la inmensa cantidad de datos a los que tienen acceso. Por ello es importante mantenerse actualizado en los últimos avances en el campo, y así aplicar dichos conocimientos para mejorar modelos y productos ya existentes, o desarrollar nuevos. Este proyecto de tesis no es la excepción, para la elaboración del sistema de asistencia se utilizaron algunos de los modelos de aprendizaje profundo más recientes y poderosos, sin embargo es importante seguir actualizando el sistema con nuevos avances tecnológicos para brindarle al usuario el mejor servicio posible.

Es importante recalcar también, que otra contribución de este proyecto de tesis fue el desarrollo de un sistema de asistencia que no se enfoca en monitorear a la personas que padecen de problemas leves de memoria, o en su defecto realizar un diagnóstico del padecimiento, como es el caso de otros trabajos similares. Este sistema busca mejorar la

calidad de vida de las personas que ya padecen de este tipo de problemas al brindarles una herramienta que les brinde un mayor grado de autonomía, además de facilitar la tarea de sus cuidadores.

Cuando una persona sufre de problemas de memoria, los principales afectados son sus familiares cercanos y en particular sus cuidadores. Esto se debe al enorme grado de dependencia que se genera entre las personas involucradas, el cual no solo afecta su entorno social, sino también genera afectaciones emocionales junto con altos grados de estrés. El asistente de memoria presentado en este proyecto fue diseñado pensando fuertemente en los cuidadores de pacientes con problemas de memoria, ya que en ellos recae una fuerte responsabilidad la cual puede llevarlos a desarrollar problemas como depresión o incluso desarrollar ellos también problemas de pérdida de memoria. Además, su estado de ánimo de los cuidadores tiene un fuerte impacto sobre la salud del paciente, por lo cual, el uso de este sistema de asistencia podría presentar una mejora en la calidad de vida de varias personas dentro de un ambiente con estas características.

El uso de inteligencia artificial es comúnmente un tema controversial debido a los diversos usos que se le pueden dar, siendo varios de ellos éticamente cuestionables o implicando en gran parte de los casos la automatización de procesos hechos actualmente por humanos, lo cual se puede traducir como desempleo. Sin embargo, es importante considerar que la humanidad siempre ha estado y seguirá estando en constante avance, innovando y creando nuevos productos que faciliten o eficienticen su trabajo. Actualmente la inteligencia artificial está presente en varias de nuestras actividades diarias como el que aparezcan productos que nos interesaría comprar en páginas como Amazon, o nos aparezcan recomendaciones de películas o videos que nos gustaría ver en plataformas como Netflix y Youtube. De hecho hay ejemplos todavía más sutiles de implementaciones inteligencia artificial como en las bandejas de correo donde se separan los correos de spam de los demás y donde nos aparecen respuestas recomendadas basadas en el contenido del correo. La inteligencia artificial también está presente en diversas aplicaciones donde se utilizan técnicas de visión por computadora, como es el caso de Facebook donde se etiquetan fotos con nuestro nombre de forma automática o dentro de nuestro smartphones, donde se agrupan todas las fotos relacionadas con una persona en específico sin que nosotros lo indiquemos de forma explícita.

La inteligencia artificial, y en general la tecnología, siempre puede ser usada para realizar buenas o malas acciones, para desarrollar productos que ayuden a realizar tareas que ayuden a los seres humanos y mejoren su calidad de vida, o para crear productos éticamente cuestionables y que resulten perjudiciales a largo plazo. Esto no implica que sea esta una disciplina con la que se debe estar en constante debate, sino una herramienta que debemos estar preparados para utilizar y aprovechar al máximo, para obtener de ella la gran cantidad de beneficios que nos puede ofrecer y crear productos que antes solo se podrían considerar como ciencia ficción.

Referencias

[1] Airehrour, D., Madanian, S. and Abraham, A. M. [2018], 'Designing a memory-aid and reminder system for dementia patients and older adults'. 26

[2] Alammar, J. [2018], 'The illustrated bert, elmo, and co.(how nlp cracked transfer learning)', recuperado de: http://jalammar.github.io/illustrated-bert/. 32

[3] Bahdanau, D., Cho, K. and Bengio, Y. [2014], 'Neural machine translation by jointly learning to align and translate', *arXiv preprint arXiv:1409.0473* . 18

[4] Bengio, Y., Simard, P. and Frasconi, P. [1994], 'Learning long-term dependencies with gradient descent is difficult', *IEEE transactions on neural networks* **5**(2), 157–166. 16

[5] Bird, S., Klein, E. and Loper, E. [2009], *Natural language processing with Python: analyzing text with the natural language toolkit*, O'Reilly Media, Inc. 7

[6] Bogdani [2018], 'Building chatbots', recuperado de: https://nlpforhackers.io/chatbots-introduction/. 27

[7] Cañete, J., Chaperon, G., Fuentes, R. and Pérez, J. [2020], Spanish pre-trained bert model and evaluation data, *in* 'to appear in PML4DC at ICLR 2020'. 37, 55, 78

[8] Carrino, C. P., Costa-jussà, M. R. and Fonollosa, J. A. R. [2019], 'Automatic spanish translation of the squad dataset for multilingual question answering'. 52, 56, 78

[9] Chan, B. [2020], 'XLM-RoBERTa: The alternative for non-english NLP', recuperado de: https://towardsdatascience.com/xlm-roberta-the-multilingual-alternative-for-non-english-nlp-cf0b889ccbbf. 39

[10] Chollet, F. [2017], *Deep learning with Python*, Manning Publications Company. 12

[11] Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, E., Ott, M., Zettlemoyer, L. and Stoyanov, V. [2019], 'Unsupervised cross-lingual representation learning at scale', *arXiv preprint arXiv:1911.02116* . 53

[12] Devlin, J., Chang, M.-W., Lee, K. and Toutanova, K. [2018], 'Bert: Pre-training of deep bidirectional transformers for language understanding', *arXiv preprint arXiv:1810.04805* . 31, 33, 34, 35, 36, 37, 38

[13] Dua, T., Nichols, P. and Setoya, Y. [2013], 'Deméncia, una prioridad de salud pública', *Monografia a internet. Washington: Organización Mundial de la Salud* . 3

[14] Facebook Wit.ai. [2019], 'Wit.ai, documentation', recuperado de: https://wit.ai/docs. 29

[15] Ganegedara, T. [2018], *Natural Language Processing with TensorFlow: Teach language to machines using Python's deep learning library*, Packt Publishing Ltd. 7

[16] Goldberg, Y. and Hirst, G. [2017], 'Neural network methods in natural language processing. morgan & claypool publishers'. 7

[17] Goodfellow, I., Bengio, Y. and Courville, A. [2016], *Deep learning*, MIT press. 12

[18] Google Cloud [2020], 'Google app engine documentation', recuperado de: https://cloud.google.com/appengine/docs. 57, 59

[19] Google Dialogflow [2019], 'Dialogflow documentation', recuperado de: https://cloud.google.com/dialogflow/docs/. 28

[20] Google Firebase [2020*a*], 'Cloud firestore', recuperado de: https://firebase.google.com/docs/firestore. 60

[21] Google Firebase [2020*b*], 'Cloud functions for firebase', recuperado de: https://firebase.google.com/docs/functions. 62

[22] Ishii, H., Kimino, K., Aljehani, M., Ohe, N. and Inoue, M. [2016], 'An early detection system for dementia using the m2 m/iot platform', *Procedia Computer Science* **96**, 1332–1340. 25

[23] Lample, G. and Conneau, A. [2019], 'Cross-lingual language model pretraining', *arXiv preprint arXiv:1901.07291* . 38, 39

[24] Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P. and Soricut, R. [2019], 'Albert: A lite bert for self-supervised learning of language representations', *arXiv preprint arXiv:1909.11942* . 35, 36, 51

[25] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L. and Stoyanov, V. [2019], 'Roberta: A robustly optimized bert pretraining approach', *arXiv preprint arXiv:1907.11692* . 39

[26] Luong, M.-T., Pham, H. and Manning, C. D. [2015], 'Effective approaches to attention-based neural machine translation', *arXiv preprint arXiv:1508.04025* . 18

[27] Martin, L., Muller, B., Suárez, P. J. O., Dupont, Y., Romary, L., de la Clergerie, É. V., Seddah, D. and Sagot, B. [2019], 'Camembert: a tasty french language model', *arXiv preprint arXiv:1911.03894* . 37

[28] Mayo Clinic [2017], 'Amnesia', recuperado de: https://www.mayoclinic.org/diseases-conditions/amnesia/symptoms-causes/syc-20353360. 1, 2

[29] Mikolov, T., Chen, K., Corrado, G. and Dean, J. [2013], 'Efficient estimation of word representations in vector space', *arXiv preprint arXiv:1301.3781* . 30

[30] Mitchell, T. M. [1997], *Machine learning*, McGraw-Hill Science/Engineering/Math. 9

[31] Olah, C. [2015], 'Understanding lstm networks, 2015', recuperado desde: http://colah.github.io/posts/2015-08-Understanding-LSTMs. 16

[32] Park, Y.-S. and Lek, S. [2016], Artificial neural networks: Multilayer perceptron for ecological modeling, *in* 'Developments in environmental modelling', Vol. 28, Elsevier, pp. 123–140. 12, 13

[33] Pennington, J., Socher, R. and Manning, C. D. [2014], Glove: Global vectors for word representation, *in* 'Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)', pp. 1532–1543. 30

[34] Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K. and Zettlemoyer, L. [2018], 'Deep contextualized word representations', *arXiv preprint arXiv:1802.05365* . 30, 32, 34

[35] Pires, T., Schlinger, E. and Garrette, D. [2019], 'How multilingual is multilingual bert?', *arXiv preprint arXiv:1906.01502* . 37

[36] Potapenko, A., Zimovnov, A., Kozlova, A., Zobnin, A. and Yudin, S. [2017], 'Natural language processing', recuperado de: https://www.coursera.org/learn/language-processing. 21

[37] Rajpurkar, P., Jia, R. and Liang, P. [2018], 'Know what you don't know: Unanswerable questions for squad', *arXiv preprint arXiv:1806.03822* . 47

[38] Rajpurkar, P., Zhang, J., Lopyrev, K. and Liang, P. [2016], 'Squad: 100,000+ questions for machine comprehension of text', *arXiv preprint arXiv:1606.05250* . 47

[39] RASA [2019], 'Rasa documentation', recuperado de: https://rasa.com/docs/. 28

[40] Seeed Studio [2020], 'Respeaker 4-mic array for raspberry pi', recuperado de: https://wiki.seeedstudio.com/ReSpeaker_4_Mic_Array_for_Raspberry_Pi/. 66

[41] Snips [2019], 'Snips documentation', recuperado de: https://snips.ai/. 27

[42] Tokunaga, S., Horiuchi, H., Takatsuka, H., Saiki, S., Matsumoto, S., Nakamura, M. and Yasuda, K. [2016], Implementation and evaluation of interactive memory-aid agent service for people with dementia, *in* 'International Conference on Digital Human Modeling and Applications in Health, Safety, Ergonomics and Risk Management', Springer, pp. 357–368. 26

[43] Vasilev, I., Slater, D., Spacagna, G., Roelants, P. and Zocca, V. [2019], *Python Deep Learning: Exploring deep learning techniques and neural network architectures with Pytorch, Keras, and TensorFlow*, Packt Publishing Ltd. 12, 14

[44] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł. and Polosukhin, I. [2017], Attention is all you need, *in* 'Advances in neural information processing systems', pp. 5998–6008. 19, 20, 33

[45] Virtanen, A., Kanerva, J., Ilo, R., Luoma, J., Luotolahti, J., Salakoski, T., Ginter, F. and Pyysalo, S. [2019], 'Multilingual is not enough: Bert for finnish', *arXiv preprint arXiv:1912.07076* . 37

[46] Weerakoon, D., Kahandawaarachchi, K., Dissanayake, J., Thilakasiri, W. and Shanthakumara, W. [2018], 'Memory improvement tool for dementia alzheimer's patients', *Procedia Computer Science* **141**, 413–420. 26

[47] Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M. and Brew, J. [2019], 'Huggingface's transformers: State-of-the-art natural language processing', *ArXiv* **abs/1910.03771**. 52

[48] World Health Organization [2018], 'Towards a dementia plan: a who guide'. 3

[49] Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K. et al. [2016], 'Google's neural machine translation system: Bridging the gap between human and machine translation', *arXiv preprint arXiv:1609.08144* . 33