Trabajo Fin de Grado

Desarrollo de un chatbot interactivo usando técnicas de Deep Learning para la comprensión de texto

Development of an interactive chatbot using Deep Learning techniques for text understanding

Autor

Álvaro Monteagudo Moreno

Directora

Paula Peña Larena

Ponente

Sandra Baldassarri

ESCUELA DE INGENIERÍA Y ARQUITECTURA
2018

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe acompañar al Trabajo Fin de Grado (TFG)/Trabajo Fin de Máster (TFM) cuando sea depositado para su evaluación).

D./Dª. Álvaro Monteagudo Moreno,

con nº de DNI 73012674W en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de (Grado/Máster) Grado, (Título del Trabajo) Desarrollo de un bot interactivo usando técnicas de Deep Learning para la comprensión de texto,

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 31 de julio de 2018

Fdo: Álvaro Monteagudo Moreno

AGRADECIMIENTOS

Agradezco al Instituto Tecnológico de Aragón que me haya ofrecido la oportunidad de trabajar con ellos durante un período de prácticas y que posteriormente hayan contado conmigo para la realización de un proyecto que completara mi grado en Ingeniería Informática. En especial me gustaría agradecer a Paula Peña y Rafael del Hoyo por orientarme y guiarme a lo largo de toda la realización del proyecto, por ayudarme en los problemas encontrados, y por haber dedicado parte de su tiempo al seguimiento del proyecto.

También me gustaría agradecer a Sandra Baldasarri su disposición para tutorar este proyecto y ayudarme a centrar las ideas del proyecto y marcar unas pautas de trabajo adecuadas. Quiero hacer especial hincapié en la relevante ayuda que me ha ofrecido, junto a Paula Peña, para la maquetación, estructura y redacción de este documento.

No quisiera olvidarme de mis compañeros de trabajo que han ofrecido su ayuda desinteresadamente para ayudarme a comprender la metodología de trabajo del departamento, así como el uso adecuado de las herramientas del mismo. También destacar el agradable entorno de trabajo que se respira dentro del departamento, siempre hay alguien dispuesto a ayudar, esto hace de la realización de un proyecto como este, una tarea mucho más amena, entretenida y de un alto nivel de desarrollo personal.

Por último quiero agradecer a mis amigos y a mi familia su constante apoyo durante toda la carrera. Sin ellos no habría sido posible seguir la senda de trabajo que me ha llevado a la finalización del grado de manera satisfactoria.

Desarrollo de un bot interactivo usando técnicas de Deep Learning para la comprensión de texto

RESUMEN

Este Trabajo de Fin de Grado consiste en el desarrollo de un sistema utilizando técnicas de aprendizaje automático y comprensión de textos que sea capaz de mantener una conversación con un usuario. El sistema final es capaz de responder a preguntas sobre el corpus formado por datos abiertos del Gobierno de Aragón así como mantener una conversación sencilla con el usuario. Esta propuesta de proyecto surge de una beca por parte de ITAINNOVA para trabajar junto al departamento de Big Data y Sistemas Cognitivos.

Hoy en día las aplicaciones de mensajería están a la orden del día; millones de personas las usan a diario y con mucha frecuencia. Esto ha supuesto la aparición de una nueva tecnología: los *chatbots*. Los *chatbots* son software que interactúan con los usuarios proporcionando una infinidad de servicios (enviar correos, descargar música, reservar un hotel, etc) a través de la comunicación escrita, es decir, usando el lenguaje natural, sustituyendo de esta manera paulatina pero continuamente a las aplicaciones convencionales usadas hasta el momento.

Este trabajo se ha centrado en abordar diferentes técnicas de aprendizaje automático y más en concreto, en aquellas relacionadas con las redes neuronales, y técnicas de comprensión de textos para responder a la problemática que plantea la búsqueda de respuestas. Para ello, se ha realizado un estudio previo de todas las tecnologías involucradas en el proyecto y se han seleccionado las herramientas más adecuadas para el desarrollo del mismo. Tras este estudio, se ha decidido continuar con la línea de investigación del ITAINNOVA en el campo de las redes neuronales de memoria, concretamente con el modelo clave-valor. La idea de estas redes es combinar las técnicas tradicionales de aprendizaje automático añadiéndole un componente extra que funciona como celda de almacenamiento, la cual permite recuperar información y razonar sobre ésta. Para que el sistema mantenga una conversación con sentido para el usuario, se ha optado por un *framework* conversacional llamado Rasa que permite el seguimiento y almacenamiento de la información más relevante de la conversación y con ésta, construir respuestas adecuadas.

Una vez determinadas las tecnologías necesarias, se ha implementado un *chatbot* que cumple los objetivos especificados. Se ha utilizado el lenguaje de programación `Python` debido a su gran versatilidad y a que, tanto `Tensorflow` (librería especializada en el modelado y validación de redes neuronales), como Rasa (librería orientada a la creación de *chatbots*), están desarrolladas con este lenguaje de programación.

Índice

Capítulo 1

Introducción

Actualmente, la tecnología está evolucionando vertiginosamente, dando lugar a la aparición de *chatbots* para hacer tareas a través del lenguaje natural, esto ha dejado obsoletas aplicaciones e interfaces Web que se utilizaban con el propósito de cubrir estas tareas (búsquedas, reserva de piso, etc.). Pero, ¿qué son los *chatbots* y cómo están ya remodelando el mundo en el que vivimos? Los *chatbots* son software que interactúan con los usuarios proporcionando una infinidad de servicios a través de la comunicación escrita usando el lenguaje natural. Entre estos servicios podemos encontrar: consultar información de manera rápida y precisa, leer y enviar correos, descargar música, películas, o incluso reservar una habitación de hotel o una mesa en un restaurante. Los *chatbots* han revolucionado la manera en la que un ser humano se comunica y realiza tareas a través de una computadora, siendo tal el alcance que grandes empresas como Facebook ya han liberado APIs para el desarrollo de los mismos [1]. Es cuestión de tiempo que otros titanes tecnológicos se sumen a esta tendencia y empiecen a ofrecer sus propios servicios para la creación de *chatbots*.

Productos como Cortana [2] y Siri [3], liberados por Microsoft y Apple respectivamente, sólo procesaban las entradas del usuario y daban como respuesta los resultados de una búsqueda. Estos dos productos serían la primera aproximación a los *chatbots* que se conocen en la actualidad. Los *chatbots* están un nivel por encima de estos productos, debido a que no sólo comprenden lo que el usuario quiere decir sino que construyen una conversación que tiene sentido para el mismo. Hoy en día Siri ya es considerada un chatbot con el que el usuario puede comunicarse y mantener una conversación.

El gran desafío de un *chatbot* es facilitar la comunicación entre los seres humanos y las computadoras a través del lenguaje natural, emulando lo más fielmente posible una conversación entre dos personas. El propósito y objetivo de los *chatbots* varía en función del individuo que lo utilice. Cada persona tiene sus propias necesidades, y por lo tanto, las

plataformas de creación de *chatbots* ofrecen la posibilidad de personalización de los mismos para que estos satisfagan las necesidades concretas del usuario que lo use. Las máquinas pueden aprender cosas, reconocer patrones y tomar decisiones tal y como lo hacen los humanos, sin embargo, todavía tienen carencias en una cosa: cubrir todas las expresiones lingüísticas que usan los seres humanos para comunicarse entre ellos.

Para alcanzar tal objetivo, un chatbot necesita comprender el lenguaje natural y la comunicación escrita utilizada por los seres humanos. Actualmente, la única manera de tener una interfaz de conversación funcional es mediante el uso de una combinación de procesamiento simbólico (aplicación de reglas básicas) sobre el texto. Los *chatbots* requieren aprender con el tiempo la forma en la que los seres humanos se comunican, detectar emociones y mostrar empatía. Para lograr todo esto es necesario el uso de lo que se conoce como Inteligencia Artificial supervisada. El bot reúne información puntual con un humano en concreto, y a medida que el humano responde, éste aprende a responder adecuadamente al usuario.

1.1. Problema a resolver

Los sistemas de diálogo se están convirtiendo en herramientas centrales en los sistemas humano-computadora. En los últimos años ha habido un auge en la creación de agentes que sean capaces de mantener una conversación con sentido entre un usuario y una máquina. De hecho, nuevos sistemas de interacción y nuevas implementaciones de librerías para el control de diálogo surgen a diario, añadiendo nuevas características a estos sistemas a un ritmo vertiginoso.

Las nuevas técnicas de *Machine Learning*, y más recientemente *Deep Learning*, están sustituyendo a las aproximaciones convencionales de programación de Inteligencia Artificial. *Machine Learning* o aprendizaje automático es una rama de la Inteligencia Artificial que otorga a las máquinas una capacidad de aprendizaje sin especificar explícitamente la manera de aprender y de este modo poder crear sus propios patrones y tomar decisiones. Existen una serie de técnicas dentro de este campo centradas en la representación de la información mediante diferentes niveles, para así poder establecer relaciones más complejas entre los datos, agrupadas bajo el nombre de *Deep Learning* o aprendizaje profundo. A esto se suma los incontables avances en cuanto a modelos de redes neuronales. La aparición de modelos de redes neuronales de memoria juegan un papel importante en el aprendizaje de los *chatbots* debido a que permiten guardar información relevante de la conversación y razonar sobre la misma. Esto produce como resultado *chatbots* más precisos y complejos con un alcance

mayor que cuando se utilizan modelos de redes neuronales clásicos[1]. Con la aparición de estos nuevos modelos es necesario encontrar soluciones que utilicen las técnicas conocidas de Procesamiento del Lenguaje Natural y búsqueda de respuestas junto a estos para conseguir *chatbots* que simulen más fielmente una conversación humana.

La comprensión y procesamiento del lenguaje natural (*Natural Language Processing* NLP) es un problema fundamental a resolver para que los *chatbots* sean capaces de interpretar y entender al usuario. Consiste en leer texto y determinar el significado implícito y explícito de cada uno de los componentes, ya sean palabras, frases o párrafos. Este se ha convertido en un problema muy complejo debido a la extensa variabilidad en la formación del lenguaje [4]. Los *chatbots* interpretan los mensajes a través de técnicas convencionales de procesamiento sobre el texto para realizar tareas como la detección de verbos, eliminación de *stop words*[2], extracción de entidades e intenciones, etc. Los avances en *Deep Learning* y la disponibilidad de conjuntos de datos de gran tamaño han dado lugar a la aparición de nuevas tecnologías de gran interés para la comprensión de textos ya que facilitan, aceleran y mejoran las tareas mencionadas anteriormente.

La búsqueda de respuestas (*Question Answering* QA), sigue siendo uno de los desafíos más difíciles al que se enfrenta la ciencia computacional en el procesamiento de lenguaje natural. Su aplicación también alcanza al desarrollo de sistemas de diálogo y *chatbots* para simular y mantener conversaciones humanas. La idea de crear un agente capaz de responder preguntas de dominio abierto, arbitrarias con respecto a documentos de distinta índole, ha cautivado la imaginación de la comunidad científica [5]. Una vez procesado el texto haciendo uso de técnicas NLP, es posible obtener una respuesta concreta y con sentido teniendo en cuenta toda la conversación previa.

1.2. Contexto y motivación

La realización de este trabajo se enmarca dentro de las líneas de trabajo del Grupo de Big Data y Sistemas Cognitivos del Instituto Tecnológico de Aragón (ITAINNOVA). Dentro de este grupo y desde hace varios años, se está investigando y trabajando muy activamente en los ámbitos de la Inteligencia Artificial, Big Data, Web Semántica y Sistemas de la Información. Este trabajo se centra en la utilización de tecnologías de procesamiento de textos y control de diálogo, la búsqueda de respuestas y la aplicación de algoritmos de *Machine Learning*. En

[1] Se entiende por modelo clásico a los compuestos de una entrada, una salida y capas intermedias que modifican la entrada aplicando funciones matemáticas sobre la misma para obtener un valor de salida, no existe recursividad o celdas de memoria.

[2] artículos, pronombres, preposiciones, conectores, palabras que no aporten significado semántico

particular, consiste en el desarrollo de un *chatbot* interactivo usando técnicas de *Deep Learning* para la comprensión de texto y que sea capaz de mantener una conversación simple con el usuario. El *chatbot* debe ser capaz de responder preguntas sobre el corpus de datos abiertos del portal Aragón Open Data, datos relativos al organigrama del Gobierno de Aragón.

La motivación principal del proyecto se fundamenta en la necesidad de mejorar y aportar soluciones en el ámbito de comprensión de textos (*text understanding*) y la búsqueda de respuestas a partir de información no estructurada. En este contexto, el desarrollo de un sistema de comunicación usuario-computadora bidireccional con un *chatbot* interactivo que proporciona al usuario respuestas precisas y con sentido respecto a la conversación con el usuario, se convierte en una aportación tecnológica y producto útil para el grupo como parte de su entorno de trabajo así como punto de partida para futuros proyectos. Actualmente se trabaja con corpus documentales muy grandes a los que de momento se consulta mediante lenguaje de base de datos. El uso de este *chatbot* facilitaría la búsqueda de documentos o datos concretos ya que realizar consultas a través del lenguaje natural es más sencillo que utilizar lenguajes de consulta a base de datos, los cuales requieren un conocimiento técnico mínimo para usarlos adecuadamente.

Además de esto, el desarrollo del *chatbot* se incluirá dentro de un proyecto del departamento, junto a un modelo de avatar virtual propiedad del mismo. En este caso el *chatbot* funciona como Inteligencia Artificial de este avatar virtual, formando un sistema conversacional con una interfaz gráfica que simula una persona humana y es capaz de mantener una conversación con sentido.

1.3. Objetivos

El objetivo principal de este Trabajo de Fin de Grado consiste en desarrollar un *chatbot* que mantenga una conversación con el usuario en la cual sea capaz de responder a preguntas sobre el corpus de datos del abierto de datos del Gobierno de Aragón, así como responder sobre datos del Instituto Tecnológico de Aragón (número de trabajadores, proyectos, colaboraciones e información relacionada con el Instituto). Adicionalmente, que sea capaz de mantener una conversación simple en la que se incluye: preguntar por estado de ánimo, informar del nombre, predicción meteorológica, etc. A continuación se listan los objetivos específicos a alcanzar:

- ▷ Conocer a nivel teórico el estado del arte de *Question Answering* y *Natural Language Processing*.
- ▷ Estudiar herramientas y algoritmos de *Machine Learning* y *Deep Learning*.

- ▷ Elegir las herramientas que mejores características ofrezcan para la implementación del *chatbot*. Desarrollar un *chatbot* mediante el uso de las mismas.
- ▷ Configurar el *chatbot* para acceder y devolver resultados sobre el corpus documental del proyecto *"Aragón Open Data"* y mantener una conversación con el usuario.
- ▷ Programar un servicio Web de comunicación con el *chatbot* a través de una sala de chat similar a cualquier aplicación de mensajería instantánea como WhatsApp.
- ▷ Integrar el *chatbot* junto a un avatar virtual.
- ▷ Validar el sistema de comunicación usuario-computadora bidireccional con el *chatbot* interactivo.

1.4. Estructura de la memoria

La presente memoria expone de forma precisa y descriptiva el trabajo desarrollado, así como los resultados obtenidos y las conclusiones finales. La memoria se estructura en 5 capítulos y un glosario, que se describen brevemente a continuación.

- ▷ **Capítulo 1 - Introducción.** En este capítulo se presenta brevemente la introducción al proyecto, una descripción del problema a resolver, el contexto en el que se enmarca el trabajo, la motivación para la realización del mismo y los objetivos que se pretenden alcanzar. Por último, se describe cómo se organiza la memoria.
- ▷ **Capítulo 2 - Estado del arte.** En este capítulo se describe el estado del arte de todos los componentes del proyecto: redes neuronales de memoria, procesamiento del lenguaje natural y *chatbots*. Se enmarca el trabajo existente y el estado actual de los diferentes componentes involucrados en este trabajo.
- ▷ **Capítulo 3 - Sistema de comunicación usuario-computadora bidireccional con un *chatbot* interactivo.** En este capítulo se plantea la arquitectura del sistema desarrollado, se explica en detalle los módulos que forman el sistema y cómo se integran y comunican entre sí. Además, se incluye una descripción de la implementación llevada a cabo. Por último, se presenta la interfaz de comunicación para determinar cómo el usuario interactúa con el sistema final.
- ▷ **Capítulo 4 - Validación y resultados.** En este capítulo se presentan las pruebas que se han realizado para la validación y verificación del funcionamiento de todos los módulos que componen el sistema. Se incluye también un listado con los problemas y dificultades encontradas durante el desarrollo del proyecto.

- ▷ **Capítulo 5 - Conclusiones, líneas de trabajo futuro y evaluación personal.** En este capítulo se exponen las principales conclusiones que se derivan de la realización de este trabajo y a partir de los objetivos planteados, incluyendo las posibles mejoras y líneas de trabajo futuro.

- ▷ **Glosario.** Descripción de términos usados frecuentemente durante el documento.

Este documento además incluye 7 anexos que completan y amplían la información de todo el trabajo desarrollado en este proyecto:

- ▷ **Anexo I - Descripción del modelo de red neuronal seleccionado.** En este último anexo se incluye una descripción detallada y técnica del funcionamiento del modelo de red neuronal utilizado en el proyecto

- ▷ **Anexo II - Extracción de entidades.** En este anexo se redacta el uso de un extractor de entidades desarrollado por el departamento de Big Data y Sistemas Cognitivos del ITAINNOVA.

- ▷ **Anexo III - *Framework* creación *chatbot* y comprensión de texto.** En este anexo se detalla el funcionamiento de la librería utilizada para el control del diálogo y la creación del *chatbot*.

- ▷ **Anexo IV - Herramientas y tecnologías utilizadas.** En este anexo se listan todas las herramientas y tecnologías utilizadas, así como las ventajas que ha ofrecido su uso para el desarrollo del trabajo.

- ▷ **Anexo V - Pseudocódigo.** Pseudocódigo del funcionamiento del chatbot.

- ▷ **Anexo VI - Integración *chatbot* con avatar virtual.** Descripción del proyecto del avatar junto a imágenes del sistema en funcionamiento.

- ▷ **Anexo VII - Gestión del proyecto.** En este anexo se presenta detalladamente la gestión del proyecto incluyendo el desglose de tareas y horas invertidas por cada una de las tareas.

Capítulo 2

Estado del arte

A lo largo de este capítulo se explica el estado del arte de los diferentes componentes que forman el sistema desarrollado. En primer lugar se introduce el estado del arte de *Machine Learning*, en concreto de las redes neuronales con memoria, desde su primera aparición, pasando por su uso a lo largo de los años, hasta la actualidad. En segundo lugar se describe el estado del arte del procesamiento del lenguaje natural y la búsqueda de respuestas, su funcionamiento junto a modelos de *Machine Learning* y el rol que desempeñan en este trabajo. En último lugar, se explica el estado actual de los *chatbots* y cómo integran tanto las técnicas de *Machine Learning* como técnicas de NLP y QA para formar un sistema de comunicación funcional mediante el uso de lenguaje escrito.

2.1. *Machine Learning*

Dentro de la ciencia computacional existen diversas ramas de la Inteligencia Artificial. Una de ellas es el aprendizaje automático o *Machine Learning*, cuyo objetivo es crear técnicas que permitan a las computadoras "aprender". Una de las técnicas más utilizadas son las redes neuronales, las cuales a partir de una serie de ejemplos de entrenamiento son capaces de generalizar y predecir correctamente a partir de nuevos datos no utilizados durante el entrenamiento.

Las redes neuronales artificiales son algoritmos matemáticos que surgen de la idea de imitar el comportamiento de las redes neuronales biológicas, en particular, las humanas. Una red neuronal está compuesta de al menos dos capas de neuronas, una capa de entrada y una de salida. Además, entre éstas se pueden incluir capas internas. Dentro de las redes neuronales se puede diferenciar dos arquitecturas principales; una primera donde las neuronas de una capa están conectadas con la capa siguiente, y una segunda en la cual la salida de

las neuronas puede estar conectada a la siguiente capa pero también a otras capas de la red, ya sean previas o posteriores. Este tipo de redes son más comúnmente conocidas como redes neuronales recurrentes. La motivación de estas redes es poder contar con una "memoria" en la que se pueda almacenar información previa y con esta, poder realizar predicciones futuras.

En el año 1997 apareció por primera vez el término *Long Short-Term Memory (LSTM)* o redes de gran memoria de corto plazo. Las redes LSTM incorporan una memoria explícita que puede ser actualizada y borrada lo que les permite aprender dependencias de largo plazo en los datos [6]. Los modelos de redes que utilizaban unidades LSTM consiguieron batir récords en comprensión del lenguaje natural, reconocimiento de texto escrito e incluso ganar la competición de escritura manual *International Conference on Document Analysis and Recognition (ICDAR)*[1] en el año 2016 [7].

Las mayores compañías tecnológicas incluyendo Google, Apple y Microsoft utilizan redes *LSTM* como un componente fundamental en sus nuevos productos. Por ejemplo, Google usa *LSTM* para el reconocimiento de voz en los smartphones, para el asistente inteligente Allo [8] y para el traductor de Google [9][10]. Apple usa LSTM para la función "*Quicktype*" de iPhone [11][12] y para Siri [13]. Amazon usa LSTM para Amazon Alexa [14].

Para este proyecto se ha continuado con una línea de investigación del departamento centrada las redes neuronales con memoria, en concreto con el modelo clave-valor. Este modelo se entrena *end-to-end*, entrenamiento basado en aceptar una entrada desde un extremo de la red y producir una salida en el otro. Consta de dos memorias distintas, denominadas memoria clave y memoria valor. El funcionamiento de la red se basa en dos fases, una primera de direccionamiento y almacenamiento en memoria, y una segunda fase de acceso a la memoria para obtener respuesta. Para ver toda la información sobre este modelo, véase el anexo I [15].

Se ha continuado con esta línea por ser un modelo novedoso y con visión de futuro muy prometedora en el campo de la búsqueda de respuestas. Además, el modelo se adecúa muy bien al objetivo del proyecto ya que la entrada del usuario se puede considerar como la clave, y la respuesta como el valor. Dentro del alcance del proyecto, el *chatbot* responde preguntas sobre el corpus de datos abiertos del Gobierno de Aragón. Usando este modelo concreto se puede considerar las preguntas como la clave y la respuesta como el valor.

[1]ICDAR: Conferencia internacional que se celebra cada dos años. Se tratan entre otros, temas de reconocimiento de caracteres, reconocimiento de texto escrito a mano, análisis de documentos, comprensión de textos.

Se ha partido de una implementación de código abierto en `Python`[2] [16], apoyándose en `Tensorflow`, una librería en Python optimizada para el cálculo tensorial y para la creación de modelos de redes neuronales.

Un tensor es cierta clase de entidad algebraica de varios componentes, que describe relaciones lineales entre vectores, escalares u otros vectores. Un par de ejemplos de cálculo tensorial comunes son el producto escalar y el producto vectorial. Un tensor generaliza los conceptos de escalar, vector y matriz de una manera que sea independiente de cualquier sistema de coordenadas elegido. El orden de un tensor será el número de índices necesario para especificar sin ambigüedad una componente de un tensor: un escalar será considerado como un tensor de orden 0; un vector, un tensor de orden 1; y dada una base vectorial, los tensores de segundo orden pueden ser representados por una matriz [17].

El punto de partida del proyecto es un modelo de red neuronal clave-valor listo para ser entrenado y validado con diferentes conjuntos de datos.

2.2. Procesamiento del lenguaje natural

El procesamiento del lenguaje natural, o NLP por sus siglas en inglés, ha ganado recientemente mucha atención por representar y analizar el lenguaje natural computacionalmente. Su aplicación se ha extendido a diferentes campos como traducción, detección de spam, extracción de información y búsqueda de respuestas entre otras.

NLP es un subcampo de la ciencia computacional, ingeniería de la información y de la Inteligencia Artificial dedicado a hacer que los ordenadores comprendan las frases o palabras escritas en un lenguaje humano. Debido a que la forma natural de comunicación de las personas es usando el lenguaje humano, NLP cubre aquellos usuarios que no tienen tiempo para aprender o perfeccionar lenguajes técnicos y a cambio utilizar el más común de los lenguajes usados por los humanos: el lenguaje natural.

Dentro del procesamiento del lenguaje natural se pueden diferenciar dos piezas que lo forman: comprensión del lenguaje (*Natural Language Understanding, NLU*) y generación del lenguaje (*Natural Language Generation, NLG*). A su vez la comprensión del lenguaje natural se compone de diversas piezas: la lingüística, es la ciencia detrás del lenguaje que incluye fonología (referente al sonido); la morfología o formación de palabras; la sintaxis o estructura

[2]Python es un lenguaje interpretado, usa tipado dinámico, esto implica que no hay que especificar el tipo de datos durante la creación de variables sino que el lenguaje sabe interpretar el valor asignado y por lo tanto tratarlo como el tipo de datos correspondiente, también es un lenguaje multiplataforma. Para más información la documentación e información junto a los enlaces de descarga se encuentran en Python

de las frases; la semántica o significado que aportan cada uno de los elementos que componen una frase y la pragmática. Ésta última se refiere a la comprensión del texto. En la figura 2.1 se puede ver una descomposición del procesamiento del lenguaje natural [18].

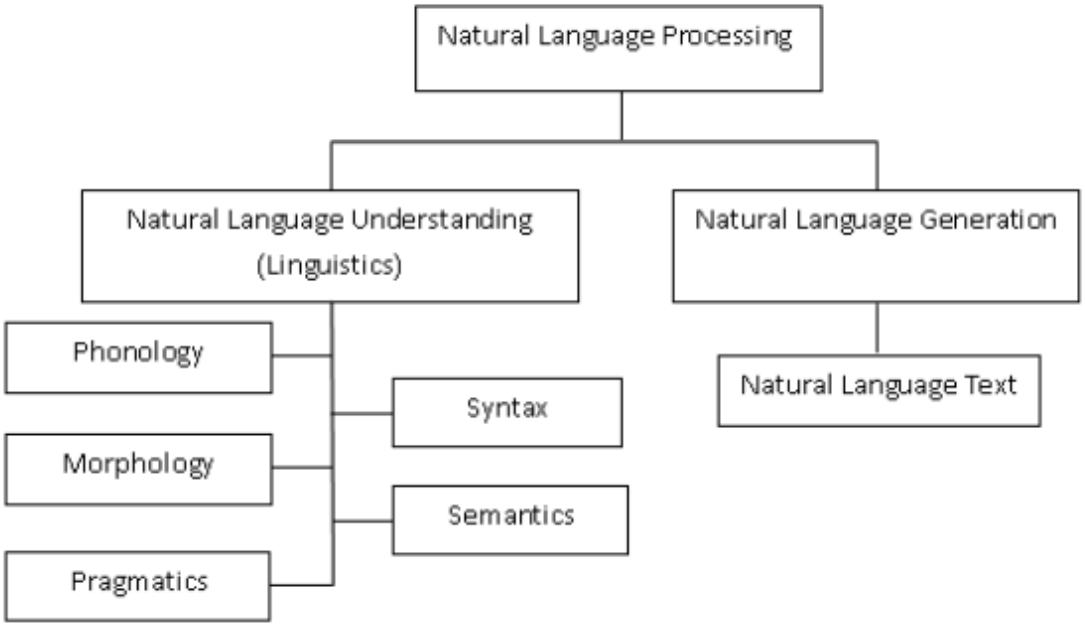

Figura 2.1: Fragmentación NLP ([18])

Una vez introducido brevemente que es el procesamiento del lenguaje natural, se explica el comienzo de este rama de la Inteligencia Artificial, desde dónde y por qué comenzó, hasta los avances más recientes en nuestros días. Por último, se detallará de qué manera el procesamiento del lenguaje natural ha sido integrado en el proyecto, determinando qué labor desempeña dentro del proyecto y cómo se combina con *Machine Learning*.

Durante la década de los 60 se empezó a utilizar la Inteligencia Artificial en algunos trabajos de NLP, como por ejemplo en los sistemas BASEBALL Q-A [19], los cuales respondían preguntas sobre datos de béisbol.

La década de los 80 fue en la que más creció la comunidad científica en estos temas. Aparecieron herramientas y recursos prácticos como *The Alvey Natural Language Tools (Briscoe et al., 1987)* [20]. Las conferencias (D)ARPA de reconocimiento del discurso y comprensión del mensaje (extracción de información) no eran sólo para reconocer los logros que se habían alcanzado sino para marcar un nuevo punto de partida para todo el futuro del procesamiento del lenguaje natural en la siguiente década, los 90.

Desde los 90, todo el foco se puso en combinar técnicas de procesamiento del lenguaje natural con los algoritmos de aprendizaje automático no supervisado o semi-supervisado, con

la intención de conseguir que la máquina aprenda a partir de las conversaciones que mantenga con el usuario. Cada año surgen nuevos modelos de aprendizaje automático, y por lo tanto, un nuevo frente de estudio aparece para combinar estos nuevos modelos con NLP.

En estos últimos 15 años han surgido una serie de herramientas y sistemas que establecen el actual estado en cuanto a NLP. Se han desarrollado herramientas de análisis de sentimientos y *pos tagging* para lenguajes europeos pero también para otros lenguajes como el arábigo o el sánscrito. El análisis de sentimientos se refiere al uso del procesamiento del lenguaje natural, análisis de textos y lingüística computacional para identificar y extraer información subjetiva de unos recursos, esto es, determinar la actitud de un interlocutor o un escritor con respecto a algún tema de un documento. Herramientas de *pos tagging*, proceso de asignar o etiquetar a cada una de las palabras del texto su categoría gramatical (sustantivo, nombre, verbo, etc.). Otras herramientas utilizadas son las de detección de emociones, que funcionan de la misma manera que las de análisis de sentimientos pero orientadas a las redes sociales.

Adicionalmente se usan herramientas de *chunking*, proceso que consiste en dividir un texto en partes sintácticamente correlacionadas de palabras, como son los grupos nominales, grupos verbales, etc, pero no especifica su estructura interna, ni su papel en la oración principal. También se utilizan herramientas de reconocimiento de entidades, tarea de la extracción de información que localiza y clasifica nombres de entidades en unas categorías predefinidas. Éstas resultan difícil de utilizar en Internet debido a que la gente no hace un uso del lenguaje gramaticalmente correcto; se usan abreviaturas, se ignoran letras mayúsculas en nombres, ciudades, etc. Esto degrada considerablemente la actuación de éstas.

Para este proyecto se necesita adaptar y utilizar un reconocedor de entidades[3] ya que los datos del corpus utilizado se componen por nombres de personas, organizaciones y roles que deben ser extraídos como unidades independientes. El *chatbot* tiene que ser capaz de responder a preguntas sobre estos datos, por lo que se necesita tokenizar nombres, organizaciones, roles y fechas como entidades únicas. Además el modelo utilizado requiere que las respuestas sean un único token para usarlo. La tokenización es el proceso que consiste en separar un texto en tokens. En un texto, se denomina token a cada cadena de caracteres separada de otra por un delimitador. Ejemplos de tokens pueden ser, por ejemplo, las palabras.

[3] *NER* (*Named Entity Recognition*) desarrollado por el departamento de Big Data y Sistemas Cognitivos del ITAINNOVA, toda la información sobre este extractor de entidades se puede encontrar en el Anexo II

2.3. *Chatbots*

Las últimas tendencias en cuanto a la oferta de servicios a través de Internet están migrando desde el modelo más común, usar una aplicación concreta que resuelva un propósito específico, a ofrecer este mismo servicio mediante el uso de *chatbots* a través de plataformas de mensajería ya sean aplicaciones conocidas como *Telegram, Facebook*, hasta interfaces Web.

Esta tendencia comenzó en la década de los 90 con el desarrollo de un lenguaje de programación: AIML [21]. Fue diseñado específicamente para ayudar en la creación de la primera entidad *chatbot* informática de lenguaje artificial online o A.L.I.C.E., siglas en inglés de *Artificial Linguistic Internet Computer Entity Chatterbot.*

Los *chatbots* son software que interactuan con los usuarios mediante el uso de infraestructuras (aplicaciones, interfaces web) proporcionando una múltiple variedad de servicios, desde reservar una habitación de hotel o una mesa de restaurante, hasta descargar contenido multimedia de la Web o incluso realizar la compra online. Cada *chatbot* es personalizable, se programa en función de las funcionalidades que debe ofrecer, por ejemplo, no tiene sentido preguntar el precio de una habitación para cuatro personas en un *chatbot* de una pizzería.

La interacción con los *chatbots* se realiza de la misma manera que se haría con una persona en una plataforma de chat, mediante la introducción de texto usando lenguaje natural. Esto supone una gran ventaja frente a aplicaciones específicas porque a diario y con el paso del tiempo, cada vez de manera más frecuente, millones de personas utilizan Internet para realizar sus tareas, como las mencionadas anteriormente.

La utilización de *chatbots* es una manera de facilitar todas estas tareas. Un solo *chatbot* puede ofrecer las mismas funcionalidades que varias aplicaciones o webs específicas. Además, el uso del lenguaje natural se desarrolla y aprende con los años, por lo que los humanos tienden a automatizar y optimizar su uso. Por ejemplo, utilizando abreviaturas, objetos directos para referenciar algo de la conversación que ya se ha especificado y no es necesario repetir, etc.

La existencia de una aplicación que sea capaz de entender el lenguaje natural, ofrece una clara ventaja sobre las aplicaciones convencionales debido a que los seres humanos están muy acostumbrados a expresarse con lenguaje natural ya sea mediante vía oral o escrita. Incluso la lectura forma parte de este desarrollo del uso del lenguaje. Al usar estos *chatbots* se evita tener que entender y comprender el uso de aplicaciones y webs específicas. Simplemente usándolos, como si de un contacto más de nuestro teléfono se tratara, se consigue el mismo resultado más rápido, se usa un único *chatbot* para esas tareas que requerían una aplicación concreta.

El hecho de que sean servicios, mayormente, accesibles a través de la web es considerada otra ventaja, ya que no requiere espacio de almacenamiento en el dispositivo desde el que se use a diferencia de las aplicaciones para *smartphones* o las aplicaciones web para las computadoras. Que sean servicios también implica que los desarrolladores pueden actualizarlo sin suponer ningún tipo de coste de espacio para el usuario ni actualización en su dispositivo. Una vez el usuario se haya conectado al servicio web, tendrá a su alcance la última versión del *chatbot* estable.

En la actualidad se pueden diferenciar tres tipos de *chatbots* [22]: basados en botones, basados en palabras clave y contextuales. A continuación se listan las características de cada uno de ellos en orden ascendente de complejidad (véase Figura 2.2):

- **Basados en botones:** Son los bots más básicos que se pueden encontrar en el mercado hoy en día. En la mayoría de casos se basan en un árbol de decisiones presentado al usuario en forma de botones. A pesar de ser capaces de dar soporte al 80 % de las preguntas más frecuentes que se realizan, son muy lentos para llegar a la respuesta deseada por el usuario. Al estar basado en un árbol de decisiones, cada botón sólo puede determinar un factor de la tarea mientras que otros *chatbots* basados en la comprensión del lenguaje, pueden determinar varios factores importantes con sólo un mensaje del usuario. Para tareas que requieren especificar muchos factores claves, como por ejemplo reservar un hotel (número de personas, estrellas, tiene o no tiene piscina, zona, rango de precios, etc.) sólo pueden detectar un factor por botón. Algunos incluso necesitan de dos, por ejemplo, fecha de entrada y salida.

- **Reconocimiento de palabras clave:** Estos pueden entender lo que los usuarios escriben y responder apropiadamente, o al menos intentarlo extrayendo palabras clave y utilizando Inteligencia Artificial para determinar una respuesta apropiada al usuario. Por ejemplo si un usuario introduce: "¿Qúe tiempo hará mañana en Zaragoza?", el *chatbot* extrae las palabras relevantes: tiempo, mañana, Zaragoza para determinar la mejor respuesta a la pregunta. Este tipo de *chatbots* fallan cuando tienen que responder preguntas similares, ya que puede tener problemas si existe redundancia de palabras. Por ejemplo, con el ejemplo anterior, si un usuario introdujera: "¿Cuándo empieza el primer tiempo del partido del Zaragoza mañana?", se obtendrían las mismas palabras claves que en el caso anterior y el bot podría no responder adecuadamente. Una tendencia muy popular es encontrarse *chatbots* híbridos que utilicen tanto botones como entrada textual directamente para mantener una conversación. Estos proporcionan la posibilidad al usuario de introducir la pregunta directamente o utilizar un menú de botones si las respuestas no están siendo adecuadas.

– **Contextuales:** Son los más avanzados de los tres. Utilizan *Machine Learning* e Inteligencia Artificial para recordar conversaciones con los usuarios y así mejorar las respuestas con el tiempo. A diferencia de los *chatbots* basados en el reconocimiento de palabras clave, éstos son lo suficientemente inteligentes y a través de técnicas de aprendizaje automático son capaces de mejorar basándose en la conversación con el usuario. Este tipo de *chatbots* se basan en la importancia de los datos, es decir, tener una base de datos que guarde el contexto de las conversaciones antiguas mediante técnicas de aprendizaje automático como redes neuronales de memoria facilita conversaciones futuras acortando la interacción en conversaciones futuras. De esta forma, se tiene un sistema que aprende con cada conversación diferente, pero que es capaz de guardar el contexto de las mismas, funcionando en el caso de que estas se repitan como una mera interfaz de usuario [23].

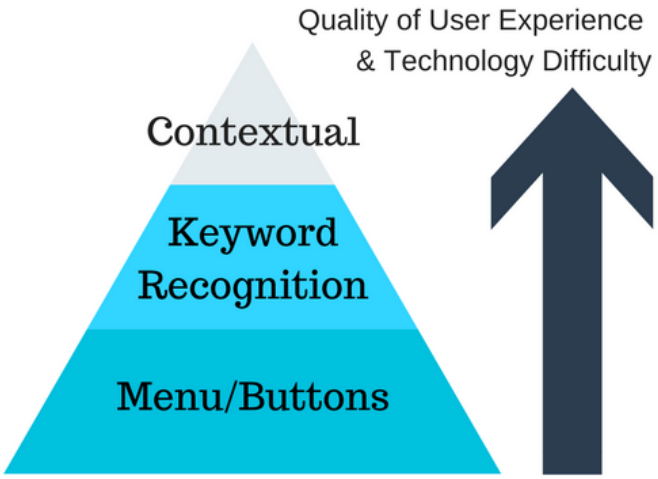

Figura 2.2: Tipos de bots ([22])

2.4. Conclusiones del estado del arte

El objetivo final de este proyecto es conseguir un *chatbot* que utilice técnicas de *Machine Learning* o aprendizaje automático, procesamiento de lenguaje natural y búsqueda de respuestas y que sea capaz de mantener una conversación con sentido con el usuario. En los apartados anteriores ya se ha establecido qué técnicas tanto de aprendizaje automático como de procesamiento de lenguaje se van a utilizar y combinar para conseguir un *chatbot* funcional y que cumplan las especificaciones del proyecto.

Dentro de las técnicas de *Machine Learning* se va a utilizar un modelo de red neuronal clave-valor para responder preguntas sobre el corpus de datos abiertos del portal "Aragón Open Data". Para el procesamiento del texto se va a utilizar un extractor de entidades desarrollado por el departamento que permite la correcta tokenización de los datos. Adicionalmente se usa una librería de detección de expresiones temporales (pasado mañana, el próximo martes, etc.).

Para el procesamiento del lenguaje natural se ha decidido usar un extractor de entidades para el reconocimiento de entidades tales como nombres de persona, roles de trabajo y organizaciones.

Se ha optado por desarrollar un chatbot contextual ya que este tipo de *chatbot* ofrece mayor abanico de posibilidades para trabajar con algoritmos de *Machine Learning*, aprenden a conversar a través de estos algoritmos. Además, este tipo de *chatbots* suelen integrar a los otros dos, esto permite que, en un futuro se puede transformar de manera más sencilla del *chatbot* a un sistema híbrido, incluyendo toma de decisiones con botones dejando al usuario el poder de decidir de que manera llegar al resultado que desee.

Para la implementación del *chatbot* se ha utilizado la librería *open-source* Rasa [24]. Es un *framework* de código abierto, altamente escalable en *Machine Learning*, útil para la construcción y creación de software conversacional. Este *framework* se compone de dos módulos separados: un primer módulo para el tratamiento del texto y su procesamiento (Rasa *NLU*), realizando tareas de extracción de entidades y detección de intenciones; y un segundo módulo (Rasa Core), que se encarga del control del diálogo. Después de comprender lo que el usuario quiere decir, es necesario responder adecuadamente al usuario, esta respuesta se basa en lo que el usuario ha dicho y en el conocimiento previo (misma conversación o entrenamiento del *chatbot*). Toda la información detallada acerca del funcionamiento de esta librería se encuentra en el Anexo III.

Todas las herramientas utilizadas se pueden encontrar en el anexo IV

Capítulo 3

Sistema de comunicación usuario-computadora bidireccional con un *chatbot* interactivo

En el presente capítulo se describe la arquitectura, desarrollo e implementación del sistema de comunicación usuario-computador que se ha denominado Pilar. Se ha elegido este nombre en honor a la ciudad de Zaragoza y Aragón.

Se ha utilizado una arquitectura de tres módulos y dos base de datos. Estos tres módulos son el modelo del red neuronal, el *framework* conversacional, el cual forma el núcleo del sistema, y un traductor de lenguaje natural a lenguaje de consulta de base de datos. El sistema se completa con dos bases de datos, una primera base de datos de indexación que guarda todos los datos extraídos por procesos de *crawling*[1], y una segunda base de datos semántica que guarda información relevante estructurada en tripletas (sujeto, predicado, objeto). Los dos primeros módulos se ejecutan de manera secuencial, es decir, si el primero falla, el segundo es el encargado de dar la respuesta al usuario. El segundo módulo se apoya en un tercero para responder preguntas sobre el corpus. La arquitectura final del sistema se puede apreciar en la figura 3.1.

La implementación se ha realizado siguiendo una metodología iterativa e incremental:

1ª iteración: Estudio teórico del funcionamiento del *framework* conversacional Rasa. Implementación de un una primera versión de un *chatbot* muy simple para ver el funcionamiento de este framework de una manera práctica. Esta primera versión es capaz de saludar, despedirse y preguntar por el estado de animo, respondiendo en función de dos valores: alegre y triste. El *chatbot* desarrollado con este *framework* es el módulo principal de nuestro sistema y se encuentra en segunda posición dentro del sistema por razones que se explican en este apartado.

[1] Automatización de la extracción de datos de diferentes páginas Web

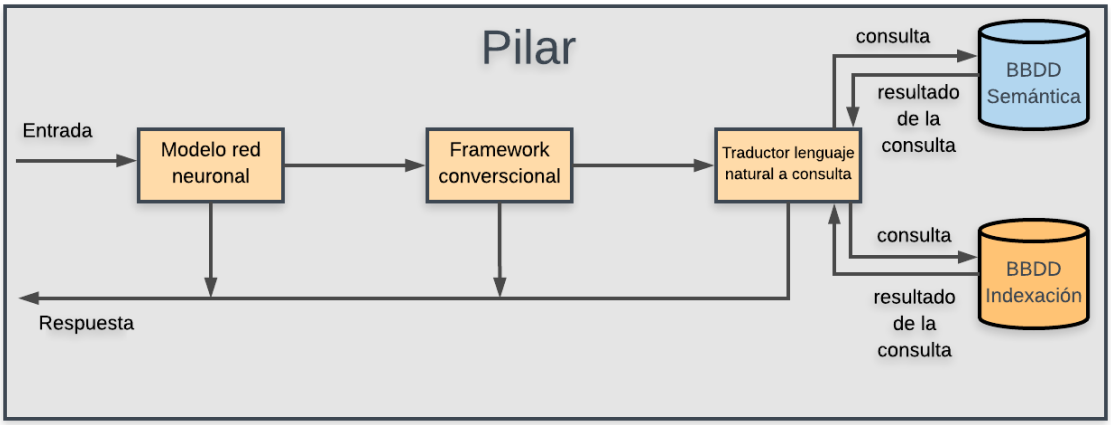

Figura 3.1: Arquitectura del *chatbot*

2ª iteración: Trabajo con el modelo de red neuronal clave-valor especificado en la sección 2.1. Esta red neuronal conforma el segundo módulo principal del sistema y se encarga de responder preguntas referentes a los datos del corpus utilizado. Al ser menor el alcance de este módulo, para dar una respuesta adecuada a la conversación puesto que está limitado a un tipo de pregunta en concreto, se ha optado porque este módulo sea el que ocupe la primera posición en el sistema, es decir, sea el primer módulo en procesar la entrada del usuario y de esta manera, si este no es capaz de responder adecuadamente, enviar la entrada del usuario al núcleo de nuestro sistema, encargado del control de la conversación.

En la figura 3.1 se puede observar un tercer módulo que sirve de apoyo al *framework* conversacional. En ocasiones, no sólo se requiere identificar que quiere decir el usuario y responder con un mensaje; a veces es necesario utilizar APIs externas para algunas tareas como buscar en una base de datos, reservar un hotel, consultar el tiempo. El *chatbot* final tiene que ser capaz de responder preguntas sobre el corpus dando la respuesta correcta. Es en estas preguntas donde interviene este tercer módulo. La red neuronal no es un sistema 100 % eficaz y puede no responder adecuadamente en algunas ocasiones sobre preguntas del corpus. Este módulo cubre estas situaciones accediendo directamente a las bases de datos.

Este tercer módulo se trata de un *framework* programado en `Python` para transformar preguntas en lenguaje natural a consultas en lenguaje de base de datos. Se llama QuePy por la combinación de consulta (*query* en inglés) y el lenguaje en el que está implementado [25]. Es muy personalizable para diferentes tipos de preguntas en lenguaje natural y consultas a base de datos. Actualmente provee funcionalidad para SPARQL y MQL (lenguajes de consulta a base de datos de ontología [26]). Se ha hecho uso de una versión adaptada por el departamento durante el año 2017 utilizada sobre los datos del corpus, usando SPARQL como lenguaje de consulta. Existe un servicio Web sobre el que se pueden realizar consultas

en lenguaje natural. El *framework* realiza la consulta a la base de datos semantizada donde se tiene toda la información relevante estructurada sobre el organigrama del Gobierno de Aragón. En caso de no obtener resultado de esta base de datos, se realiza la misma consulta a la base de datos de indexación.

3ª iteración: Integración de los dos módulos desarrollados en las iteraciones previas junto al desarrollo de un servicio Web mediante el cual el usuario se pueda comunicar con el *chatbot*. Para ello se ha diseñado una página Web que simula una sala de chat virtual con un aspecto similar a la interfaz gráfica de Telegram (aplicación de mensajería instantánea). Dentro de esta iteración también se ha incluido el *chatbot* dentro del proyecto del avatar virtual del departamento de Big Data y Sistemas Cognitivos. El chatbot es la inteligencia que hay detrás del avatar. Se puede decir que es el "cerebro" del avatar.

3.1. Modelo red neuronal

Como ya se ha comentado en el estado del arte (véase sección 2.1), se ha utilizado un modelo de red neuronal con memoria para continuar con una de las líneas de investigación del departamento en cuanto a la búsqueda de respuestas. En concreto, se ha utilizado un modelo de red neuronal clave-valor [15], considerando la pregunta como la clave y la respuesta como el valor.

Se ha partido de una implementación de código abierto disponible en GitHub e implementada en el lenguaje de programación `Python` [16]. En el Anexo I se encuentra una descripción detallada del modelo clave-valor, incluyendo las matemáticas involucradas, los accesos a la memoria clave y a la memoria valor.

A continuación se va a describir la representación de la información utilizada para entrenar el modelo clave-valor. En el estado del arte se ha mencionado que era necesario crear datos de entrenamiento a partir de los datos recogidos por el corpus utilizado. El entrenamiento recoge datos para un total de 923 personas; relativos a:

1. Organización a la que pertenece
2. Rol que ocupa dentro de la organización
3. Fecha de inicio de desempeño del rol
4. Fecha de finalización del período de trabajo

Con esta premisa, se han definido cuatro preguntas "tipo" que abarcan toda la información disponible para cada una de las personas (véase Tabla 3.1).

¿Dónde trabaja *persona*?
¿Qué puesto ocupa *persona*?
¿Cuándo empezó a trabajar *persona*?
¿Cuándo terminó de trabajar *persona*?

Tabla 3.1: Preguntas para *persona*

A partir de estas preguntas, se han definido las frases que contienen la respuesta a las mismas. Al ser la primera vez que se trabajaba con estos datos y este modelo en concreto, se han creado datos muy sencillos en los que sólo existiese una frase que respondiera a cada una de las preguntas. De esta manera para cada una de las personas se proporcionaban las siguientes frases "tipo" (véase Tabla 3.2):

persona trabaja en *organización*
persona ocupa el puesto de *rol*
persona empezó a trabajar el $fecha\_inicio$
persona terminó de trabajar el $fecha\_fin$

Tabla 3.2: Frases "tipo" para *persona*

A partir de las frases y las preguntas era posible determinar la respuesta a las preguntas, véase Tabla 3.3:

Pregunta	**Respuesta**
¿Dónde trabaja *persona*?	*organización*
¿Qué puesto ocupa *persona*?	*rol*
¿Cuándo empezó a trabajar *persona*?	$fecha\_inicio$
¿Cuándo terminó de trabajar *persona*?	$fecha\_fin$

Tabla 3.3: Preguntas y respuestas frases "tipo"

Para poder usar la implementación elegida, las respuestas tienen que ser un único token. Esta tarea de tokenización y obtención de entidades referentes a *persona*, *organización*, *rol*, $fecha\_inicio$ y $fecha\_fin$, se realiza a través del extractor de entidades (véase sección 2.2 y anexo II).

Con estos datos se creó un fichero de entrenamiento. Una apariencia de este fichero se puede observar en la Figura 3.2:

Con todas las palabras de los datos, se ha creado un diccionario ordenado alfabéticamente. A cada palabra se somete a un proceso de des-acentuación y se le asigna el índice de su posición en el diccionario. La des-acentuación se realiza debido a que un usuario no tiene por qué acentuar las palabras y es de obligatorio cumplimiento que la entrada del usuario se empareje con las palabras guardadas en el diccionario. Por lo tanto, cada entrada del usuario sufrirá el mismo proceso. Se asegura así utilizar siempre el mismo valor

```
1 Antonio Alastrué Tierra trabaja en, Dirección General de Trabajo
2 ¿Dónde trabaja Antonio Alastrué Tierra?  Dirección General de Trabajo  1
3 Antonio Alastrué Tierra ocupa el puesto de, DIRECTOR GENERAL DE TRABAJO
4 ¿Qué puesto ocupa Antonio Alastrué Tierra?  DIRECTOR GENERAL DE TRABAJO 3
                                    .
                                    .
                                    .
3014 ¿Cuándo empezó a trabajar Maria Antonia García Huici?  2011-07-16  3013
3015 Maria Antonia García Huici terminó de trabajar el 2015-07-04
3016 ¿Cuándo terminó de trabajar Maria Antonia García Huici?  2015-07-04  3015
3017 Elena Marquesán Díez trabaja en Servicio de Estudios Autonómicos
3018 ¿Dónde trabaja Elena Marquesán Díez?  Servicio de Estudios Autonómicos  3017
                                    .
                                    .
                                    .
```

Figura 3.2: Fichero de datos de entrenamiento

sin importar como el usuario escribe el texto. El diccionario que contiene todas las palabras ordenadas junto al índice correspondiente tiene el formato presentado en la Figura 3.3.

```
DICCIONARIO:
{
 '1978-04-09': 1, '1979-05-21': 2, '1980-04-28': 3,
 ...
 'antonio alastrué tierra': 96, 'antonio angulo borque': 97,
 ...
 'dirección general de trabajo': 325,
 ...
 'trabaja': 1792,
 ...
}
```

Figura 3.3: Diccionario de palabras-valor

Cada una de las palabras es sustituida por su número asociado en el diccionario. Las respuestas se codifican como vectores *one-hot*[2]. Véase Tabla 3.4 como ejemplo de *one-hot*.

a	azul	casa	donde	es	la	roja	tiene
0	1	1	0	1	1	0	0

Tabla 3.4: Codificación *one-hot* para la frase: **La casa es azul**

Las dimensiones de los tensores en los que se almacenan las historias, las preguntas y las respuestas se van a describir de manera abstracta sin valores concretos debido a que esta implementación puede ser utilizada para múltiples conjunto de datos.

[2]Codificación categórica de los datos, vectores de 1s y 0s.

– **Historias (x, y, z)**

1. x es el número de preguntas del conjunto
2. y es la longitud de la historia más larga en número de tokens
3. z es la longitud de la frase más larga, considerando tanto historia como preguntas

– **Preguntas (x, y)**

1. x es el número de preguntas
2. y es la longitud de la frase más larga

– **Respuestas (x, y) vector de vectores *one-hot***

1. x es el número de respuestas
2. y es el número de palabras en el diccionario

Dado que `Tensorflow` no tiene implementado trabajar con matrices irregulares, todas aquellas frases de los datos que por su tamaño de frase no alcancen a cubrir el tamaño máximo del vector, se rellenará con 0s hasta cumplir con el tamaño establecido. Se sigue el mismo procedimiento para las historias.

Para que el módulo de red neuronal responda, la entrada del usuario tiene que cumplir una serie de requisitos:

1. Comprobar si es una pregunta verificando si hay signos de interrogación.
2. No debe superar la longitud máxima que soporta la red como parámetro de entrada.
3. Al menos tres tokens de la entrada tienen que estar en el diccionario.
4. De los tres tokens, uno debe estar dentro de los 923 nombres de personas entrenados.

En la Tabla 3.5 hay un muestrario de entradas de ejemplo junto al motivo por el que son o no son entradas válidas.

Hola	✗	No es pregunta
¿En que año nació el inventor de la imprenta?	✗	No cumple restricción de tamaño
¿Dónde trabaja Álvaro Monteagudo?	✗	Falta nombre del diccionario
¿Dónde estudia Antonio Alastrué Tierra?	✗	Faltan palabras del diccionario
¿Que puesto ocupa Antonio Alastrué Tierra?	✓	

Tabla 3.5: Entradas de ejemplo

3.2. Control de diálogo

Dentro de los agentes virtuales existen dos pilares fundamentales: la comprensión del texto (extracción de entidades, eliminación de *stopwords*, etc.) y el control del diálogo, es decir, comprender el orden de la conversación, mantener las palabras más relevantes de la misma y construir una conversación que tenga sentido para el usuario.

Para el control del diálogo se ha hecho uso de la librería Rasa basada en el lenguaje de programación Python. Se ha decidido optar por esta librería debido a que es *open-source*, lo que quiere decir que es gratis de usar y con el código disponible en red. Además, es una de las tecnologías nuevas de gestión de diálogo más completa y con desarrollo continuo, tienen un foro *online* para responder cuestiones sobre el uso de Rasa. Y por último, Rasa no utiliza servidores de Microsoft o Google, por lo que se puede utilizar en entornos internos que requieren mantener la confidencialidad de los datos.

Esta librería se compone de dos módulos separados: un primer módulo para el tratamiento del texto y su procesamiento (*Rasa NLU*), que realiza tareas de extracción de entidades y detección de intenciones (una intención se puede definir como el significado del mensaje del usuario), recibe texto en formato libre y lo convierte en datos estructurados; y un segundo módulo que se encarga del control del diálogo (*Rasa Core*). Este último módulo mantiene una conversación con sentido para el usuario, actúa como gestor del diálogo, manteniendo el flujo de la conversación y decidiendo como proceder con la misma. En esta sección se introduce el trabajo realizado con Rasa, para obtener información más detallada acerca del funcionamiento de esta librería véase el Anexo III.

Tras haber estudiado el funcionamiento de Rasa, se implementó un *chatbot* muy básico. Este primer *chatbot* estaba programado para saludar, preguntar por el nombre, por tu estado de ánimo, reaccionar a este, darte la bienvenida por tu nombre y despedirse. Los pasos a seguir para la implementación de esta versión son los siguientes: usar Rasa Core para definir historias, intenciones y acciones; usar NLU para la comprensión del lenguaje sobre los datos de ejemplo.

El primer componente necesario para construir un *chatbot* con Rasa es definir el flujo de la conversación, qué intenciones sabe identificar y cómo responde a las mismas. Para esto se utiliza el módulo Core de la librería. El primer paso es escribir una serie de historias, de esta manera el *chatbot* es capaz de aprender de las mismas y generalizar. Una historia se define como una intención seguida de una serie de acciones que el *chatbot* tiene que realizar una

vez identificada esa intención. El formato de las historias se define usando *Markdown*[3]. Las frases que comienzan con '`##`' son la cabecera de la historia (útil para etiquetar y diferenciar historias), las líneas que comienzan por '$*$' son las intenciones detectadas por el *chatbot* en función de una entrada del usuario, y las que comienzan por '$-$' son las acciones que lleva a cabo el *chatbot*. Normalmente, las acciones son mensajes de respuesta hacia el usuario, también llamados *utterances*, pero en general una acción (*action*) puede hacer cualquier tipo de labor, incluyendo llamadas a una API externa, bases de datos, etc. Véase Figura 3.4.

En la figura 3.4 también se puede observar una historia que no tiene intención declarada, esta historia es predeterminada de Rasa, denominada '`## fallback`'. Esta historia cubre la función de responder a entradas del usuario que el bot no está programado para responder por su configuración. Esta historia responde con un mensaje: "No le he entendido" y se ejecuta cuando el *chatbot* no tiene un porcentaje de identificación de una intención por encima de un umbral determinado.

```
## Welcome story
* greet
   - utter_welcome
## Goodbye story
* goodbye
   - utter_goodbye
## fallback
   - action_fallback
```

Figura 3.4: Ejemplo historias

El segundo componente necesario para el *chatbot* es el dominio. El dominio determina todas las intenciones que el *chatbot* sabe identificar, todas las acciones que puede realizar y la información que puede guardar. Ésta viene definida por entidades y *slots*, las primeras son piezas de información que se desea extraer de los mensajes, y los segundos son el espacio donde almacenar estas piezas. Dentro de las respuestas al usuario podemos ver que se han declarado diferentes *utterances* pero también se ha declarado una acción personalizada (`actions.ActionFallback`). En la Figura 3.5 se puede ver el dominio de este primer *chatbot*.

Para declarar una acción personalizada es necesario declarar una clase con el nombre definido en el dominio (ActionFallback) la cual contenga dos métodos: un método que devuelva el nombre de la acción (éste tiene que corresponderse con el nombre utilizado en las historias), y un segundo método con la función de la acción, ya bien sea devolver un mensaje como en este caso, realizar cálculos, buscar datos, etc. Véase Listing 3.1.

[3]Lenguaje de marcado que tiene como objetivo el hacer más fácil la tarea de dar formato a un texto mediante el uso de algunos caracteres. Un lenguaje de marcado es un tipo de formateo de texto más o menos estandarizado, que ocupa poco espacio y es fácil de editar con un editor de texto.

```
intents:
  - greet
  - goodbye
  - inform
  - happy
  - sad

actions:
- utter_goodbye
- utter_welcome
- utter_welcome_name
- utter_happy
- utter_sad
- actions.ActionFallback

entities:
- name

slots:
  name:
    type: unfeaturized

templates:
  utter_goodbye:
  - text: "Adiós"
  - text: "Hasta la próxima"
  - text: "Esperamos verle de nuevo por aquí"
  utter_welcome:
  - text: "Bienvenido al asistente de búsqueda de Aragón OpenData. ¿Como se llama?"
  utter_welcome_name:
  - text: "Bienvenido {name}. ¿Cómo estas?"
  utter_happy:
  - text: "Me alegro"
  utter_sad:
  - text: "¿Qué puedo hacer para ayudarte?"
```

Figura 3.5: Ejemplo dominio

```
from rasa_core.actions.action import Action
from rasa_core.events import SlotSet

class ActionFallback(Action):
    def name(self):
        return 'action_fallback'

    def run(self, dispatcher, tracker, domain):
        dispatcher.utter_message("No le he entendido")
        return []
```

Listing 3.1: Acción personalizada

El núcleo del *chatbot* es capaz de reconocer intenciones, responder a éstas, realizar acciones específicas e incluso tiene reservado espacio para guardar información de la conversación. Sólo falta añadirle textos de ejemplo para que sepa diferenciar entre las diferentes intenciones y de ese modo identifique adecuadamente qué quiere decir el usuario y responder acordemente. Hasta ahora, el *chatbot* sólo sabe lo que puede hacer pero no tiene ningún ejemplo para diferenciar entre responder un "Hola" o un "Adiós".

A continuación se tiene que configurar el *chatbot* para que comprenda el lenguaje natural. Para ello es necesario establecer un intérprete que sea capaz de comprender los mensajes y transformarlos en datos estructurados comprensibles por el *chatbot*. Aunque existen otras alternativas para el procesamiento del lenguaje, para este proyecto se ha utilizado Rasa NLU, ya que para el control del diálogo se ha hecho uso del módulo Core y así se usaba toda la librería al completo.

Con Rasa NLU es necesario definir mensajes del usuario que el *chatbot* tiene que entender. No es necesario incluir todos los mensajes posibles, sólo unos casos para que el *chatbot* sepa inferir con otros nuevos. Para la generación de datos, se ha utilizado una herramienta online llamada Chatito [27] ya que posee una interfaz Web que permite transformar texto etiquetado en intenciones, entidades y frases de ejemplo. La generación de datos produce un resultado como el mostrado en la Figura 3.6, en la que se puede observar que los mensajes están asociados con una intención, con los valores referentes a las entidades buscadas con posición inicial, final y valor de la entidad.

La última pieza del *chatbot* es la configuración del modelo NLU, esto quiere decir, qué procesos sufren los datos de entrada del usuario para extraer intenciones y entidades tal y cómo se tiene en los datos de ejemplo. Estos componentes pueden ser muy diversos, entre ellos se puede definir un extractor de entidades, el lenguaje del modelo a utilizar (corresponde con el lenguaje que se espera que utilice el usuario), librerías externas para extraer fechas como *Duckling*[4], la forma de tokenizar los datos, ya sea mediante palabras, frases, etc, por nombrar algunos de ellos. Para cada *chatbot* se requiere una configuración diferente en función de la finalidad de éste. Para esta primera versión se ha utilizado la configuración de la Figura 3.7, es una configuración sencilla para un modelo de comprensión del lenguaje en castellano.

Con todas estas piezas se pasa al entrenamiento del modelo *NLU* y el entrenamiento del control de la conversación (*Core*). Véanse listings 3.2 y 3.3.

[4]Biblioteca escrita en Clojure que extrae entidades temporales dentro de un texto, e.g. pasado mañana, las dos del mediodía, dentro de una semana, etc.

```
{
  "rasa_nlu_data": {
    "regex_features": [],
    "entity_synonyms": [],
    "common_examples": [
      {"text": "Hola", "intent": "greet", "entities": []},
      ...
      {
        "text": "Me llamo Lucia",
        "intent": "inform",
        "entities": [
          {"end": 14, "entity": "name", "start": 9, "value": "Lucia"}
        ]
      },
      ...
      {
        "text": "Mi nombre es Fran",
        "intent": "inform",
        "entities": [
          {"end": 17, "entity": "name", "start": 13, "value": "Fran"}
        ]
      },

      {"text": "Estoy bien", "intent": "happy", "entities": []},
      ...
      {"text": "Mal", "intent": "sad", "entities": []},
      ...
      {"text": "Nos vemos", "intent": "goodbye", "entities": []},
      ...
    ]
  }
}
```

Figura 3.6: Ejemplo datos de entrenamiento *NLU*

```
language: "es"
pipeline:
- name: "nlp_spacy"
  model: "es"
  case_sensitive: true
- name: "tokenizer_spacy"
- name: "intent_featurizer_spacy"
- name: "intent_entity_featurizer_regex"
- name: "intent_classifier_sklearn"
- name: "ner_crf"
```

Figura 3.7: Configuración básica para un modelo *NLU* en Español

```
def train_nlu(training_data, config_file, model_dir, fixed_model_name):

    from rasa_nlu.training_data import load_data
    from rasa_nlu.model import Trainer
    from rasa_nlu import config

    training_data = load_data(training_data)
    trainer = Trainer(config.load(config_file))
    trainer.train(training_data)
    model_directory = trainer.persist(model_dir,
                                      fixed_model_name=fixed_model_name)
    return model_directory
```

Listing 3.2: Entrenamiento modelo NLU

```
def train_dialogue(domain_file, model_path, training_data_file):

    fallback = FallbackPolicy(fallback_action_name="utter_default",
                              core_threshold=0.25, nlu_threshold=0.3)

    agent = Agent(domain_file,
                  policies=[AugmentedMemoizationPolicy(max_history=2),
                            KerasPolicy(), fallback])

    training_data = agent.load_data(training_data_file)
    agent.train(training_data, epochs=300, batch_size=64,
                validation_split=0.2)

    agent.persist(model_path)
    return agent
```

Listing 3.3: Entrenamiento diálogo (Core)

En la Figura 3.8 se puede observar una pequeña conversación con esta primera versión del *chatbot*. Para el proyecto se han añadido más datos de entrenamiento y todo lo necesario para que este sea capaz de reconocer preguntas relacionadas con el corpus de datos abiertos (historias, acciones, intenciones, etc).

```
Hola
Le doy la bienvenida al asistente de búsqueda de Aragón OpenData.
¿Podría decirme su nombre?
Me llamo Álvaro
Bienvenido Álvaro. ¿Como estas?
Muy bien
Me alegro
Adiós
Esperamos verle de nuevo por aquí
```

Figura 3.8: Conversación de ejemplo con el *chatbot*

3.3. *Framework* traductor lenguaje natural a consulta de base de datos

Para responder a las preguntas sobre el corpus desde Rasa, se ha utilizado un *framework* de Python llamado QuePy [25], adaptado y configurado por el departamento de Big Data y Sistemas Cognitivos del ITAINNOVA. Este *framework* permite transformar preguntas en lenguaje natural a consultas en lenguaje de base de datos. Actualmente provee funcionalidad para SPARQL y MQL (lenguajes de consulta a base de datos). Este framework se usa a través de una acción personalizada cuando el modelo de red neuronal no ha sabido responder a une pregunta sobre los datos del corpus. Con este problema resuelto se consigue tener un sistema completo funcional y que cumple los objetivos del proyecto.

3.4. Interfaz

Para la comunicación con el *chatbot* se ha implementado una interfaz gráfica que simula una sala de chat. Se ha utilizado HTML, CSS y JavaScript para el diseño de la Web.

Se ha partido de una plantilla de Bootstrap[5] [28] basada en una pantalla principal con una caja de texto y un botón, a través del cual se envía un mensaje al *chatbot*. Inicialmente, una vez cargada la aplicación Web, el chat proporciona un mensaje de bienvenida al usuario enumerando las capacidades qué tiene y que tipo de mensajes entiende el *chatbot*. Un ejemplo de conversación simple se encuentra en la figura 3.9 y 3.10.

El pseudocódigo del sistema completo se puede encontrar en el anexo V, este pseudocódigo incluye tanto el modelo clave-valor como la gestión de rasa, se han obviado declaración de variables.

[5]Bootstrap es una herramienta open-source para el desarrollo con tecnología Web: HTML, CSS y JS.

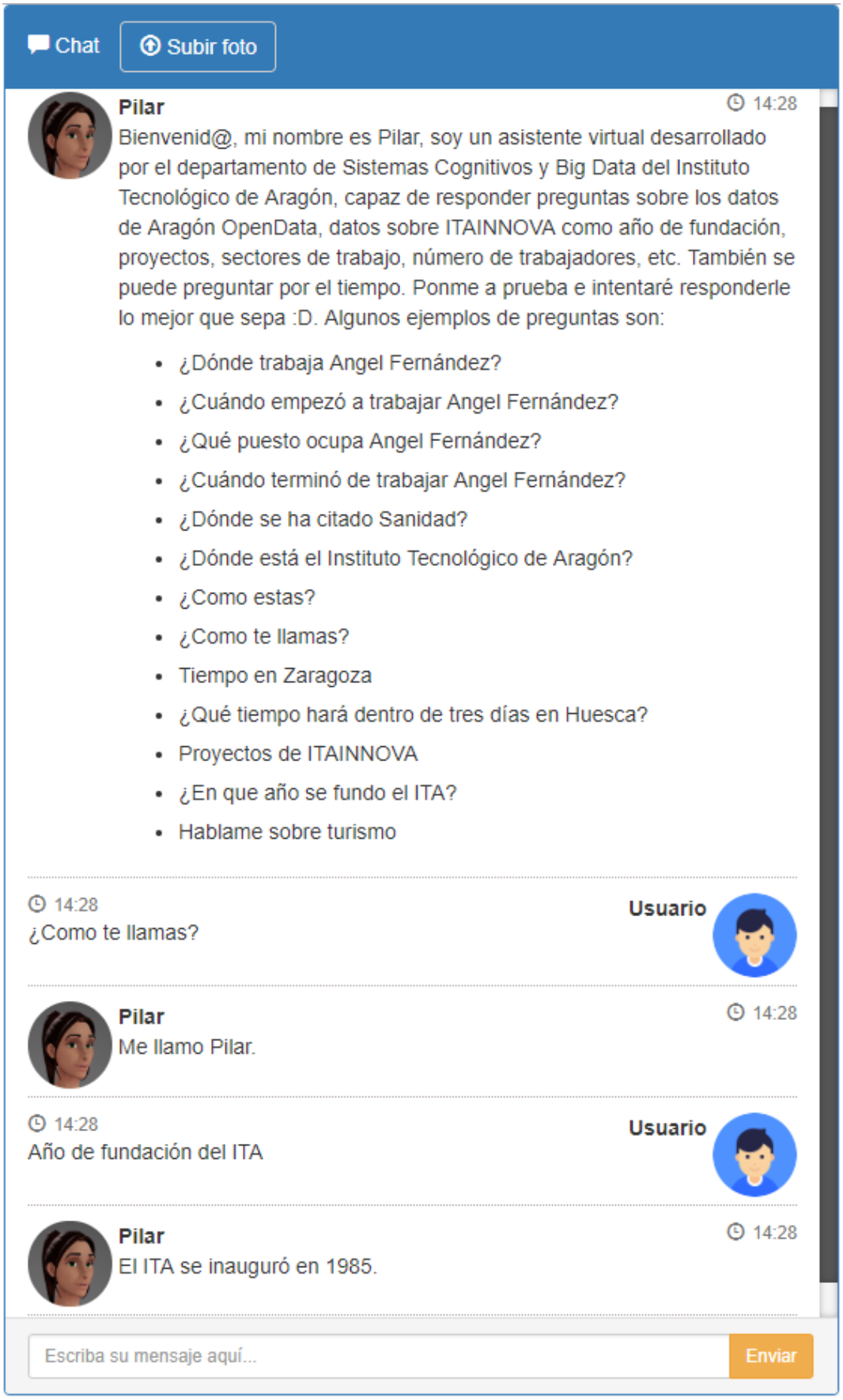

Figura 3.9: Ejemplo conversación Web 1

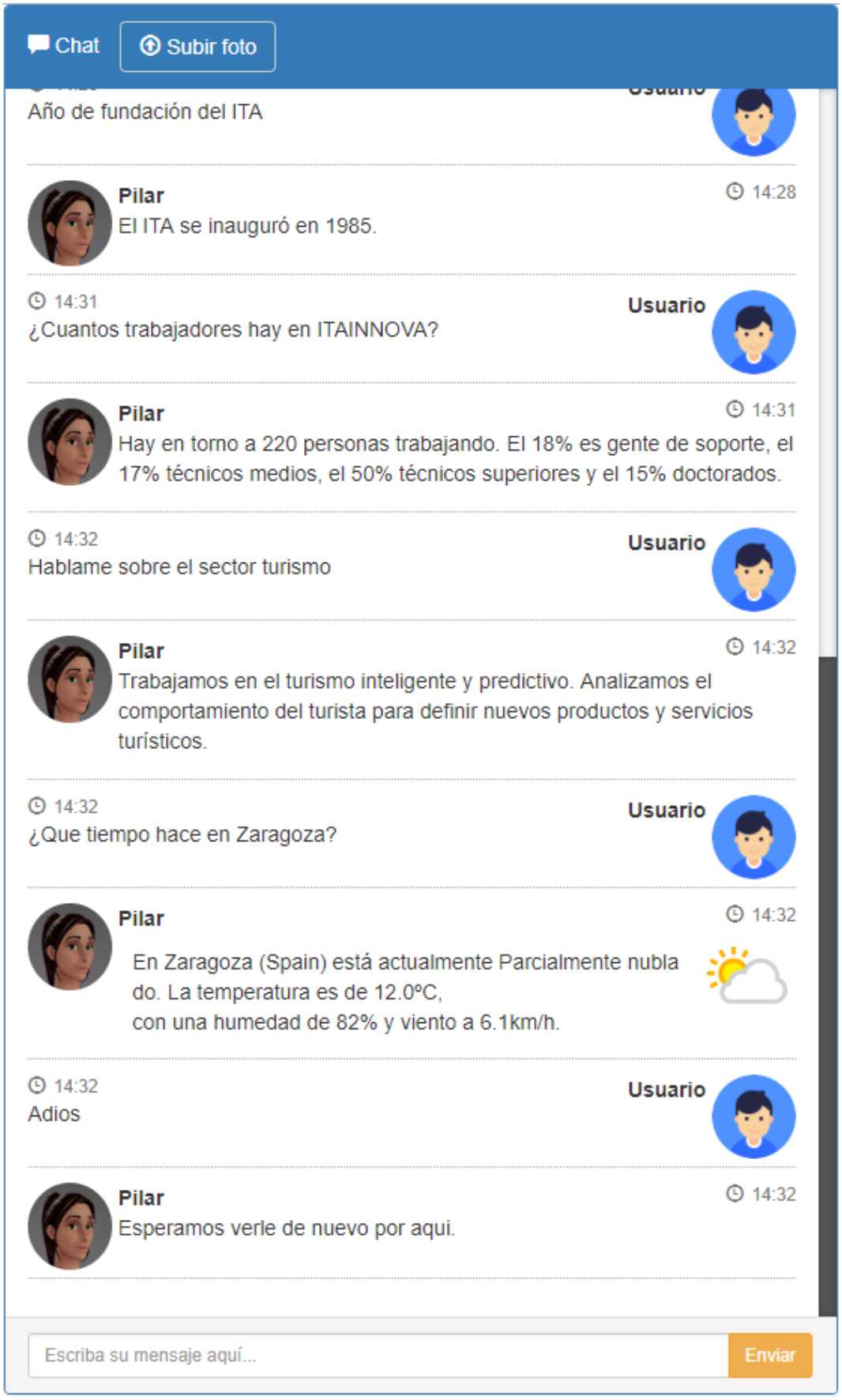

Figura 3.10: Ejemplo conversación Web 2

Capítulo 4

Validación y resultados

A lo largo de esta capítulo se van a detallar todas las pruebas realizadas para la validación tanto del modelo de red neuronal clave-valor, como de la librería Rasa para el control del diálogo. Adicionalmente se expondrán en detalle las conclusiones obtenidas. Por último, se enumerarán todas las dificultades encontradas a la largo del proyecto.

4.1. Validación modelo clave-valor

A la hora de entrenar el modelo de red neuronal es necesario explicar los parámetros que influyen en los resultados del entrenamiento: éstos son el número de épocas de ejecución, el tamaño del lote o *batch* y el número de saltos. Una época en una red neuronal es una pasada hacia adelante y otra hacía atrás de **todos** los datos de entrenamiento en la red. El tamaño de lote determina el número de ejemplos de entrenamiento en una pasada hacia delante y hacía atrás. El número de saltos determina el número de veces que durante una época se realiza una pasada completa de los datos. Se ha utilizado un 80 % de los datos para entrenamiento y un 20 % para validación. Antes de analizar los resultados obtenidos con los diferentes entrenamientos realizados, es necesario especificar la tarjeta gráfica utilizada, se trata de una Titan XP, véase Tabla 4.1.

Especificaciones del motor de GPU	
NVIDIA CUDA® Cores	3840
Frecuencia acelerada (MHz)	1582
Especificaciones de la memoria	
Frecuencia de la memoria	11.4 GB/s
Config. de memoria estándar	12 GB GDDR5X
Ancho de la interfaz de memoria	384 bits
Ancho de banda de memoria (GB/s)	547.7 GB/s

Tabla 4.1: Especificaciones tarjeta gráfica

Para los entrenamientos estos parámetros han tomado los siguientes valores:

- Épocas: desde 100 hasta 600, incremento de 100.
- Tamaño del lote: potencias de 2 desde 32 hasta 512.
- Saltos (*hops*): desde 1 hasta 6, incremento de 1.

El valor de las épocas y el tamaño del lote se aumenta para comprobar si cuanto más grande sea el valor, mejor resultado se obtiene. Los valores del tamaño del lote se han elegido basándonos en la implementación original y en los valores de uso más comunes, normalmente se utilizan potencias de 2 entre 1 y varios cientos [29]. Los resultados obtenidos en los diferentes entrenamientos se pueden observar en la Tabla 4.2

Épocas	**Batch**	**Hops**	**Train acc**	**Val acc**
100	64	3	0.57	0.42
200	64	3	0.82	0.49
300	64	3	0.92	0.42
400	64	3	0.96	0.48
500	64	3	0.99	0.59
600	64	3	0.95	0.52
500	64	1	0.89	0.41
500	64	2	0.97	0.46
500	64	4	0.98	0.42
500	64	5	0.97	0.48
500	64	6	0.96	0.40
500	32	3	0.91	0.44
500	128	3	0.97	0.42
500	256	3	0.82	0.51
500	512	3	0.79	0.42

Tabla 4.2: Resultados entrenamiento variando número de saltos

La primera prueba realizada fue el aumento de épocas para diferentes entrenamientos hasta determinar un número de épocas concreto a partir del cual los resultados comienzan a empeorar, este valor se trata de 500, para el resto de pruebas se ha establecido como parámetro fijo el número de épocas a dicho valor. La siguiente prueba consistió en aumentar y disminuir el número de saltos (*hops*), como se puede ver en la Tabla 4.2 se puede determinar que el número de saltos que mejor resultado ofrece es 3. Una vez establecidos el número de épocas y de saltos, la última prueba fue utilizar diferentes tamaños de batch, en la tabla de resultados se puede observar que aumentar o disminuir el tamaño de batch no mejora los resultados. Finalmente se ha utilizado el modelo de 500 épocas, 3 *hops* y tamaño de lote de 64 por obtener los mejores resultados. A continuación se detallan las conclusiones obtenidas de estos resultados comparándolos con los obtenidos utilizando el mismo modelo con las tareas bAbI de Facebook [16].

Antes de comparar los resultados, es necesario especificar que el corpus utilizado consta de 7384 datos con un total de 1841 palabras diferentes y que el diccionario de las tareas de bAbI ronda entre 30 y 60 palabras.

En la tabla 4.3 se puede observar la comparación de los resultados de nuestros datos (Opendata) frente al de los de la tarea 1 de bAbI, ésta consiste en un único hecho para dar apoyo a una pregunta al igual que en los datos utilizados del corpus.

Corpus	**Precisión (en %)**	
	Entrenamiento	Validación
bAbI 1	100	100
Opendata	99	59

Tabla 4.3: Tabla comparativa

Los porcentajes de entrenamiento son bastante parecidos. Con el porcentaje de validación no se ha conseguido el mismo resultado, el desfase que se aprecia se debe a la cantidad y el formato de datos que se ha utilizado. Algunas palabras se queden fuera de los datos de entrenamiento y sólo se encuentran en los datos de validación, por este motivo el porcentaje de validación disminuye. Esto no sucede con los datos de la tarea bAbI porque sólo consta de 29 palabras distintas y los datos están formados por diferentes historias, lo que supone que todas las palabras aparezcan en los conjuntos de entrenamiento y validación. En nuestro caso sólo se tiene una única historia por lo que hay palabras que no son entrenadas y por lo tanto fallan en la validación.

Para verificar su funcionamiento se han realizado una serie de pruebas con el modelo preguntándole diferentes preguntas sobre el corpus. Esta prueba consiste en evaluar el resultado devuelto por el modelo cogiendo las 5 palabras con confianza más alta devueltas por el modelo y comparando la palabra con confianza más alta con la respuesta correcta. Por ejemplo, para la pregunta: ¿Dónde trabaja Antonio Alastrué Tierra?, la respuesta adecuada es Dirección General de Trabajo; el resultado de la respuesta se puede ver en la Tabla 4.4.

Respuesta	**Confianza**
Dirección General de Trabajo	0.95241714
Servicio de Gestión de Personal y Asuntos Generales	0.03495254
Dirección General de Deporte	0.00404497
Servicio de Comercialización y Calidad Agroalimentaria	0.00266335
Dirección General de Desarrollo Estatuario	0.00125885

Tabla 4.4: Respuesta a pregunta **¿Donde trabaja Antonio Alastrué Tierra?**

Dentro de la interfaz Web, para apoyar la respuesta, se ha incluido este *ranking* de palabras y confianza. El resultado Web para la pregunta anterior se puede observar en la Figura 4.1. Sólo se apoyan las respuestas del modelo de red neuronal. Los casos que el usuario pregunta por algo que la red neuronal no sabe responder o no tiene la confianza suficiente para hacerlo, desaparece la información de apoyo.

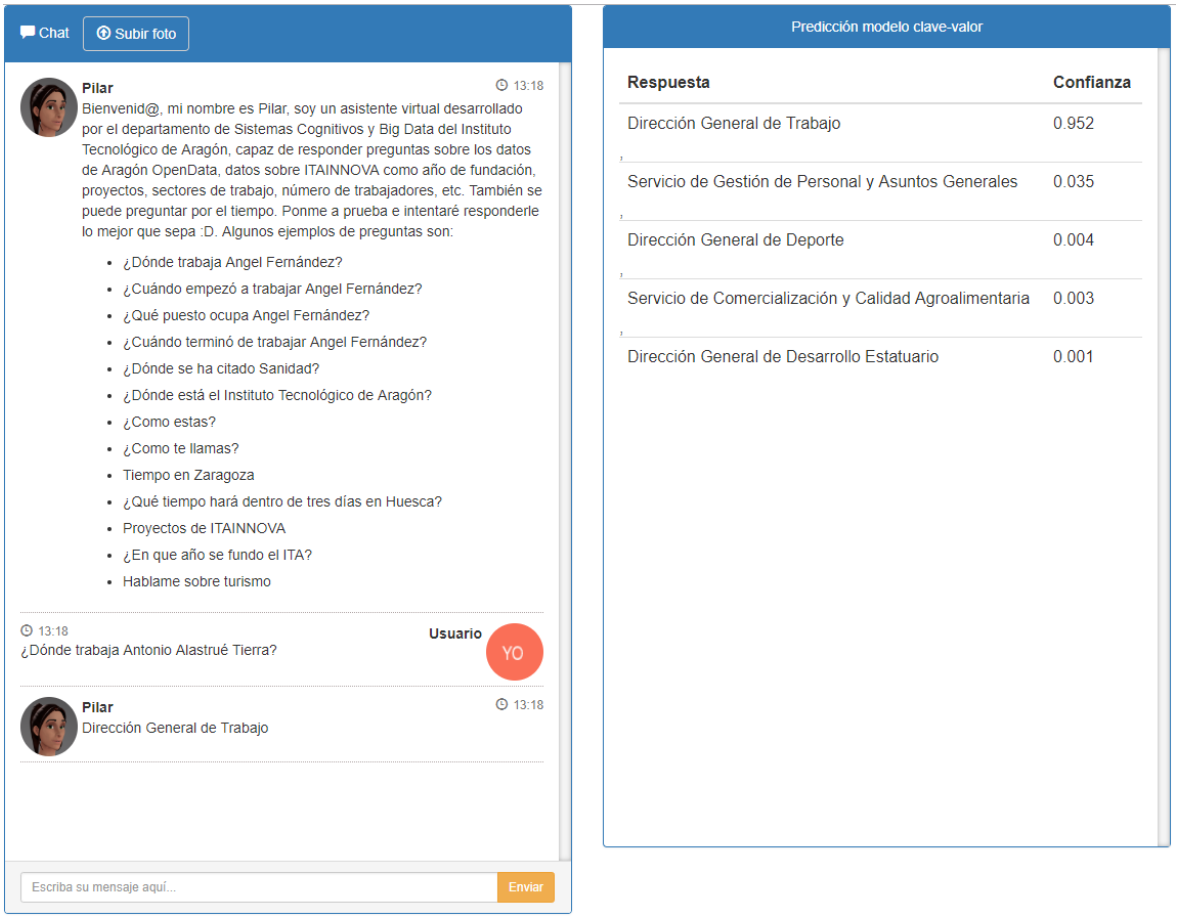

Figura 4.1: Ejemplo conversación Web con apoyo de la red neuronal

4.2. Validación control del diálogo

Para la validación del control del diálogo sólo se tiene que seguir la metodología especificada en el capítulo 3. Simplemente hay que modificar los componentes involucrados en el procesamiento del texto e ir probando qué configuración ofrece los mejores resultados.

Para este proyecto se han evaluado tres configuraciones diferentes para el procesamiento del lenguaje natural, estas configuraciones están formadas por los componentes que van a actuar sobre el texto (extractor de entidades, tokenizador, etc.). Para la evaluación de las mismas se ha probado la misma conversación con todas. La conversación tiene el formato mostrado en la Figura 4.2.

```
Chatbot: Bienvenida chatbot
Usuario: Decir nombre
Chatbot: Bienvenido nombre, pregunta por animo
Usuario: Informar estado de animo
Chatbot: Respuesta
Usuario: Preguntar tiempo actual
Chatbot: Predicción tiempo
Usuario: Despedida
Chatbot: Despedida
```

Figura 4.2: Pseudocódigo de conversación, intenciones del mensaje

Antes de comenzar la evaluación de las configuraciones elegidas se explica que características tienen cada una de ellas. Las tres configuraciones se basan en usar modelos de *spaCy* y un backend de `tensorflow` de manera individual o conjunta. Esto nos deja con tres configuraciones: una primera que utiliza exclusivamente *spaCy*, una segunda que utiliza ambos, y una tercera que utiliza sólo `tensorflow`.

SpaCy es una librería de código abierto escrita en Python y Cython[1] para el procesamiento del lenguaje natural [30]. Está diseñada para realizar tareas tales como la extracción de entidades, *POS Tagging*, dependencias entre palabras, etc.

La mayor diferencia entre *spaCy* y `tensorflow` es que la primera utiliza vectores de palabras pre entrenados, ya sea usando GloVe [31] o fastText [32], algoritmos de representación del texto. Por otro lado, el segundo no utiliza vectores pre entrenados, los crea específicamente para el conjunto de datos entrenados.

[1]Lenguaje de programación utilizado para simplificar la escritura de módulos en C y C++ y que funcionen con Python, la sintaxis es la misma que Python pero con algunas características añadidas.

Los resultados del chatbot utilizando la primera configuración (la que utiliza exclusivamente *spaCy*) son los mostrados en la Figura 4.3.

```
Le doy la bienvenida al asistente de búsqueda de Aragón OpenData. ¿Podría
decirme su nombre?
Álvaro
Bienvenido Álvaro. ¿Como estas?
Muy bien
¿Qué puedo hacer para ayudarte?
Tiempo en Zaragoza
[05/Oct/2018] INFO - Requested http://ipinfo.io/json
En Zaragoza (España) está actualmente parcialmente nublado. La temperatura es
de 19.0ºC, con una humedad de 73% y viento a 9.0km/h.
Adiós
Adiós
```

Figura 4.3: Configuración *spaCy*

Como se puede observar, esta primera configuración no sabe diferenciar entre estados de ánimo ya que la respuesta del *chatbot* **"¿Qué puedo hacer para ayudarte?"** se devuelve sólo si el usuario se encuentra mal y no al contrario.

Esta configuración tampoco sabe extraer la ciudad como la entidad que debería. Como se observa en la Figura 4.4, la entidad que se ha extraído ha sido nombre, y no ciudad. A pesar de reconocer la intención del usuario correctamente, el extractor de entidades de la configuración no es capaz de extraer la entidad adecuadamente. Por este motivo, en la Figura 4.3 aparece una frase resultante de geolocalizar la posición del usuario. Esto se realiza cuando la intención es predecir el tiempo y no se ha guardado un valor en el slot correspondiente a la localización.

```
{'entities': [{'confidence': 0.2817079930733084,
               'end': 18,
               'entity': 'name',
               'extractor': 'ner_crf',
               'start': 10,
               'value': 'Zaragoza'}],
 'intent': {'confidence': 0.9628804072410028, 'name': 'weather'},
 'intent_ranking': [{'confidence': 0.9628804072410028, 'name': 'weather'},
                    {'confidence': 0.012865603088272204, 'name': 'inform'},
                    {'confidence': 0.006779479504835268, 'name': 'change_name'},
                    {'confidence': 0.005428257697408604, 'name': 'greet'},
                    {'confidence': 0.0052055995823857195,
                     'name': 'ask_opendata'},
                    {'confidence': 0.003022197000350178,
                     'name': 'weather_period'},
                    {'confidence': 0.0015688759457208562, 'name': 'goodbye'},
                    {'confidence': 0.001332486065604286, 'name': 'happy'},
                    {'confidence': 0.0009170938744204841, 'name': 'sad'}],
 'text': 'Tiempo en Zaragoza'}
```

Figura 4.4: Resultado respuesta a: **"Tiempo en Zaragoza"**

La segunda configuración utiliza una combinación del modelo español de *spaCy* y el backend de `tensorflow`. Se abstrae del lenguaje del modelo para extraer entidades. Esta configuración funciona de tal manera que la respuesta del chatbot se basa en la mayor confianza de ambos componentes. Como se puede observar en la Figura 4.5, la respuesta al estado de ánimo se ha corregido, pero no lo ha hecho la extracción de la entidad de la localización.

```
Bienvenido al asistente de búsqueda de Aragón OpenData. ¿Podría decirme su
nombre?
Álvaro
Bienvenido Álvaro. ¿Como estas?
Muy bien
Me alegro
Tiempo en Zaragoza
[05/Oct/2018] INFO - Requested http://ipinfo.io/json
En Zaragoza (España) está actualmente parcialmente nublado. La temperatura es
de 19.0ºC, con una humedad de 73% y viento a 9.0km/h.
Adiós
Adiós
```

Figura 4.5: Configuración *spaCy* y `tensorflow`

La tercera y última configuración sólo utiliza el backend de `tensorflow`. En la figura 4.6 se puede observar como la abstrayéndose del lenguaje y creando vectores de palabras que se ajusten a los datos de entrenamiento, el sistema es capaz de responder y extraer entidades adecuadamente. En el sistema final se ha utilizado esta última configuración por ofrecer los mejores resultados.

```
Bienvenido al asistente de búsqueda de Aragón OpenData. ¿Podría decirme su
nombre?
Álvaro
Bienvenido Álvaro. ¿Como estas?
Muy bien
Me alegro
Tiempo en Zaragoza
En Zaragoza (España) está actualmente parcialmente nublado. La temperatura es
de 19.0ºC, con una humedad de 73% y viento a 9.0km/h.
Adiós
Adiós
```

Figura 4.6: Configuración `tensorflow`

4.3. Dificultades y problemas encontrados

La dificultad más representativa del proyecto ha sido la utilización de librerías tan extensas y complejas como `Tensorflow` y Rasa, las cuales requieren un aprendizaje extenso para usarlas correctamente.

En concreto, el mayor problema durante el proyecto ha sido el hecho de utilizar `Tensorflow` para guardar un modelo de red neuronal tras su entrenamiento y luego restaurarlo posteriormente para su uso. Esto es debido a que a la hora de restaurar un modelo, `Tensorflow` inicializa todas las variables declaradas previa carga del modelo junto a las del modelo. Rasa internamente utiliza y declara variables usando `Tensorflow`, las cuales no pueden ser inicializadas en la carga del modelo. Además, `Tensorflow` trabaja con sesiones de trabajo en las que se pueden usar las variables mientras ésta esté activa. El control de la sesión y restauración adecuada de las variables llevó dos semanas de trabajo completas, lo que equivale a unas 70 horas.

Una dificultad externa a la realización del proyecto ha sido el uso de la librería Rasa, que se encuentra en constante desarrollo. Semanalmente, se lanzan parches o nuevas versiones de alguno de los módulos que la componen: NLU ó Core. Esto ha supuesto la reestructuración de algunos métodos centrales del *chatbot* ya que facilitaban el uso del *chatbot* o bien porque versiones antiguas se quedaban anticuadas.

Otro problema fue la utilización de un extractor de entidades, debido a que los datos del corpus guardan toda la información referente al Gobierno de Aragón. Esto quiere decir, que algunos datos no se corresponden con la etiqueta que tienen, por ejemplo: algunos roles no tienen persona asociada en algunos períodos de trabajo, esto se guarda en la base de datos como Vacante temporal, un extractor de entidades no puede detectar un rol dentro de Vacante temporal por lo que se tuvo que realizar un análisis manual de todos los casos en los que el extractor de entidades no identificaba adecuadamente el valor de la entidad y guardar manualmente el valor.

Otra de las dificultades del proyecto ha sido el aprendizaje para utilizar todas las herramientas de las que dispone el departamento de Big Data y Sistemas Cognitivos para la gestión de proyectos. Entre estas herramientas se incluye Subversión para el control de versiones y Moriarty Studio (plataforma o conjunto de herramientas para desarrollar servicios de análisis y procesamiento de grandes conjuntos de datos de manera ágil).

Capítulo 5

Conclusiones, líneas de trabajo futuro y evaluación personal

En este capítulo final se presentan las conclusiones del proyecto explicando en primer lugar, de qué manera se han logrado los objetivos específicos del proyecto, en segundo lugar, mostrando un listado descriptivo con posibles líneas de trabajo futuras y en tercer tercer lugar, exponiendo mi experiencia y valoración personal respecto a la realización de este proyecto.

5.1. Conclusiones

Los objetivos planteados en el capítulo 1 han sido cumplidos de manera satisfactoria. A lo largo del proyecto se ha explicado todo lo realizado, entre lo que se incluye:

1. El estudio del actual estado de tecnologías sobre procesamiento y comprensión del lenguaje natural, búsqueda de respuestas y aprendizaje automático. En concreto se ha continuado con la línea de investigación de ITAINNOVA en el campo de la búsqueda de respuestas usando un modelo de *Machine Learning*.
2. La elección de un *framework* conversacional para el desarrollo del *chatbot* y el control del diálogo.
3. La integración de todo lo anterior, formando un sistema completo capaz de mantener una conversación simplemente cumpliendo con el objetivo marcado: responder a las preguntas referentes al corpus de datos abiertos del portal "Aragón Open Data".
4. La realización de todas las pruebas necesarias para validar todo el sistema, de manera independiente para cada módulo y de manera global para el sistema completo.
5. La combinación del *chatbot* con un avatar virtual propiedad del departamento. El *chatbot* dota de inteligencia al avatar. Véase Anexo VI.

5.2. Líneas de trabajo futuro

En esta sección se introducen diferentes líneas de trabajo futuro para el proyecto:

1. En este proyecto sólo se han utilizado datos concretos y muy específicos. Generar diferentes datos de entrenamiento para mejorar los porcentajes de validación del modelo de manera que sepa responder adecuadamente a más preguntas así como que el lenguaje de las preguntas sea más flexible.
2. Modificar el modelo de red neuronal para que responda con frases elaboradas y no sólo con una palabra o entidad.
3. Realizar tareas de la intranet de ITAINNOVA a través del *chatbot* como reserva de material de oficina, imputación de horas de trabajo, petición de temporadas de vacaciones.
4. Ampliar el alcance del *chatbot*, incrementando la cantidad de datos de entrenamiento para que sea capaz de comprender más casos y situaciones. Incluyendo, por ejemplo, datos de más proyectos a parte de "Aragón Open Data", más ejemplos de intenciones del usuarios, etc.
5. Continuar con el proyecto del avatar virtual incluyendo más funcionalidades y características como por ejemplo: reconocimiento de caras.

5.3. Valoración personal

A nivel personal este proyecto ha servido para entender mejor de que manera funciona un entorno profesional, las metodologías seguidas durante los proyectos y la constancia necesaria en proyectos de larga duración, así como la importancia de mantener una buena y organizada documentación a lo largo de todo el proyecto.

Además me ha servido como punto de partida para labrar un perfil profesional y competitivo para el futuro, adquiriendo conocimientos de gestión de proyectos, trabajo colaborativo, seguimiento continuo de los proyectos, uso de metodologías ágiles, aprendizaje de nuevas herramientas y tecnologías.

Glosario

Análisis de sentimientos: El análisis de sentimientos se refiere al uso del procesamiento del lenguaje natural, análisis de textos y lingüística computacional para identificar y extraer información subjetiva de unos recursos. Esto es, determinar la actitud de un interlocutor o un escritor con respecto a algún tema de un documento.

Chunking: Es el proceso que consiste en dividir un texto en partes sintácticamente correlacionadas de palabras, como son los grupos nominales, grupos verbales, etc., pero no especifica su estructura interna, ni su papel en la oración principal.

Corpus: En lingüística, se denomina corpus a un conjunto amplio y estructurado de textos (etiquetado) de acuerdo a unos ciertos criterios que, normalmente, suele ser almacenado y procesado automáticamente.

Lenguaje natural: Es la lengua o idioma hablado o escrito por humanos para propósitos generales de comunicación.

POS-Tagger (o etiquetado gramatical): Es el proceso de asignar o etiquetar a cada una de las palabras del texto su categoría gramatical (sustantivo, nombre, verbo, etc.).

Parsing (o analizador sintáctico): Es el proceso que consiste en analizar sintácticamente una frase.

Reconocimiento de entidades (NER): Es una tarea de la extracción de la información que localiza y clasifica nombres de entidades en unas categorías predefinidas.

Stopwords: Son palabras sin poder discriminatorio, que no aportan significado.

Token: En un texto, se denomina token a cada cadena de caracteres separada de otra por un delimitador. Ejemplos de tokens pueden ser, por ejemplo, las palabras.

Tokenización: En el análisis léxico, la tokenización es el proceso que consiste en separar un texto en tokens.

Bibliografía

[1] Facebook *facebook for developers*. Fecha de consulta: noviembre de 2018. https://developers.facebook.com/docs/messenger-platform/

[2] Microsoft. *Cortana*. Fecha de consulta: septiembre de 2018. https://www.microsoft.com/es-es/windows/cortana.

[3] Copyright © 2018 Apple Inc. *Apple*. Fecha de consulta: septiembre de 2018. https://www.apple.com/es/siri/

[4] Xiang Zhang & Yann LeCun. *Text Understanding from Scratch*. Fecha de consulta: julio de 2018. https://arxiv.org/pdf/1502.01710.pdf.

[5] Victor Zhong, Caiming Xiong. *State of the art deep learning model for question answering*. Fecha de consulta: junio de 2018. https://www.salesforce.com/products/einstein/ai-research/deep-learning-model-for-question-answering/

[6] Erik Zamora. *Redes Neuronales Recurrentes. Un panorama de las redes neuronales recurrentes*. Fecha de consulta: septiembre de 2018. https://es.scribd.com/doc/295974898/Redes-Neuronales-Recurrentes.

[7] Wikipedia. *ICDAR. International Conference on Document Analysis and Recognition*. Fecha de consulta: julio de 2018. https://en.wikipedia.org/wiki/International_Conference_on_Document_Analysis_and_Recognition.

[8] Google AI Blog. *Allo. Asistente inteligente*. Fecha de consulta: julio de 2018. https://ai.googleblog.com/2016/05/chat-smarter-with-allo.html.

[9] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi. *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation*. Fecha de consulta: julio de 2018. https://arxiv.org/pdf/1609.08144.pdf

[10] Cade Metz. *An infusion of AI makes Google Trnaslate more powerful than ever.* Fecha de consulta: julio de 2018. https://www.wired.com/2016/09/google-claims-ai-breakthrough-machine-translation/

[11] Amir Efrati. *Apple's Machines Can Learn Too.* Fecha de consulta: julio de 2018. https://www.theinformation.com/articles/apples-machines-can-learn-too

[12] Steve Ranger. *iPhone, AI and big data: Here's how Apple plans to protect your privacy.* Fecha de consulta: julio de 2018. https://www.zdnet.com/article/ai-big-data-and-the-iphone-heres-how-apple-plans-to-protect-your-privacy/

[13] Chris Smith. *iOS 10: Siri now works in third-party apps, comes with extra AI features.* Fecha de consulta: julio de 2018. https://bgr.com/2016/06/13/ios-10-siri-third-party-apps/

[14] Werner Vogels. *Bringing the Magic of Amazon AI and Alexa to Apps on AWS.* Fecha de consulta: julio de 2018. https://www.allthingsdistributed.com/2016/11/amazon-ai-and-alexa-for-all-aws-apps.html

[15] Alexander H. Miller et al. *Key-Value Memory Networks for Directly Reading Documents.* Fecha de consulta: abril de 2018. https://arxiv.org/pdf/1606.03126.pdf

[16] Siyuan Zhao. *Key Value Memory Networks.* Fecha de consulta: abril de 2018. https://github.com/siyuanzhao/key-value-memory-networks

[17] Wikipedia. *Calculo tensorial.* Fecha de consulta: agosto de 2018. https://es.wikipedia.org/wiki/C%C3%A1lculo_tensorial

[18] Diksha Khurana et al. *Natural Language Processing: State of The Art, Current Trends and Challenges.* Fecha de consulta: julio de 2018. https://arxiv.org/ftp/arxiv/papers/1708/1708.05148.pdf

[19] Green Jr, B. F., Wolf, A. K., Chomsky, C., & Laughery, K. (1961, May). *Baseball: an automatic question-answerer. In Papers presented at the May 9-11, 1961, western joint IREAIEE-ACM computer conference (pp. 219-224). ACM.*.

[20] Lea, , W.A. *Trends in speech recognition , Englewoods Cliffs , NJ: Prentice Hall , 1980.*

[21] Wikipedia. *AIML.* Fecha de consulta: octubre de 2018. https://es.wikipedia.org/wiki/AIML

[22] Casey Phillips. *The 3 Types of Chatbots & How to Determine the Right One for Your Needs. Chatbots Magazine.* Fecha de consulta: julio de 2018.

https://chatbotsmagazine.com/the-3-types-of-chatbots-how-to-determine-the-right-one-for-your-needs-a4df8c69ec4c

[23] Casey Phillips. *To Build Powerful AI Focus on the Data Not the Chatbot. Chatbots Magazine*. Fecha de consulta: julio de 2018. https://chatbotsmagazine.com/to-build-powerful-ai-focus-on-the-data-not-the-chatbot-8cf8db0aaa37

[24] RASA © 2018, Rasa Technologies GmbH. *Rasa Documentation*. http://www.rasa.com/docs/getting-started/overview/

[25] QuePy. *QuePy*. https://github.com/machinalis/quepy

[26] Wikipedia. *Ontología (informática)*. Fecha de consulta: octubre de 2018. https://es.wikipedia.org/wiki/Ontolog%C3%ADa_(inform%C3%A1tica)

[27] Rodrigo Pimentel. *Chatito*. https://github.com/rodrigopivi/Chatito

[28] Simolation. *Collapsible Chat Widget*. Fecha de consulta: septiembre de 2018. https://bootsnipp.com/snippets/featured/collapsible-chat-widget

[29] Yoshua Bengio. *Practical Recommendations for Gradient-Based Training of Deep Architectures*. Fecha de consulta: septiembre de 2018. https://arxiv.org/pdf/1206.5533.pdf

[30] SpaCy. *Industrial-Strength Natural Language Processing*. https://spacy.io/

[31] Jeffrey Pennington, Richard Socher & Christopher D. Manning. *GloVe: Global Vectors for Word Representation*. Fecha de consulta: octubre de 2018. https://nlp.stanford.edu/projects/glove/

[32] Copyright © 2018 Facebook Inc. *Library for efficient text classification and representation learning*. Fecha de consulta: octubre de 2018. https://fasttext.cc/

Lista de Figuras

Lista de Tablas