Universidad Internacional de La Rioja

Escuela Superior de Ingeniería y Tecnología

Grado en Ingeniería Informática

Asistente virtual para la gestión de información de empresas (TuEmpresaBot)

Trabajo fin de estudio presentado por:	Sergio Aparicio de la Mata
Línea de investigación:	Computación en la nube
Director/a:	José Alberto Benítez Andrades
Fecha:	06/09/2022

Resumen

A principios del siglo pasado se dio inicio al concepto de asistente virtual, en un principio estaban basados en sistemas expertos, debido al auge que han tenido en la última década las Redes Neuronales Artificiales, estos han pasado a implementarse con sistemas cognitivos. Los principales componentes de un asistente virtual son el NLU, TTS, ASR y DM.

Las empresas tienden a generar silos de información debido por un lado a que están departamentadas y por otro lado a la gran cantidad de datos. TuEmpresaBot es un asistente virtual para la gestión de información de empresas que soluciona ambos problemas.

Utilizando una infraestructura Cloud de pago por uso, una arquitectura flexible y unos perfiles multidisciplinares hemos construido una herramienta profesional para empresas, con capacidad de evolución constante bajo métricas de desempeño de la herramienta.

Palabras clave: (Máximo 5 palabras)

Asistente virtual, NLU, TTS, ASR y DM.

Abstract

In the last century the concept of virtual assistant began, at first they were based on expert systems, due to the boom that Artificial Neural Networks have had in the last decade, these have been implemented with cognitive systems. The main components of a virtual assistant are NLU, TTS, ASR and DM.

Companies tend to general silos of information due to the fact that they are departmentalized on the one hand and due to the large amount of data on the other. TuEmpresaBot is a virtual assistant for the management of company information that solves both problems.

Using a pay-per-use Cloud infrastructure, a flexible architecture and multidisciplinary profiles, we have built a professional tool for companies, with the capacity for constant evolution under the tool's performance metrics.

Keywords:

Virtual Assistants, NLU, TTS, ASR y DM.

Índice de contenidos

Índice de figuras

Índice de tablas

1. Introducción

En estos momentos es de todos sabido el auge de la inteligencia artificial en nuestra sociedad. Las herramientas basadas en IA están cambiando la visión de los humanos del mundo, hoy en día la conducción autónoma, identificación de patrones de imágenes o predicciones en los sectores financieros se realizan con algoritmos de IA y debido a su alto nivel de precisión han tenido una buena acogida por los usuarios.

Una de las herramientas más utilizadas son los asistentes virtuales del hogar, herramientas como Alexa u Ok Google están experimentando un crecimiento exponencial. Este incremento y acogida por la sociedad es debido en gran medida a las herramientas tecnológicas basadas en IA que utiliza.

Aunque el estudio y la aplicación de algoritmos de procesamiento del lenguaje natural llevan mucho tiempo en funcionamiento, desde la irrupción de los sistemas basados en redes neuronales artificiales han hecho que el nivel de precisión y buena percepción por parte del usuario haya llegado a niveles que se han convertido en una nueva herramienta tecnológica en la vida cotidiana del usuario.

Estos modelos cognitivos tienen la característica que cuando más datos obtengan mejor será su precisión, por lo que el auge de las arquitecturas en cloud unido con sistemas BIG DATA han ayudado en gran medida en toda esta revolución.

Durante la última década las empresas de todo el mundo están en un proceso constante de transformación digital. Es debido a que las empresas requieren de cambios tecnológicos constantes para mejorar la eficiencia de procesos, productos o servicios. Todo parece indicar que estamos en el comienzo de una etapa de transformación digital en las empresas basada en la IA.

Las empresas tienden a estar departamentadas con estructuras mayoritariamente verticales, esto hace que los datos tienda a acumularse en silos de información. La principal problemática de estos es la ausencia de transferencia de información entre silos, eso significa que podría existir un departamento A y otro B que necesitan información reciproca, pero no tienen la visión para saber de esta circunstancia.

1.1.Justificación del tema elegido

En este TFG vamos a implementar un asistente virtual para la gestión de información de empresas, que los vamos a denominar "TuEmpresaBot". Estos asistentes han experimentado un auge gracias a la irrupción de modelos basados en aprendizaje profundo. La gestión de la comunicación de empresa es un proceso costoso, donde existen escasas herramientas que ayuden a la empresa y al trabajador. Los principales problemas de estas herramientas han sido:

- Consumo excesivo de hardware.
- Los usuarios no lo ven beneficioso debido a una operatorias tediosas.
- Las puestas en marcha se alargan en el tiempo.
- Ausencia de resultados tangibles.
- Proyectos no estructurados, generalmente con perfiles acoplados.

Vamos a abordar esta problemática dotando de soluciones a los problemas descritos, para ello utilizaremos las últimas herramientas tecnológicas basadas en aprendizaje profundo. Para poder realizar este Asistente virtual vamos a realizar un estudio científico de las siguientes áreas de la Inteligencia Artificial:

- Sistemas Expertos: Es un campo de la IA que simulan el comportamiento de un experto humano en un dominio concreto, están basados en reglas.
- Sistemas Cognitivos: También llamadas Redes Neuronales Artificiales, es un campo de la IA que está basado en el aprendizaje profundo.
- Arquitecturas: Para poder construir un Asistente virtual se necesita varios componentes que deben tener una arquitectura optima, así evitamos latencias elevadas en la respuesta a los usuarios.
- Componentes o servicios: Investigaremos sobre los componentes que necesitamos para nuestra arquitectura, haremos hincapié:
    - ASR: Son sistemas de reconocimiento de habla, el nombre viene de las siglas en ingles de *Audio Speech Recognition.*

- NLU: El *Natural Language Understanding* (NLU) es un analizador de textos capaz de extraer intenciones de estos.
- DM: Denominado *Dialog Manager* su funcionalidad es determinar de la acción a realizar.
- TTS: Nos permite pasar a voz cualquier texto de un idioma determinado, son las siglas de *Text To Speech*.

Dotando a este TFG de un área novedosa tanto en el ámbito académico ya que la Inteligencia Artificial es una rama en auge y con cambios constantes, como en el ámbito empresarial debido a que vamos a solucionar un problema de comunicación que tienen innumerables organizaciones, que es el exceso de información y la incapacidad que llegue a las personas oportunas.

1.2. Problema y finalidad del trabajo

Las empresas históricamente tienen una estructura vertical, pero a la vez una acción de un departamento A, puede afectar a un departamento B o C de la misma empresa. Esto ha hecho que se fomente una cultura de información horizontal. Esto ha provocado que cada vez la información de que reciben los empleados sea mayor y por distintos canales como mail, mensajería interna, sms o WhatsApp. Esta información a menudo está sin filtrar y poco sintetizada.

Esta problemática hace que los trabajadores tengan un exceso de información que hace que les sea imposible procesar diariamente la gran cantidad de información que les llega. Por lo que muchos datos relevantes se pueden quedar sin leer, por otro lado, hay que hace hincapié que esta información que recibe el usuario es diaria y tiende a incrementarse con el tiempo, por lo que muy difícilmente un empleado puede acordarse de una información que recibió meses atrás, por ejemplo, de una nueva normativa de una empresa.

Por lo que tenemos dos problemas relacionados a resolver en este TFG: Silos de información en departamentos o áreas, filtros de información a los empleados y dotar de un sistema de información en tiempo real. En este TFG vamos a solucionar esta problemática de gestión de la información que tienen las empresas implementando un asistente virtual.

1.3. Objetivos del TFG

A continuación, pasamos a desarrollar los objetivos de este TFG.

1.3.1. Objetivos generales

Vamos a diseñar una herramienta empresarial, un Asistente Virtual con las siguientes características:

- Capacidad de interactuar mediante voz o texto.
- Operativo 7x24 con capacidad de múltiples idiomas y dar servicio a nivel mundial.
- Resolver los siguientes 3 tipos de tareas en una empresa:
    - Comunicaciones mediante voz o texto para realizar gestiones cotidianas en una empresa.
    - Conversaciones mediante texto que requieran del envió o recepción de ficheros.
    - Alertas a empleados para que sean informados de un hecho relevante.

1.3.2. Objetivos específicos

A continuación, vamos a en enumerar los objetivos para llegar a los objetivos indicados en el punto anterior.

- Modelado de las historias de usuario.
- Diseño infraestructura necesaria.
- Creación de la metodología para el proyecto.
- Elección de roles.
- Diseño de conversaciones.
- Desarrollo de las interconexiones con terceros.
- Gestión de analíticas y desempeño.
- Despliegue de la solución.
- Puesta en marcha.

2. Marco teórico

En este apartado vamos a dotar de una fundamentación teórica a este TFG. El objetivo es profundizar acerca de los asistentes virtuales. Estos están muy ligados a la Inteligencia Artificial, primero haremos evolución teórica de cómo ha evolucionado la tecnología, después investigaremos sobre las arquitecturas actuales y por último profundizaremos sobre los distintos componentes de un Asistente Virtual.

2.1. Concepto

2.1.1. Sistemas expertos

Podemos indicar que el concepto de inteligencia artificial es debido a John McCarthy de la universidad de Stanford en un congreso en 1956 (McCarthy, 1989), en esta conferencia acuño conceptos nunca visto antes como:

- La Inteligencia Artificiales la ingeniería y ciencia en crear maquinas inteligentes.
- La utilización de la computación para comprender la inteligencia humana.
- Que puede no existir una correlación entre métodos de observación biológica e inteligencia artificial.
- La relación entre el aprendizaje y un modelo en inteligencia artificial.
- Aspectos éticos y morales sobre la capacidad de una máquina de administrar el libre albedrio.

Como herramienta para materializar estos conceptos diseño un lenguaje de programación denominado LIPS (*List processing*), estaba pensado para facilitar el tratamiento de listas de datos. Esta herramienta fue pionera en la utilización de if-then-else o utilización de macros, ambas técnicas muy utilizadas en décadas posteriores hasta la actualidad en multitud de lenguajes de programación. Algunas de las características básicas de este lenguaje de programación son:

- Lenguaje interpretado con posibilidad de compilado.
- Todas las variables se pasan por referencia.
- Gestión de memoria automática.

- Optimizado para cálculos simbólicos.

Con este lenguaje podemos construir funciones que nos indique si un argumento está en una lista, devolviendo el valor T en caso de ser verdadero y NIL es caso de ser negativo. Gracias a estas funciones se pueden resolver problemas de lógica de predicados siendo pionera esta herramienta en la rama de la IA del razonamiento automático.

A partir de estos conceptos empezó a desarrollarse los denominados Sistemas Expertos (Rossini, 2000), estos lo que buscan es capturar el conocimiento de un ser humano para resolver problemas en el que necesitaría a un especialista o experto en la materia, el termino es una abreviatura de "sistema experto basado en conocimiento". Para conseguir esta funcionalidad se realiza una simulación del razonamiento que haría un ser humano experto para resolver ese problema. Por último, se desarrolló este sistema para que no sólo fuera capaz de actuar como un experto, sino que el sistema fuera capaz de razonar y resolver mejor que un ser humano en un área específica, este se acuño con el nombre dominio (Turban, 1995). Siendo el origen del concepto de asistente virtual ya que dan un valor añadido al día a día del ser humano.

Con el auge de los Sistemas Expertos empezaron a aparecer distintos lenguajes y herramientas de programación, gracias a estos se podía plasmar los fundamentos académicos en casos de uso reales con seres humano:

- Prolog: Es un lenguaje de programación lógica de propósito general asociado con la inteligencia artificial y lingüística computacional (Balbin, 1985). El nombre viene del término PROgramación LÓGica, es del tipo declarativo y está basado en reglas.
- CLIPS: Fue desarrollado por la NASA a principios de la década del ochenta, tenían la necesidad de desarrollar sistemas expertos para reducir el tiempo y el error de sus proyectos, es uno de los sistemas de referencia a la hora de desarrollar estos tipos de sistemas (CLIPS, 1994).
- Jena: Desarrollado en JAVA sus principales novedades fueron que dispone de una API ontológica, se pueden crear vocabularios formalizados de términos y gracias a estos cubrir un determinado dominio (Rajagopal, 2005).

- OpenCyc: Es una versión software libre del motor CyC, el pilar en que el que se basa este framework es en una gran base de datos de conocimiento con relaciones y conceptos que realiza el ser humano de forma cotidiana, con lo que se consigue dotar de sentido común a nuestro sistema expertos a la hora de realizar razonamientos. (POHL, 2012)

Las ventajas de estos Sistemas Expertos serían las siguientes:

- No tienen limitaciones o situaciones propias al ser humano (Envejecer, enfermedades o rotación laboral).
- Son totalmente replicables para otras áreas del conocimiento.
- La respuesta es homogénea con un razonamiento lógico justificado mediante trazas.
- La velocidad de respuesta es muy superior a la del ser humano.

Por el contrario, tienen las siguientes desventajas:

- Requiere de programación, si existe un nuevo caso en el dominio hay que codificarlo.
- No son capaces de aprender de sus errores, carecen de sentido común.
- Incapacidad de discernir entre lo relevante y secundario en la resolución de un problema.
- Imposibilidad de poder tener una conversación en lenguaje natural con el Sistema Experto.

Podemos indicar que un sistema experto está muy acotado a un dominio especifico de aplicación, es una tecnología madura y de gran efectividad siendo la precursora de los denominados asistentes virtuales (Badaró, S., Ibañez, L. J., & Agüero, M. J. 2013).

2.1.2. Sistemas cognitivos

Como hemos visto en el punto anterior los Sistemas Expertos tienen una limitación cuando existen datos incompletos en el conocimiento empleado, esta incertidumbre es muy común en las relaciones en los seres humanos como por ejemplo cuando nos comunicamos mediante natural.

Las Redes Neuronales Artificiales (RNAs) simulan el comportamiento del ser humano a partir de un conjunto de datos. Este aprendizaje profundo o *Deep learning* es una parte de la

Inteligencia Artificial con un gran auge en la última década. Como hemos visto en el punto anterior los Sistemas Expertos están basados en reglas, esto hacía que resultaran relativamente sencillos para una máquina, sólo hay que codificarlo, pero por el contrario muy tediosas para comprenderlo por un ser humano, al requerir de una base de razonamiento, algorítmica y programación. Por lo que las tareas abstractas y formales suelen ser fáciles de resolver para una máquina, mientras que las tareas más sencillas, que resultan intuitivas y directas para una persona (como reconocer una cara) son a su vez las más difíciles de ejecutar para un ordenador. (Hochreiter, S. y Schmidhuber, J. 1997).

Las RNAs vienen a solucionar la problemática donde los Sistemas Expertos no son capaces de resolver, ya que obtienen su conocimiento gracias a las extracciones de patrones a partir de un conjunto de datos de entrada. Estos datos son fundamentales para la resolución mediante *Machine Learning,* al igual que ocurre en el mundo real, ya son los que les van a dar una percepción de la realidad y a partir de estos tomaran una decisión. Básicamente los problemas que se resuelven mediante Redes Neuronales Artificiales necesitan una serie de características en forma de datos, que se denomina *features*. La estrategia que sigue el aprendizaje profundo es conseguir que las maquinas construyan de forma automática conceptos complejos a partir de conceptos sencillos. (Hochreiter, S. et al. 1997).

2.1.2.1. Evolución de los sistemas cognitivos:

El concepto de aprendizaje profundo ha tenido varios nombres a lo largo de la historia, siendo el más utilizado Redes Neuronales Artificiales. El motivo de este nombre es la inspiración biológica que existió desde el primer momento, ya que lo que se intentaba era replicar los conceptos de inteligencia y aprendizaje e intentar algoritmos para conseguir un sistema inteligente. El principal problema fue que el conocimiento del cerebro humano no es suficientemente avanzado por lo que ha sido imposible esta replicación quedando únicamente la inspiración.

Los primeros modelos están basados en una neurona aislada, los pioneros en proponer un modelo son McCullochy Pitts en 1943. Se inspiraban en un modelo biológico donde las entradas se llamaban dendritas y sólo había un único canal de salida denominado axón. Gracias a este modelo se podía simular todas las puertas lógicas el algebra booleana. En 1957

Frank Rosenblatt diseño el concepto de Perceptrón. Este modelo ya era una Red Neuronal Artificial con capacidad reconocer patrones sin programarlos previamente a través de reglas, era capaz de generalizar a partir de unos datos de entrada. (Zhang, L., & Zhang, B. 1999).

Bernard Widroll y Marcian Hoff realizaron la primera Red Neuronal Artificial Aplicada a un problema cotidiano, conseguían realizar filtros adaptativos para eliminar ecos de la línea del teléfono. A este modelo se le denomina ADALINE (Adaptive Linear Element), en 1960 consiguieron diseñar un sistema de entrenamiento similar al *stochastic gradient descent*. En 1965 Robinson propone un algoritmo para capaz de aplicar criterios de razonamiento lógico. Al mismo tiempo, aparece en el MIT el primer chatbot denominado Eliza. En el período comprendido entre 1966 y 1974 se profundiza en el concepto de la complejidad computacional, algo esencial para las futuras evoluciones de la disciplina.

En la década de los 70 se desarrollan los primeros sistemas con base de datos de conocimiento. Uno de los primeros trabajos fue con coches autónomos, con una estructura muy primitiva pero capaces de dar valor añadido a un coche con control remoto. La década de los 80 es la que más prolifero los sistemas expertos. En la década de los 90 hubo un resurgir de las redes neuronales, este empezó unos años antes concretamente en 1988 nuevas soluciones como algoritmos genéticos o *soft computing* contribuyen al enriquecimiento de la disciplina. En 1997 el ordenador de IBM, *Deep Blue*, derrota a Gary Kasparov en uno de los enfrentamientos que más expectación han generado. En el año 2011, unos de los programas más famosos de estados unidos es *Jeopardy*, IBM como estrategia comercial decide poner su sistema de IA, denominado Watson, a competir y consigue ganar el concurso. En el año 2015, la red neuronal desarrollada y entrenada por Google, *DeepMind AlphaGo,* gana al campeón mundial de Go. (Acevedo, E., Serna, A., & Serna, E. 2017).

Vadym (2017) nos indica que el gran cambio a nivel de hardware es la utilización de procesamiento gráfico en entrenamiento de los modelos de las redes neuronales, las GPU son básicamente calculadoras de coma flotante cuya arquitectura paralela de núcleo permite realizar operaciones simultáneamente. Vamos a ver 2 ventajas que hacen que tengan más velocidad las GPUs con respecto a las CPUs en el cálculo de operaciones de vectores y matrices, cruciales en el procesamiento de RNA:

- La GPU disponen de muchas más ALU que una CPU.
- La GPU tiene de unidades de control / caches independientes para cada zona de ALUs.

En la siguiente figura podemos ver la arquitectura de una CPU y una GPU, se puede apreciar estas 2 ventajas competitivas, en color verde están las ALU, en color azul las unidades de control y por último en color naranja las caches.

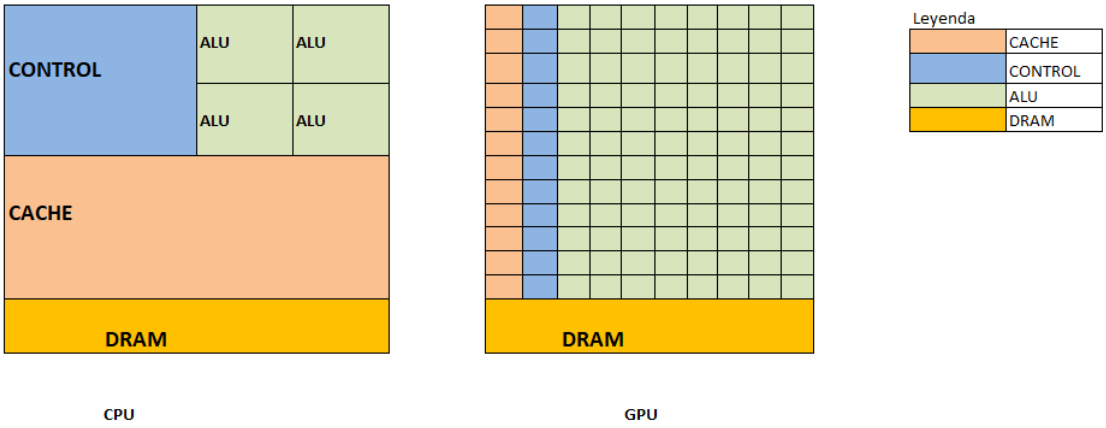

Figura 1. Arquitectura de una CPU vs GPU Fuente: Realización propia

No todo han sido éxitos en la historia de la inteligencia artificial, los períodos que abarcan los años 1974-1980 y 1987-1993, son conocidos como los inviernos de la inteligencia artificial. Estas franjas de tiempo se caracterizan por una gran desilusión en las posibilidades de esta disciplina y descenso del número de aportaciones. (Arrestegui, L. B. 2012).

2.1.2.2. Estructura:

En estos sistemas la unidad principal es la neurona, básicamente es un nodo dentro de una de red neuronal. En la siguiente figura podemos ver su estructura.

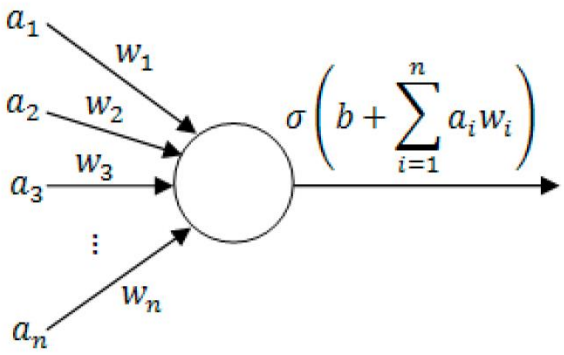

Figura 2. Expresión matemática de una neurona en una red neuronal.
(https:/towardsdatascience.com/a-gentle-introduction-to-neural-networks-series-part-1-2b90b87795bc)

A continuación, pasamos a describir los siguientes componentes:

- Donde las entradas son los "a": Es el conjunto de datos de entrada también denominados features.
- Los pesos o weights serían las "w": Es un método de ponderación entre los datos de entrada y salida final.
- Un bias (b): Representa un valor constante, permiten a una neurona modelar un valor constante e independiente del conjunto de entrada e influye la salida de la neurona.
- Función de activación que se aplica sobre la suma del bias con el productor de cada input por su peso correspondiente.

Las funciones de activación son muy diversas en la figura podemos ver una de tipo Sigmoid.

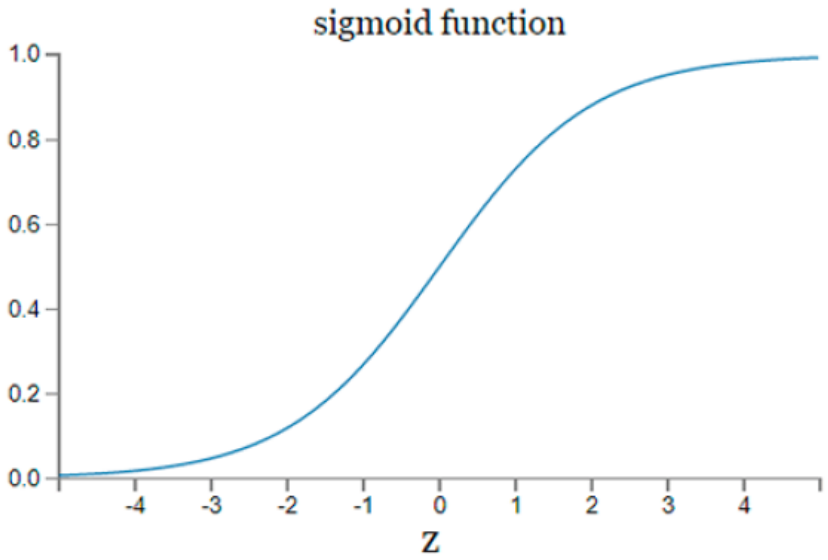

Figura 3. Función sigmoid (http://neuralnetworksanddeeplearning.com/chap1.html)

Esta toma valores entre 0 y 1: Tiende a infinito en valores cercanos a 1 y a 0 para valores cercanos a menos infinito. Se suele utilizar para modelar probabilidades.

Como hemos visto una neurona representa una función lineal, la interconexión entre estas representa una Red Neuronal Artificial, con esto conseguimos resolver problemas más complejos y permitir las interacciones con distintos tipos de entradas mediante la conexión entre neuronas. Por lo que podemos decir que una red neuronal, también llamada *multilayer perceptron* (MLP), es un conjunto de neuronas simples situadas en capas (Goodfellow, I., Bengio, Y. y Courville, A. 2016).

En la siguiente figura podemos apreciar la estructura de una Red Neuronal Artificial, en este caso todos los nodos o neuronas van a estar conectados y se le denominan: *Fully connected networks.*

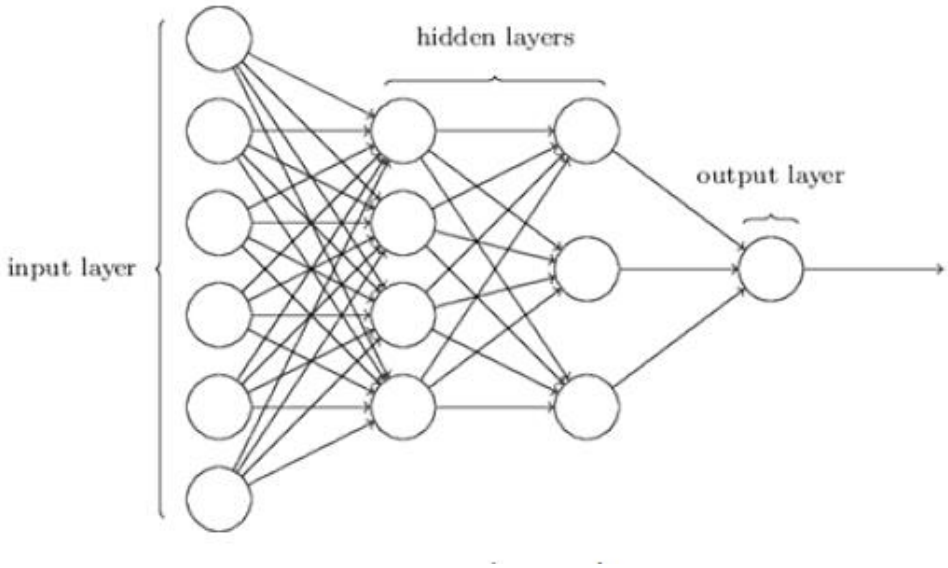

Figura 4. Red neuronal, Fully connected networks
(http://neuralnetworksanddeeplearning.com/chap1.html)

En esta figura podemos ver las distintas capas que puede tener una Red Neuronal Artificial:

- El *input layer* es la capa que gestiona los datos de entrada.
- La última capa representa los datos de salida, se denomina *output layer*.
- Las capas intermedias se denominan *hidden layers* estas campas son la clave que permite que nuestra Red Neuronal Artificial aprenda tareas complejas.

Como hemos visto en una neurona artificial tenemos unos pesos denominados (w) y bias (b) cuando en decimos que estamos entrenando una Red Neuronal Artificial, lo que se está realizando es encontrar los pesos y bias idóneos que describir mejor los datos de salida (Salas, R. 2004).

Uno de los conceptos más cruciales en Redes Neuronales Artificiales es el *Backpropagation*, es la parte más importante ya que recae el entrenamiento de la RNA. Previamente se tiene que conocer las derivadas de cada neurona, la función de coste y lo que se hace es calcular el gradiente de coste final y se propaga hacia atrás hasta obtener los gradientes respecto a todos los parámetros. Este proceso es general y funcionará para cualquier tipo de arquitectura de red siempre y cuando esté constituida de operaciones diferenciables.

A continuación, vamos a poner detallar un caso concreto de un *input* individual x de la red:

- *Forward pass:* Aplicamos el valor de x a la red neuronal y guardamos todos los valores intermedios de salida de cada operación o neurona. Obtenemos el valor final de la función de pérdida para x.
- *Backward pass:* Obtenemos el gradiente de la función de pérdida respecto a sus *inputs* y empezamos a propagar el gradiente hacia atrás mediante el uso de la regla de la cadena, calculando los gradientes respecto a cada parámetro.
- Una vez obtenidos todos los gradientes, tenemos los valores necesarios para aplicar *stochastic gradient descent.*

El proceso de *stochastic gradient descent* se hace sobre *mini-batches*, de modo que realizamos el proceso de *backpropagation* para cada ejemplo en la *batch*. En la práctica, los cálculos de *backpropagation* se hacen a la vez para toda la *batch*, utilizando cálculo de matrices. Esto permite entrenar redes neuronales más rápido y aprovechar la eficiencia de operaciones vectoriales. (Lillicrap, T. P., Santoro, A., Marris, L., Akerman, C. J., & Hinton, G. 2020).

2.1.3. Sistemas expertos vs sistemas cognitivos

En este TFG hemos podido analizar en el punto 2.1.1. que los sistemas expertos disponen de una base de conocimiento desacoplada del procesado del conocimiento. Gracias a este sistema por capas cualquier conocimiento adicional se puede añadir a la base de datos sin tener que adicionar nuevo código. Por otro lado, en caso de necesidad de la adicción de una nueva regla tendríamos que codificarlo en la capa de procesamiento. (López, R. F., & Fernández, J. M. F. 2008).

Por otro lado, también hemos podido profundizar en el punto 2.1.2 del este TFG como en una RNA no hay que insertar el conocimiento en una base de datos o programar nuevas reglas, ya que la RNA gestiona el aprendizaje de las reglas mediante las conexiones entre las distintas capas y neuronas de la red (López et al. 2008).

Debido a los puntos anteriores, podemos sacar la conclusión que una de las principales diferencias es que los sistemas expertos dependen de reglas mientras que los sistemas cognitivos no necesitan programar nuevas reglas ya que su aprendizaje no depende la codificación sino de la adicción del conjunto de datos, ya que el aprendizaje se consigue gracias al cambio de los pesos de las conexiones de las redes a partir de los datos de entrada

facilitados. Este sistema es muy beneficioso que nos permite decir que las redes neuronales aprenden a partir de la experiencia (López et al. 2008).

Otra diferencia es que los Sistemas Expertos necesitan que los datos estén estructurados como por ejemplo ontológicas, mientras que una Red Neuronal Artificial al estar el conocimiento distribuido en los nodos y conexiones de las distintas capas de la red es muy tolerable a ruido y datos desestructurados como el que se realiza en el lenguaje natural. Ya que, aunque la gramática tiene una estructura fija y es la misma para todos los usuarios su aplicación es casi infinita.

Este sistema facilita que cuantos más datos de entrada tenga la RNA mejor va a ser su desempeño, los Sistemas Expertos no disponen de esta capacidad por lo que una vez que hayamos alimentado la base de conocimiento, no conseguiremos aumentar el desempeño por incrementar datos del dominio del Sistema Experto. El único inconveniente de las RNAs a nivel de datos de entrada es el denominado *Overfitting* o sobreentrenamiento.

Las RNAs se entrenan mediante un conjunto de datos, lo que hacemos con estos es hacer encajar (*fit* en inglés) los datos de entrada con los de salida. En este entramiento se pretende que nuestro sistema cognitivo sea capaz de generalizar, conseguir una función universal para que cualquier dato de entrada sea capaz de generar un dato de salida preciso. Una vez que tenemos a nuestro modelo entrenado insertaremos un nuevo conjunto de datos desconocido, este debe de ser capaz de dar la salida correcta (Galvañ Sala, D. A. 2021). Vamos a poner un ejemplo, supongamos que tenemos una RNA capaz de detectar si una persona lleva mascarilla o no, vamos a ver los posibles conjuntos de entrenamiento y su consecuencia que pueden tener en el desempeño de la salida:

- Si las fotografías que conforman los datos de entrenamiento son muy pocas el sistema no será capaz de generalizar y el desempeño será pobre, a este efecto se le denomina *underfitting.*
- Si las fotografías que conforman los datos entrenamiento son de un número considerable pero sólo de personas que llevan puestas mascarillas de tela, nuestro sistema no sabrá detectar cuando un ser humano lleve una mascarilla FP2, a esta causa se le llama *overfitting*.

Los conjuntos de datos en las Redes Neuronales Artificiales deben tener un número considerable, por un lado, para no entrar en *underfitting* mientras por otro tienen que ser los más heterogéneos posibles y con poco sesgo para que no caer en overfitting. Los Sistemas Expertos debido a su estructura que está basada en reglas no tiene esta problemática.

En la siguiente tabla podemos ver un resumen de las diferencias entre un Sistema Experto y un Sistema Cognitivo.

Tabla 1. Comparativa entre un Sistema Experto y Cognitivo

Sistemas Expertos	**Sistemas Cognitivos**
Llegado a un nivel de datos no mejora el desempeño del problema.	Cuando más datos de entrada mejor es el desempeño.
Basado en reglas.	Basado en redes neuronales artificiales.
No tiene problemas de sobreentrenamiento.	Puede tener problemas de *underfitting y overfitting.*
No tienen limitaciones o situaciones propias al ser humano	No tienen limitaciones o situaciones propias al ser humano.
Dispone de traza para comprender el resultado.	No dispone de trazas para comprender el resultado.
No se puede establecer una conversación en lenguaje natural con ellos.	Se puede establecer una conversación en lenguaje natural con ellos.
No puede discernir entre lo relevante y secundario.	Puede discernir entre lo relevante y secundario, por ejemplo, en una imagen.

2.2. Asistentes virtuales

El crecimiento de los asistentes virtuales en la última década es exponencial, como hemos visto en los puntos anteriores esta debido en gran medida al auge de las RNAs. Unas de las ventajas competitivas de un asistente virtual es que el usuario se comunica mediante lenguaje natural, por lo que la detección y generación de voz es una de las tareas más importantes a nivel de procesamiento en tiempo real. (González Ávila, N., López Martínez, I., & Hernández García, N. 2020). Hay que destacar que en algunos textos hemos podido ver los siguientes sinónimos de asistentes virtuales: Agente virtual, Asistente inteligente, Agente conversacional.

Siempre funcionan de la misma manera, un usuario mediante voz o texto utilizando el lenguaje natural realiza una pregunta o cuestión, el asistente virtual la procesa y da una respuesta eficiente en lenguaje natural mediante voz o texto. Por lo que es un requerimiento de estos que esta comunicación se realice de la una forma comprensible, tanto para el usuario como para el asistente. (Mondejar, 2007).

Un asistente virtual va a requerir de un componente de reconocedor de voz por un lado y por otro de generador de voz, ambos tienen que funcionar en tiempo real y mediante lenguaje natural ya que tienen que comunicarse con un ser humano como si fueran un humano. Esto supone de grandes desafíos que los vamos a desarrollar a continuación.

Existen palabras cuyo significado dependen de las palabras anteriores o de las posteriores, un claro ejemplo serían las palabras “vaya”,” baya” y “valla”. Las 3 serían homófonas al sonar igual y se escriben de forma distinta, la primera seria es verbo “ir” la segunda es una fruta y la última un obstáculo, como podemos apreciar sabemos su significado cuando hablamos gracias a la palabras anteriores o posteriores. Por otro lado, existen idiomas muy utilizados en el mundo como el inglés o español, esto provoca que una palabra pueda variar el significado según el país y la región donde te encuentras, como por ejemplo la frase: “¿Nos tomamos un tinto?”, si la escucha un español entenderá que hace referencia a un vino tinto, mientras que si la escucha un colombiano pensara en un café. Estos casos son innumerables en la comunicación mediante los seres humanos:

- Neologismos: Siempre aparecen nuevas palabras en los idiomas, *clickear*, *influencer* o castear serian ejemplo en el idioma español.
- Ironías: Frases como ¡Menudo fenómeno!, cuando realmente no ha funcionado la instrucción del asistente virtual.
- Poesía: "¿Qué es poesía? ¿Y tú me lo preguntas? Poesía... eres tú". ¿Un ser humano podría ser una poesía ?, en cualquier comunicación todos los miembros tendrían claro que hace referencia a una poesía, pero una maquina debe de saber entender esta comparación.

Son licencias o figuras que se utilizan en cualquier idioma que un Asistente Virtual debe de saber interpretar en tiempo real. Por último, otro gran desafío es que es la utilización de las gramáticas por parte de los humanos, ya que estas disponen de una estructura fija mientras que su aplicación es casi infinita. (Marchena, O. G. 2009).

La principal funcionalidad de un Asistente Virtual es recibir órdenes en lenguaje natural, estas pueden ser mediante voz o texto. El NLU (Natural Language Understanding) permite la interacción persona-computadora. Es la comprensión del lenguaje humano como el inglés, el español y el francés, por ejemplo, lo que permite a los móviles u ordenadores comprender comandos sin sintaxis de lenguajes de programación. La respuesta del Asistente Virtual tiene que ser lo más rápida posible, eso conlleva una gestión del retardo óptima. Hay que tener en cuenta que tenemos varios componentes o capas que nos pueden provocar retardos como el procesamiento del lenguaje, la ubicación y la conectividad del usuario. Una de las partes con mayor peso computacional es la comprensión del lenguaje natural, ya que analiza los datos para transformar el habla en una ontología estructurada, a partir de esta conseguir reconocer los siguientes elementos:

- Intenciones: Se le denomina al proceso de identificar el sentimiento del usuario, este proceso se realiza a partir de un texto y lo que se pretende es poder identificar el objeto por el cual el usuario se ha puesto en contacto con el asistente.
- Entidades: En este proceso lo que se pretende es extraer información relevante de un texto. Estas pueden ser numéricas o caracteres, las primeras son números, monedas o

porcentajes, mientras que las segundas se pueden agrupan en categorías como personas, números o porcentajes.

Gelbukh (2010) nos indica que los asistentes virtuales tienen cuatro desafíos técnicos a resolver que a continuación vamos a enumerar:

- Procesamiento del habla: Capacidad de pasar la voz a texto para que las maquinas sean capaz de entenderlo.
- Conducción del dialogo: Capacidad de entender un dialogo para extraer el objetivo de la conversión.
- Generación del lenguaje: Capacidad que la maquina responda en el mismo lenguaje al humano para que este sea capaz de comprenderlo.
- Relacionar palabras con las acciones: Capacidad de realizar una acción por parte a partir de la intención que tiene el usuario.

El crecimiento exponencial de internet ha sido otro de los factores clave en la proliferación de asistentes virtuales. Al mismo tiempo cada vez hay más dispositivos conectados a internet, esto hace que haya una demanda de los usuarios para que las maquinas se comuniquen con estos mediante lenguaje natural. Los diálogos se centran en tareas específicas, como por ejemplo vender un producto en una tienda de electrodomésticos de acuerdo con unos requerimientos del cliente. En este ejemplo la complejidad reside en el número de productos y en las aplicaciones existentes para que el asistente pueda conseguir la información. (Aguado, L. R., Serrano, A. M. G., & Fernández, P. M. 2002)

Aguado et al. (2002) realizaron una de las primeras arquitecturas para la implementación de un asistente virtual, que podemos apreciar en la siguiente figura.

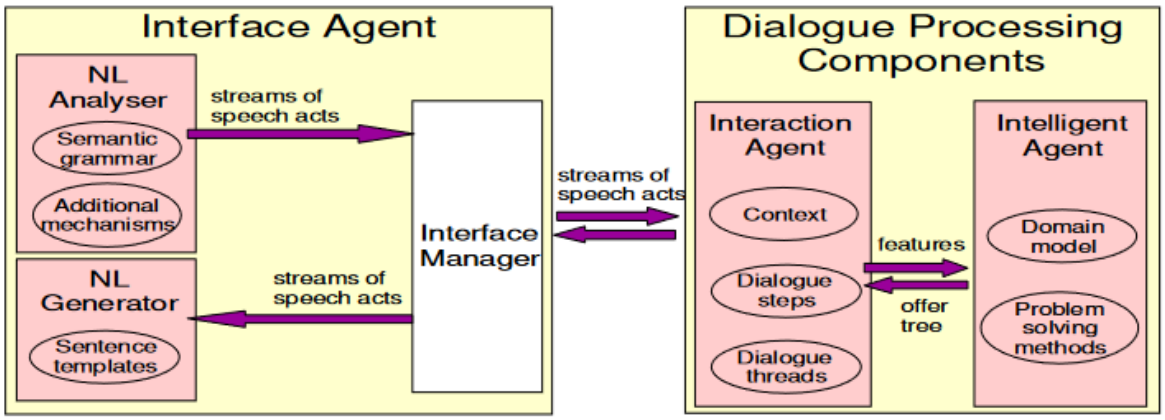

Figura 5. Arquitectura general de un asistente virtual

(http://journal.sepln.org/sepln/ojs/ojs/index.php/pln/article/download/3306/1794)

2.3. Audio Speech Recognition

El reconocimiento de voz denominado ASR (*Automatic Speech Recognition*) es el área que transforma la voz a texto, uno de los principales retos es que cada persona pronuncia una palabra de forma distinta, por lo que la variabilidad en la entrada del sistema es muy alta. En un principio se diseñaban ASR mediante sistemas expertos, el fundamento de estos es comprobar cómo se pronuncia una palabra e ir recorriendo todo el banco de sonidos que tenía el sistema. La herramienta clásica en esta área es el algoritmo de Viterbi (Catalán Ludwig, I. 2011).

El reconocimiento del lenguaje natural por un lado es en tiempo real y por otro lado es en secuencia, por lo que requiere una flexibilidad en los datos de entrada, a priori no sabemos cuánto va a durar el fragmento de voz a reconocer. Las RNA que hemos visto en puntos anteriores tienen un tamaño fijo de entrada y de la salida, con la irrupción del aprendizaje profundo se diseñó un tipo de red neuronal capaz de solucionar este problema, las Redes Neuronales Recurrentes. Estas proporcionan una flexibilidad con arquitecturas de longitud variable tanto en inputs como en outputs, tienen la capacidad de trabajar con información secuencial, muy valioso en el reconocimiento del lenguaje donde el contexto y orden son determinantes. Mantienen un estado interno con memoria de lo visto hasta el momento, aplicando una formula recurrente sobre la secuencia de entrada de manera que, en cada paso

de la secuencia, se depende del valor input “x” de ese momento y del *hidden state* anterior “h”. Cada paso en la secuencia se conoce como *time step*.

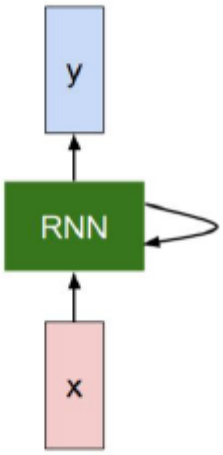

Figura 6. Relación datos entrada y salida en una red neuronal recurrente. (Stanford CS231N)

Las RNNs, siglas en ingles *Recurrent Neural Networks* son eficaces en modelar relaciones temporales entre los elementos de la secuencia a través del estado interno de la red y también son muy versátiles al tener un ancho variable en los datos de entrada y salida. Los modelos del lenguaje son una aplicación de las redes recurrentes donde estás han conseguido muy buenos resultados.

Un modelo del lenguaje es un modelo que asigna una probabilidad a una secuencia de palabras (o de caracteres). Un modelo del lenguaje puede tratarse de manera natural con RNN. Podemos suministrar una secuencia de palabras a nuestra red recurrente, de manera que la red nos dé una distribución de probabilidad de la palabra siguiente. El *hidden state* de la RNN codifica en cierta medida el texto visto hasta ahora, por lo que la red es capaz de condicionar la probabilidad de la siguiente palabra a lo visto anteriormente. El *input* de cada *time step* puede ser un *word vector* o una representación *one-hot*. La salida en cada time step viene dada por una capa *softmax* con las probabilidades de que cada palabra del vocabulario sea la siguiente en la secuencia.

En este tipo de arquitecturas hay dos problemas muy típicos:

- *Vanishing gradients*: los gradientes “se desvanecen” y van a 0.

- *Exploding gradients*: los gradientes “explotan” y se hacen muy grandes.

Estos 2 problemas vienen por el hecho de tener que hacer backpropagation sobre una secuencia larga de elementos, hacen que el entrenamiento se vuelva inestable o la red no entrene correctamente.

Al tener que hacer backpropagation sobre una secuencia larga de elementos, hacen que el entrenamiento se vuelva inestable o la red no entrene directamente, intuitivamente el problema es similar a lo que pasa cuando multiplicamos por sí mismo en muchas ocasiones:

- Si es mayor que 1, obtenemos valores cada vez más grandes estaríamos hablando del *exploding gradients*.
- Si esta entre 0 y 1, obtenemos valores más cercanos 0 seria *vanishing gradient*.

Para solucionar estos dos problemas han surgido arquitecturas mejoradas como LSTM o GRU.

LSTM data 1997 cuyas siglas viene de *Long Short-Term Memory*, En la práctica, las redes recurrentes simples presentan problemas para aprender relaciones entre elementos en *time steps* muy separados, debido a factores como los *vanishing* y exploding gradients. Las LSTM se diseñaron para solucionar este problema de “memoria” de las RNN originales o Vanilla, mantienen un estado interno adicional, el denominado *cell state*, aparte del tradicional hidden state. Las LSTM no sufren de *vanishing* o *exploding gradients*, la solución a estos problemas viene del estado interno ct, que crea una especie de “autopista de la información” que ayuda durante backpropagation. Este nuevo estado lo podemos apreciar en la siguiente figura.

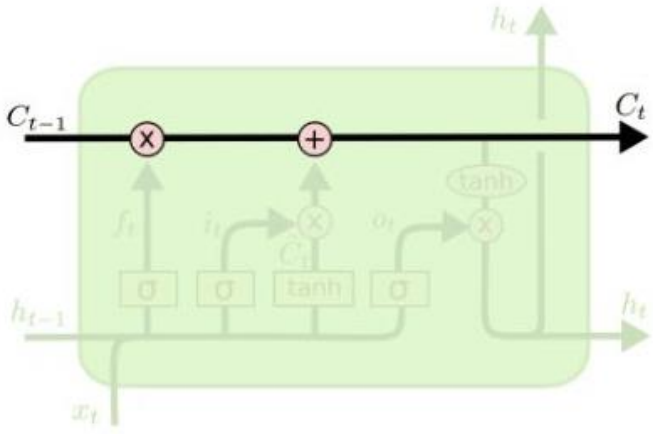

Figura 7. Estado interno ct de una RNA LTSM. (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

La arquitectura GRU fue introducida en 2014, para solucionar los problemas de vanishing o exploding gradients lo que hacen es combinar el cell state y el hidden state en un solo elemento. (Dey, R., & Salem, F. M. 2017). En la siguiente figura podemos ver que en las GRU no existe C y se hace esta labor h.

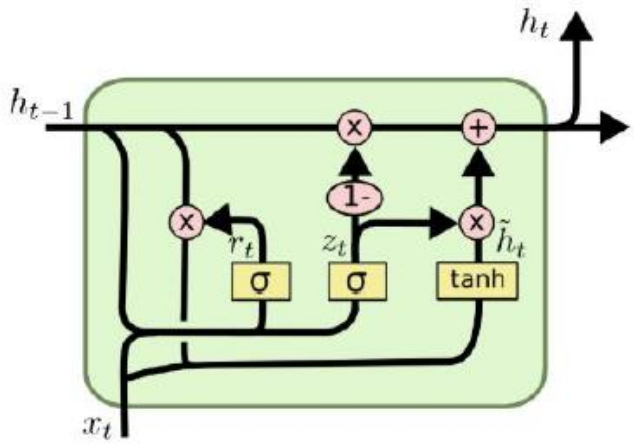

Figura 8. Gated Recurrent Unit. (http://colah.github.io/posts/2015-08-Understanding-LSTMs)

2.4. Natural language understanding

La compresión del lenguaje natural cuyas siglas en ingles es NLU se encargan de comprender lo que un humano ha dicho a una máquina, en esta acción hay un componente de predicción ya que la maquina tiene que saber reconocer lo que se le está pidiendo a partir de unos enunciados que ha dicho el usuario, que tienen que reconocer la intención del usuario en tiempo real y mediante lenguaje natural (Sierra Rivera, W. 2020).

Se pueden distinguir 5 tipos de conocimientos en los motores de comprensión del lenguaje natural a continuación, vamos a profundizar en estos:

- Fonológico: Información acerca de los sistemas de sonidos, acentos, entonaciones y sonoridad de las palabras.

- Morfológico: Gestión sobre la estructura de las palabras, por ejemplo, serían los fonemas /s/ y /z/ se añaden a los nombres para indicar que es un plural.
- Sintáctico: Reconocimiento de las reglas gramaticales.
- Semántico: Significados de los distintos elementos de la comunicación y como se combinan para dar un significado global.
- Pragmático: Gestión de las intenciones comunicativas que existen en una frase.

Cada uno de estos 5 tipos de conocimiento se pueden implementar mediante reglas, por lo al inicio la comprensión del lenguaje natural se realizaba mediante sistemas expertos. (Vásquez, A. C., Quispe, J. P., & Huayna, A. M. 2009).

La principal función del NLU es extraer la intención del mensaje proporcionado del usuario. Lo primero que se realiza es analizar los enunciados del mensaje, gracias a este análisis será capaz de extraer la intención y los datos relevantes para que podamos realizar una acción. La obtención de la intención idónea está muy ligada con el verbo de los enunciados y con las entidades que son conceptos específicos del enunciado como el nombre, lugar, fecha, profesión o moneda. (Macias-Huerta, P. I., Cabañas-Rocha, R., Valdespino, C. N. L., & Santamaría-Bonfil, G. 2021).

Por lo que un NLU es un clasificador de intenciones. Los clasificadores están dentro del ámbito del aprendizaje automático. Este busca una relación entre la entrada con la salida, en la siguiente figura podemos ver un ejemplo donde se busca una función f(x) que opere con los datos X para producir las respuestas Y.

Figura 9. Esquema proceso de aprendizaje automático (Realización propia)

En los problemas de clasificación hay que elegir entre una o varias categorías por medio de ejemplos previamente etiquetados, se pueden dividir en problemas de clasificación binarias o multi-clase, en el primero de los casos se utilizan para predecir o estimar la probabilidad de pertenencia de una clase entre 2 posibles, en segundo en vez de tener 2 clases tendríamos un vector multiclases.

Uno de los mayores retos de las clasificaciones en la evaluación del desempeño, debido a que se puede correr el riesgo de ser subjetivo, por lo que es necesario establecer unos métodos analíticos de evaluación. Es lógico pensar que la mejor métrica de rendimiento de un clasificador es poder comprobar el existo que ha tenido en su propósito, una de las herramientas más utilizadas es la Matriz de confusión. Para construirla existen distintos tipos de datos: Los valores reales de las clases, los valores predichos y la probabilidad estimadas de la predicción, generalmente se mantienen dos vectores de datos, uno para la verdad de la clasificación y otro con los valores predichos, ambos tienen el mismo tamaño y orden. Hay que tener en cuenta que los datos de la clase verdadera se conocen a priori y es la clase para predecir del conjunto de datos. Los valores predichos se obtienen de un modelo previamente entrenado. Es un método fiable hasta para clasificadores de una sola clase, un ejemplo es en los clasificadores de correo SPAM, ya que no es lo mismo que la probabilidad sea del 0,9 o del 0.,51. Si dos modelos cometen los mismos errores, pero uno es más capaz de tener en cuenta la incertidumbre, es mejor modelo. Lo ideal es encontrar modelos que tengan mucha confianza en las predicciones correctas y sea precavido en las dudosas, esta estrategia es clase en la gestión de modelos. (Muñoz, J. M. S. 2016).

Para evaluar el desempeño también existe otra vertiente que se denomina métricas de error, a continuación, pasamos a explicarlas:

- Acurracy: Es la ratio de éxito, representa la proporción de predicciones correctas entre el número total de predicciones.

$$\text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

- Ratio de error: Indica la proporción de predicciones incorrectas.

$$\text{error rate} = \frac{FP + FN}{TP + TN + FP + FN} = 1 - \text{accuracy}$$

- Sensibilidad: Es la ratio de los ejemplos positivos correctamente clasificados.

$$\text{sensitivity} = \frac{TP}{TP + FN}$$

- Especificidad: Es la ratio de los ejemplos negativos correctamente clasificados.

$$\text{specificity} = \frac{TN}{TN + FP}$$

- Precisión: Indica la proporción de ejemplos que son verdaderamente positivos. Cuando un modelo predice la clase positiva.

$$\text{precision} = \frac{TP}{TP + FP}$$

- Recall: Indica que tan completos son los resultados, lo mismo que sensibilidad.
- F-measure: También conocida como F1, es una métrica que combina precision y recall. Se utiliza con mucha frecuencia puesto que reduce el rendimiento de un clasificador con una solo métrica. La métrica estándar asume que el precision y recall tienen la misma importancia, pero es posible ponderarlos para dar mayor peso a uno u otro.

$$\text{F-measure} = \frac{2 \times \text{precision} \times \text{recall}}{\text{recall} + \text{precision}} = \frac{2 \times TP}{2 \times TP + FP + FN}$$

2.5. Dialog Management

El gestor de diálogos es el módulo que analiza las intenciones para determinar que respuestas tiene que dar el Asistente Virtual. En muchas ocasiones se tiene que interconectar con un sistema externo para obtener datos o determinar qué acción se ha de realizar. El objetivo de este módulo es elegir una estrategia de comunicación y elegir la infraestructura para entregar el mensaje al usuario. A continuación, vamos a enumerar distintos puntos en los que se encarga este componente:

- Elegir quien dirige la conversación: El usuario, el sistema o ambos. Cuando el usuario dirige la conversión es un gran desafío ya que pueden existir casos que nos indique el usuario, por el cual no estén entrenados. Si por el contrario lidera la conversación el sistema no se dan casos como el anterior, pero por el contrario se convierte en una conversación poco natural y muy encasillada por eso se recomienda utilizar un sistema mixto que dependa del contexto.
- Estrategia de confirmación de errores: En caso de que el asistente virtual no pueda gestionar la conversación, se diseña un sistema para esos casos donde el sistema va a intentar reconducir la conversación para encontrar un punto de entendimiento y pueda continuarse la conversación.

Una de las técnicas utilizadas en este componente es el aprendizaje por refuerzo, gracias a este se consigue encontrar estrategias óptimas de interacción y confirmación (Cahn, J. 2017). Este tipo de aprendizaje se da cuando tenemos un agente interactuando con un entorno:

- El agente se encuentra en un estado “s” y lleva a cabo acciones “a”.
- Al realizar una acción, el agente se mueve a un nuevo estado y recibe una recompensa (reward).

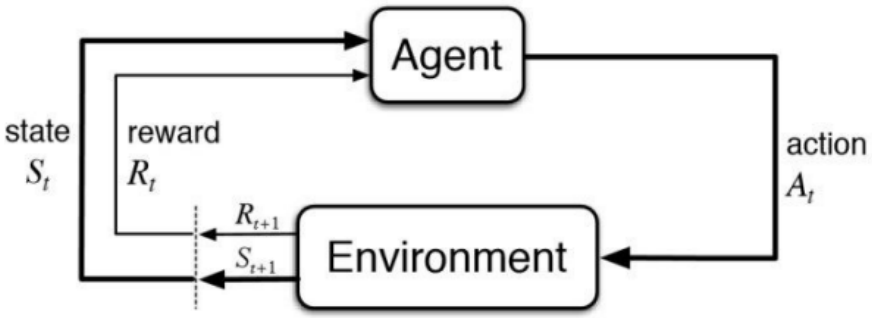

Figura 10 Esquema aprendizaje por refuerzo (https://www.kdnuggets.com/2018/03/5-things-reinforcement-learning.html)

En la anterior figura hemos visto un esquema de cómo funciona el aprendizaje por refuerzo, donde el agente es el modelo que queremos entrenar y que realizara acciones o decisiones, el ambiente será un entorno finito donde nuestro agente interactúe y aprenda.

En este componente del asistente virtual vamos a tener que interactuar con sistemas externos, por ejemplo, un cliente esta interactuando con un asistente virtual y desea saber la temperatura que va a realizar mañana, el sistema tendrá que hacer una llamada a un sistema externo que le devuelva el resultado de la consulta.

2.6. Text to speech

El mecanismo por el cual se convierte el texto en habla se denomina Text To Speech (TTS), principal objetivo es que la forma sonora sea lo más parecido a la del ser humano. Para conseguir el objetivo existen dos factores fundamentales:

- Naturalidad: Es el grado de semejanza a la voz del ser humano.
- Inteligibilidad: El grado de entendimiento que realiza el ser humano del audio generado.

Por lo que ambos factores deben tener un umbral de precisión alta para que se pueda generar una síntesis de voz de calidad. Hay que destacar que un TTS debe de ser capaz de expresarse con un determinado estado de ánimo o acento. (García Romillo, V. 2019).

Para conseguir estas características propias de la voz humana se desarrolló el lenguaje SSML. La principal característica es que ofrece más control sobre el fragmento hablado a partir del

código generado, al ser un lenguaje de marcado como el HTML nos permite insertar etiquetas en el texto que se va a reproducir.

Este componente es fundamental en un Asistente Virtual ya que debemos de indicar al usuario si su intención se ha realizado correctamente o por el contrario requerimos de más información para poderla llevar a cabo. En la siguiente figura podemos ver un ejemplo de la etiqueta <break/> que nos permite realizar una pausa.

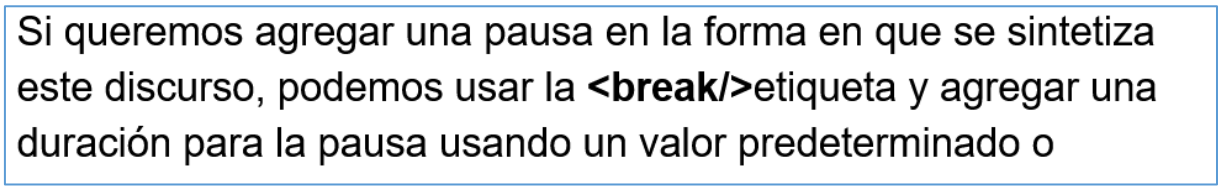

Figura 11. Ejemplo etiqueta break (realización propia)

Existen multitud de guías de referencia del lenguaje de marcado SSML a continuación vamos a enumerar las operaciones más importantes que se pueden realizar:

- Cambiar volumen y velocidad de la voz.
- Pronunciar en otro idioma.
- Reproducir pausas de párrafos o frases.
- Realizar una reproducción fonética concreta, utilizando Ipa o X-sampa.
- Controlar la velocidad, volumen y tono del habla.
- Pronunciar caracteres como fracciones, dígitos, cardinales, fechas, direcciones …
- Susurrar.

Los primeros proyectos de situación de voz se realizaban mediante sistemas expertos. Hoy en día se basan en arquitecturas de aprendizaje profundo como la que podemos apreciar en la siguiente figura.

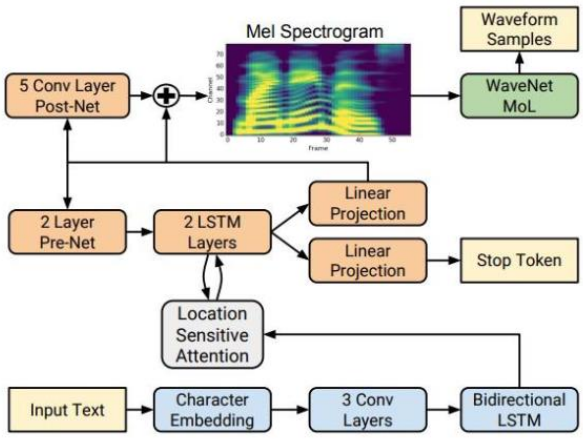

Figura 12. Arquitectura de bloques del sistema Tacotron 2 (https://doi.org/10.1109/ICASSP.2018.8461368)

Esta arquitectura representa un sistema Text To Speech conseguir sintetizar la voz y que se aparezca a la de un humano pasa por distintas redes neuronales artificiales cuya descripción es la siguiente:

- *Character embedding*: Transforma los caracteres en vectores de número reales, estos vectores tienen son de 512 dimensiones esta representación es muy útil para un entrenamiento de RNA's.
- *Conv Layers*: Cada capa tiene 512 filtros con un filtro de 5, lo que se consigue con estas capas convolucionales es normalizar los vectores generados en el componente anterior.
- *Biditectional LSTM*: Hacen de decodificador para el espectrograma.
- *Mel Spectrogram*: Emite una onda de 24 kHZ compatible con la arquitectura del siguiente modelo que es que vamos a explicar a continuación.
- *WaveNet*: En una red neuronal artificial convolucional entrenada y diseñada para que dada una señal de entrada sea capaz de sintetizarla en una función de salida. Una de las principales características es que es capaz de sintetizar voces distintas.

2.7. Arquitecturas de sistemas de IA.

Uno de los principales desafíos de los sistemas de aprendizaje profundo es la cantidad de recursos hardware que consumen, sobre todo de GPU. Por otro lado, el tiempo de respuesta

al usuario es fundamental ya que el lenguaje natural es en tiempo real, por lo que cualquier demora supondrá que el humano no sea capaz de comprender la conversación.

Los asistentes virtuales tienen que ser capaces de ser concurrentes ya que varios usuarios pueden estar haciendo peticiones al mismo tiempo, por ejemplo, si el asistente virtual es de recomendaciones y varios usuarios se conectan a la vez si el sistema no fuera concurrente los últimos tendrían que esperar demasiado tiempo, no haciendo eficiente las recomendaciones. Esta concurrencia se ha conseguido tradicionalmente con varios procesos o hilos que están a la espera de recibir peticiones, hoy en día la principal arquitectura es publicar modelos previamente entrenados, ya que por un lado reducimos el tiempo de entrenamiento y por otro podemos versionar los modelos, pudiendo volver atrás en caso de que la nueva versión del modelo tenga peor desempeño. En la siguiente figura podemos apreciar una arquitectura con un modelo previamente entrenado, primero vemos un componente que es donde realizamos los ajustes y el entrenamiento, se genera una nueva versión que se publica en el servidor de modelos y aplicaciones externas mediante peticiones REST hacen uso del modelo. (Astobiza, A. M. 2021).

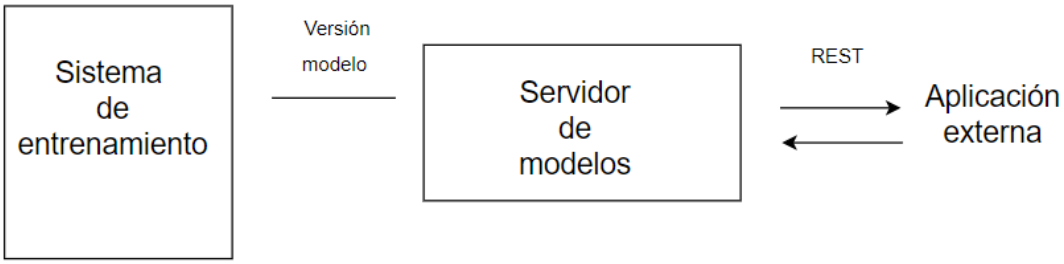

Figura 13. Arquitectura de un sistema de IA con interconexión con terceras aplicaciones (Realización propia)

Estos servidores de modelos tienden a estar en la nube, ya que así podemos crecer y decrecer de hardware según necesidades de la producción.

Hay que tener en cuenta que los sistemas de entrenamiento, donde se aloja el código de las RNAS y el servidor de modelos sólo es una parte de todo el conjunto para poder poner un sistema de IA en producción, tenemos que contar con los siguientes componentes adicionales:

- Colecta y tratamiento de datos: Elegir los datos que se van a utilizar en nuestros modelos, verificando su consistencia y corrección.
- Herramientas de análisis: Para verificar que los retardos y desempeño están dentro de unos umbrales de calidad.

Por último, como hemos comentado en párrafos anteriores los modelos se tienen que versionar. Muchos sistemas se tienen que versionar con mucha frecuenta, por ejemplo, un modelo que caracterice el análisis de sentimientos del Twitter no será igual un día tranquilo del mes de agosto que una noche postelectoral. En estos casos se hace necesario automatizar la pipeline de entrenamiento y puesta en producción, añadiendo una mayor complejidad al sistema. Una continuous training pipeline automatiza el proceso de recolecta de datos, entrenamiento del modelo y deployment del modelo a servidores. En la siguiente imagen podemos apreciar una arquitectura con todos puntos vistos en este apartado.

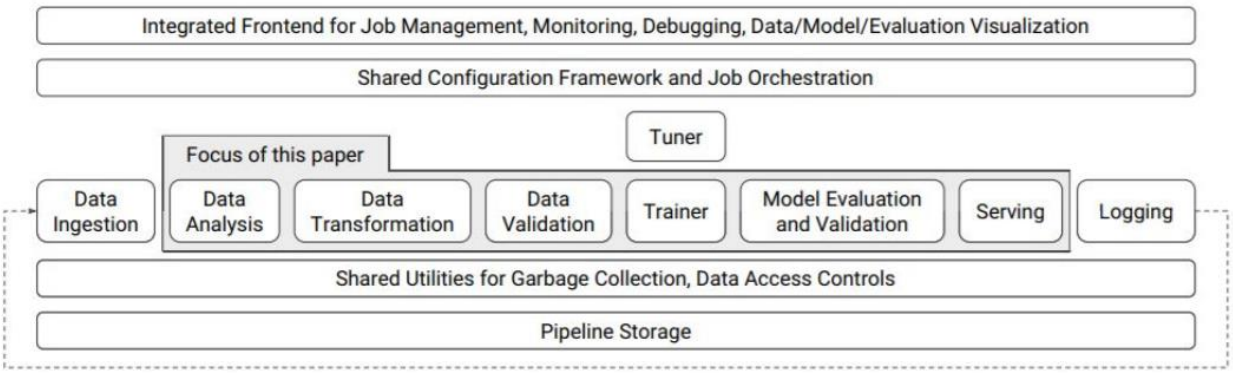

Figura 14. Continuous training pipeline (https://dl.acm.org/doi/10.1145/3097983.3098021)

En sistemas de Asistentes Virtuales la elección de una buena arquitectura es crucial debido a los retardos y variaciones del retardo que pueden provocar los distintos componentes que lo forman y el entorno. A continuación, vamos a describirlos:

- Equipo del usuario: Donde está localizado el micrófono y altavoz donde el usuario va a interactuar.
- Conexión a internet: Como hemos visto en este punto, el modelo cognitivo va a necesitar que gran infraestructura de computación, esto solo se puede proporcionar

en data center, por lo que las peticiones del usuario al asistente virtual se van a realizar por internet.

- ASR: Coste de computación del reconocedor de voz.
- NLU: Coste de computación de la comprensión del lenguaje natural.
- DM: Coste de computación de la gestión del dialogo e interconexión con terceras aplicaciones.
- TTS: Coste de computación en la síntesis de voz para indicar al usuario la respuesta del asistente virtual.

Hemos podido apreciar hasta 6 puntos donde podemos tener retardos en la comunicación, ya sea por la red, como por el tiempo de computación del componente y por último retardos de interconexión de los propios componentes.

3. Contextualización

Durante el estado del arte se han detectado componentes necesarios para la implementación y desarrollo de un asistente virtual, pasamos a enumerarlos:

- ASR: Es el reconocedor de VOZ, este es el componente encargado de la transcripción de la voz a texto.
- NLU: La entrada de este componente es el texto que fue generado por el ASR, la misión de este componente es identificar la intención y los posibles datos relevantes que nos ha indicado el usuario de la aplicación.
- DM: Este módulo obtiene datos relevantes de la intención, a partir de esto toma una decisión, es muy probable que dependa de un backend externo para obtener datos, por lo que tiene que ser fácilmente integrable con terceras aplicaciones
- TTS: A partir de la acción que se ha realizado en el DM, este módulo transforma el resultado de la acción de texto a voz para que el usuario pueda saber cómo ha terminado su petición.

Vamos a poner un primer ejemplo de caso de uso de un asistente virtual para empresa. Las salas de reuniones es un recurso escaso de la organización, ya que la mayoría de las veces están vacías, pero pueden tener un pico de ocupación. Esto lleva a que se tenga que

implementar un sistema de reservas de sala. Un usuario de la organización podría enviar el siguiente orden a un asistente virtual: “Deseo reservar una sala para 4 personas el día 23 de septiembre”, a continuación, vamos a describir el flujo que tendría que hacer esta comunicación por los componentes del asistente virtual:

- ASR: Transformaría el audio en texto, por lo que los datos de salida ya sería el siguiente literal: “Deseo reservar una sala para 4 personas el día 23 de septiembre”.
- NLU: Este componente detectaría que se quiere realizar una acción de reserva de una sala con los siguientes parámetros:
    - Num_personas: 4
    - Fecha: 23/09/2022
- DM: Recibiría la acción a realizar y haría una petición a un backend para realizar la reserva el día indicado por el usuario, también debería tener en cuenta que debe de ser una sala donde como mínimo entren 4 personas. La salida de este módulo dependerá de la respuesta del backend del sistema de reserva:
    - Reserva realizada en la sala Y a la hora y fecha solicitada.
    - Reserva no realizada debido a que no hay salas en la hora y fecha solicitada.
- TTS: La respuesta del DM es transformada a voz para que el usuario sepa el estado final de su petición.

En la siguiente ilustración podemos ver un mapa de los componentes y el flujo descrito en los puntos anteriores.

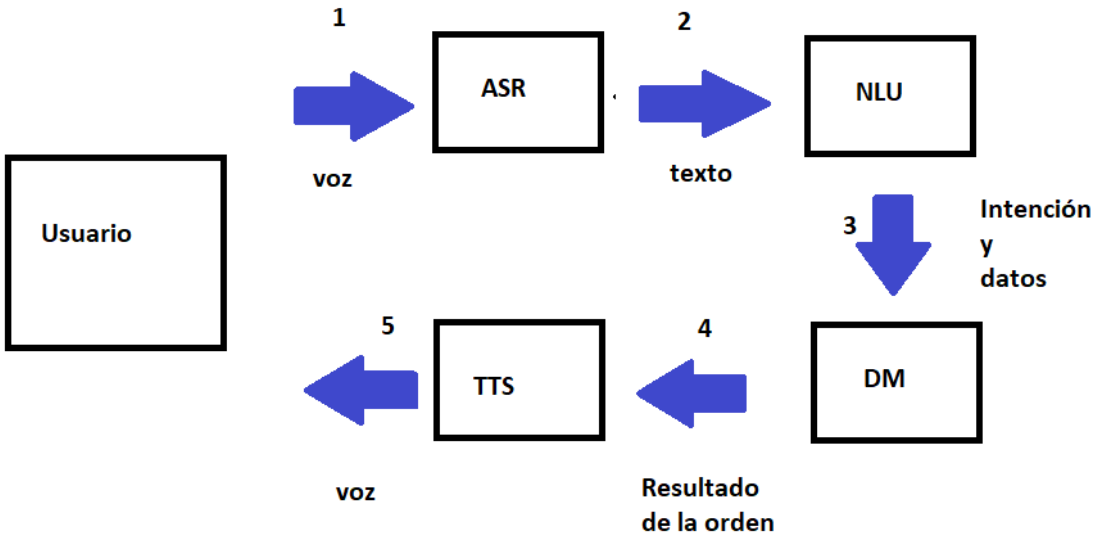

Figura 14. Mapa del flujo de una orden en un asistente virtual (Realización propia)

Una de las necesidades principales de este proyecto es encontrar infraestructura para el desarrollo de estos componentes, como hemos visto en el marco teórico de este TFG, todos estos componentes dependen en gran medida de dos factores, por un lado, un buen desempeño está muy ligado al aprendizaje profundo y aun buen conjunto de datos, y por otro lado una infraestructura con capacidad de un gran procesamiento a nivel de GPU y unos retardos muy ajustados ya que toda la comunicación tiene que ser en tiempo real.

Dos de los más importantes proveedores de infraestructura y servicios cognitivos son Amazon y Google, ambos además disponen asistentes virtuales del hogar lo cual les permiten tener un amplio espectro de entrenamiento. A continuación, vamos a analizar las tecnologías que ofrecen estas empresas.

3.1. Amazon Lex

Es un servicio de AWS que nos permite crear conversaciones para aplicaciones tanto de voz como de texto. Dispone de gran variedad de idiomas y permite un despliegue rápido con latencias muy pequeñas en cualquier punto del mundo, alto rendimiento a nivel de GPU y redundancia. Tiene integrado tanto el módulo ASR como NLU, aunque el servicio ASR se puede utilizar de forma independiente y se denomina Transcribe, por otro lado, para gestión de TTS

se utiliza un servicio que se denomina Polly y por último para gestión del DM se utiliza funciones Lambda. A continuación, vamos a describir cada servicio que proporciona AWS.

3.1.1. AWS Transcribe.

Como hemos indicado anteriormente el componente ASR se denomina Transcribe, este nos permite pasar la voz a tiempo. Permite formato de archivo FLAC, MP3, MP4, Ogg, WebM, AMR o WAV, los archivos tienen las siguientes características; menos de 4 horas de duración y menos de 2 GB de tamaño (500 MB para trabajos de análisis de llamadas).

Los modelos de idioma personalizados utilizan los datos de textos (datos de entrenamiento) para mejorar la precisión de la transcripción de su caso de uso específico. Por ejemplo, podemos proporcionar a Amazon Transcribe términos o acrónimos específicos del sector que podría no reconocer de otro modo. Los acrónimos permiten una acepción mucho más libre de fragmentos de palabras, sean iniciales o no, para formar una palabra cuyos términos no se separan y cuya letra inicial únicamente va en mayúsculas.

Permite la inserción de información mediante *streaming* mediante las siguientes API: AWS SDKs, HTTP/2, *WebSockets* y AWS Management *Console*. Asimismo, se pueden almacenar grabaciones y enviárselas al servicio Transcribe por lotes, en este caso tendríamos un procesamiento *off-line*. En ambos casos se pueden utilizar scripts para almacenaje del texto generado en una base de datos o un contenedor.

Además, dispone de la funcionalidad de identificación de información personal. Detectando información sensible y transcribirla con asteriscos. Un ejemplo sería el número de tarjeta de crédito de un cliente, donde por protección de datos o posibles fraudes no queremos que se transcriban a texto.

3.1.2. AWS Polly

Es un servicio en la nube que convierte el texto en un segmento hablado, está basado en redes neuronales (NTTS), como variable de entrada se introduce el texto que se desea sintetizar y el servicio devuelve una secuencia de audio. Hay que destacar que permite que el formato de entrada sea sin formato o con formato SSML, este último formato como hemos visto en punto 2 de este TFG nos permite modelar la voz de salida para que no sea uniforme. Respecto al

formato de salida el servicio nos permite varios formatos de audio como MP3, PCM u Ogg Vorbis, siendo el primero el más común del mercado mientras que los dos siguientes nos permiten especialización en IOT o en sistemas móviles.

3.1.3. AWS Lambda

Es un servicio que proporciona AWS que permite ejecutar código sin necesidad de aprovisionar o gestionar servidores, por lo que sólo hay que subir el código al servicio y este se va a encargar de ejecutar y escalar el código en un sistema redundante. Otra de las principales características es que el código puede ser invocado en tiempo real por otro servicio de AWS o por algún evento externo. Este código puede acceder a servicios propios de AWS como DynamoDB para acceder a datos o hacer peticiones mediante API aplicaciones. Por último, nos permite utilizar varios lenguajes de programación como Node.js, Java, Python, Go o C#, el código fuente se puede segmentar en funciones Lambda lo que nos permite dar modularidad a nuestro código. En la siguiente figura podemos ver un mapa de la solución.

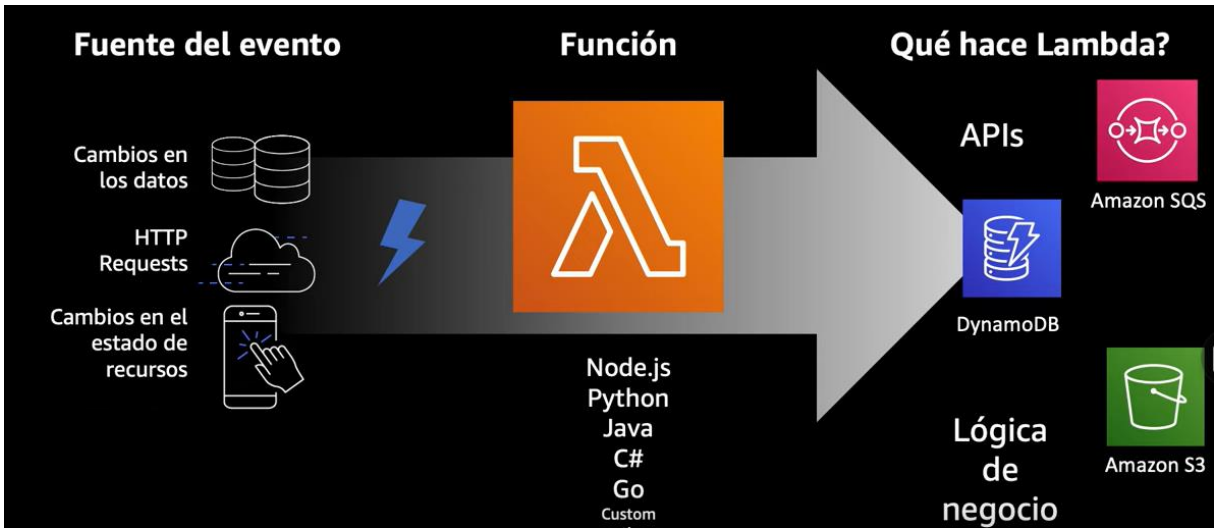

Figura 15. Esquema funcionamiento Funciones Lambda (AWS Lambda)

Al ser un servicio propio de AWS la interconexión con servicios vistos en este TFG como Polly, Transcribe o Lex es nativa, por lo que no hay problemas de permisos de ejecución o transferencia con retardos de datos. Por lo que se pueden crear *endpoints* bajo demanda

orientados a los componentes que se necesitan para crear un asistente virtual sin necesidad de gestionar y administrar recursos de servidores.

3.1.4. AWS Lex v2.

Es el servicio de AWS para crear interfaces de conversación, con este servicio vamos a poder realizar tareas automatizadas, pudiendo comprender los valores de entrada proporcionado por el usuario. Este servicio nos va a permitir crear bots que a su van a poder conversar en uno o más idiomas y el despliegue en una o más regiones.

Dentro de este servicio está integrado los servicios de AWS Transcribe, Polly y Lambda por lo que desde un único punto de configuración nos va a permitir configurar todos los servicios. Podemos indicar que además de ser el servicio de compresión de lenguaje natural, también hace de gestor del resto de servicios asociados.

Las intenciones son la representación de la acción que quiere realizar el usuario, este servicio permite hasta 100 intenciones por bot creado. Las intenciones tienen los siguientes campos obligatorios:

- Nombre de la intención: Seria el literal que define a la intención, cabe destacar que como hemos visto en el párrafo anterior, el servicio permite hasta 100 intenciones que el nombre tiene que ser lo más identificativo posible.
- Enunciados de muestra: Serian textos de como el usuario puede manifestar esa acción, AWS recomienda 10 mínimo, cabe destacar que Lex es un sistema cognitivo por lo que necesita estos datos para entrenar no para hacer match como ocurriría en un sistema experto.
- Cumplimiento de la intención: Que acción se va a llevar a cabo si se cumple una intención y el usuario ha proporcionado la información necesaria, en esta funcionalidad hay integración nativa con funciones Lambda, así el NLU se puede comunicar con el DM.

Una intención puede requerir de cero o más parámetros para poder ejecutar la acción, estos se pueden capturar del enunciado inicial del usuario o mediante conversación desde la

intención. Estos parámetros se denominan dentro del servicio *slots* o ranuras existiendo dos tipos:

- Propios: Serian los proporcionados por el servicio como ejemplo fecha, hora o ciudad.
- Integrados: Son personalizados y los puede configurar el administrador del servicio.

La conversación que se está diseñando dentro de la intención puede verse en tiempo real. Esta utilidad se denomina flujo de conversación, en el siguiente pantallazo se puede apreciar esta funcionalidad.

Figura 16. Herramienta de flujo de conversación (AWS Lex)

El servicio dispone de intenciones preinstaladas a continuación vamos a describirlas:

- Cancelación: Responde ante enunciados que sugiera que el usuario desea cancelar la conversación, desde esta intención tenemos acceso a los slots capturados y a funciones Lambda por si tenemos que realizar una labor de eliminación antes de cancelar la conversación.
- Ayuda: En este caso responderá ante palabras o frases que indique el usuario necesita ayuda, en caso de no conseguir proporcionar esa información tiene la capacidad de entregar la conversación a un humano.

- Pausa: Se entrará a esta intención cuando se detecte que el usuario quiere una pausa en la conversación para seguir más adelante, en este caso todas los datos y flujo se guardaran en unas variables de sesión para que en un futuro se puedan rescatar.
- Repetición: Cuando el usuario requiere de una repetición de un mensaje anterior, en esta intención se requiere de una función Lambda para guardar toda la información y poder rescatar el mensaje anterior que requiera el usuario.
- Inicio: El usuario desea comenzar de nuevo, el mecanismo es muy parecido al de cancelación salvo que en este caso se iniciará la conversación.

Este servicio dispone de una herramienta de monitorización para la gestión de disponibilidad y desempeño. El sistema nos proporciona por un lado el número de peticiones con los retardos de las peticiones que se hacen al servicio, por otro el número de conversaciones que han tenido éxito. Las que conversaciones que no han tenido éxito nos va a indicar en que parte se han quedado, por ejemplo, una intención donde el usuario ha cancelado la conversación.

3.1.5. CloudFormation

Es un servicio que proporciona AWS para la generación de plantillas, estas pueden incluir servicios o código fuente, gracias a estas plantillas podemos administrar soluciones software, estas pueden ser despliegues, borrados, el complimiento de configuraciones o control de versiones. A continuación, vamos a describir las distintas funciones que tiene este servicio:

- Plantilla: En este apartado se añaden los servicios, atributos, configuraciones, dependencias y estados. Una vez creada la plantilla, el servicio creará un fichero JSON o YAML que contendrá la estructura y datos de la plantilla.
- Conjunto de cambios: Esta funcionalidad nos permite realizar cambios en nuestra plantilla, un ejemplo seria la adicción de un nuevo servicio con sus configuraciones *default*.
- Pila (*Stack*): Es el grupo de recursos y sus estados previstos una vez que el servicio haya ejecutado la plantilla.
- Conjunto de pilas (*StackSet*): Es un grupo de pilas que se pueden desplegar a través de cuentas o regiones, esta funcionalidad nos asegura que independientemente de la

región donde esta alojada nuestra solución siempre es la misma, realizándose un único despliegue para todas las regiones donde este configurado.

3.1.6. Alexa

Alexa es el asistente del hogar de Amazon, es uno de los más populares del mundo. Actualmente está presente en millones de hogares, este asistente está basado en los servicios de AWS descritos en los puntos anteriores. Debido la utilidad que puede tener en nuestro TFG vamos a describir el flujo de su funcionamiento, vamos a utilizar el siguiente ejemplo: "¿Qué tiempo hará mañana?".

1) Alexa dispone de un aparato hardware que se denomina Amazon Echo, este dispone como mínimo un micrófono, altavoz y capacidad de almacenamiento, pudiendo llegar a disponer de pantallas táctiles. Otra de las características es que puede haber más de uno en el hogar. Amazon Echo empieza a procesar con una palabra de activación, que por defecto es "Alexa".
2) El usuario cuando pronuncia "Alexa, ¿Qué tiempo hará mañana?" el dispositivo Echo se prepara para recibir una orden y en el dispositivo se genera un fichero de grabación con la voz del usuario que ha dicho después de la palabra Alexa.
   En esta etapa el procesamiento se realiza en el dispositivo Echo como podemos ver en la siguiente figura, en color verde el dispositivo Echo que está recibiendo la orden.

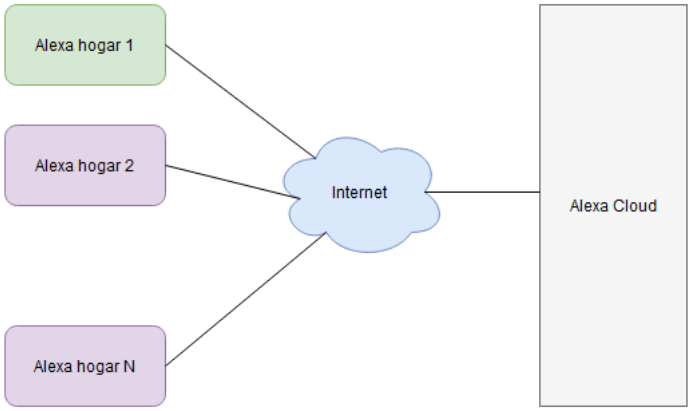

Figura 17. Flujo funcionamiento Alexa cuando recibe una orden (Realización propia)

3) La grabación viaja por internet a la nube de Alexa, donde el servicio Transcribe transforma la grabación en texto.
4) El servicio Lex detecta la intención, en este caso que se quiere saber el tiempo y los slots, que sería del día de mañana.
5) Mediante una función Lambda consultaríamos mediante API un servicio de meteorología, una vez que obtenemos el resultado construimos un mensaje de texto que la respuesta que queremos proporcionar al usuario.
6) El servicio Polly a partir del texto proporcionado por función Lambda construirá una grabación que se envía al dispositivo Echo, estos 4 últimos puntos se procesan en la nube en color ver cómo se puede apreciar en la siguiente figura.

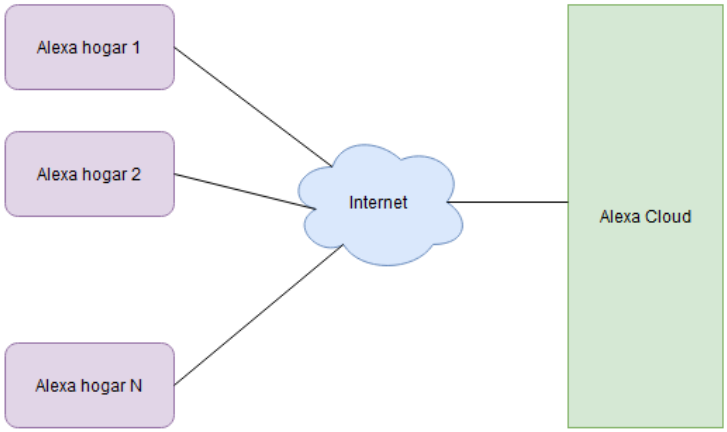

Figura 18. Flujo funcionamiento Alexa cuando se procesa una orden (Realización propia)

7) La grabación se recibe por el dispositivo Echo y se reproduce. Esto ya se realiza en el dispositivo del hogar, como podemos observar en la figura 19.

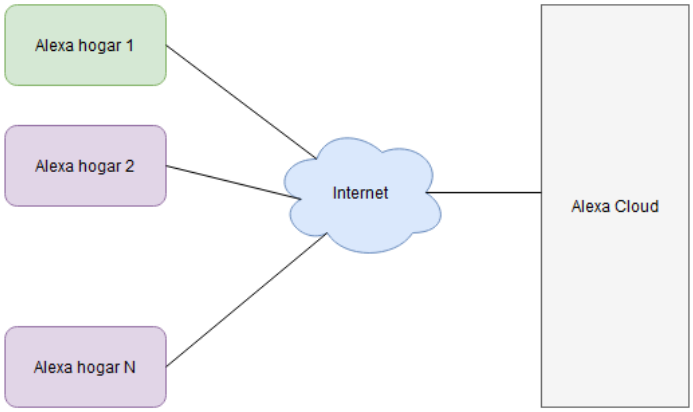

Figura 19. Flujo funcionamiento Alexa cuando se reproduce una orden (Realización propia)

3.2. Dialog Flow

El servicio de Google para crear comunicaciones sofisticadas con los usuarios al igual que AWS dispone de todos los componentes necesarios y analizados en el apartado 2 de este TFG, a continuación, vamos a describirlos:

3.2.1. Google Text-to-Speech

Es el servicio de sintización de Google, dispone de hasta 220 voces en 40 idiomas, Donde 90 voces utilizan aprendizaje profundo y el resto están basadas en sistemas expertos. Al igual que AWS tiene compatibilidad con SSML permitiendo añadir números, pausas, horas, personalizar tonos, modificar la velocidad de elocución modificar las ganancias tanto para subir el volumen como disminuirlo, trabajando en el siguiente margen de -96 dB hasta 16 dB.

Mediante aprendizaje automático permite personalizar voces, el resultado es una voz muy parecida a los datos que se ha facilitado. Con esto conseguimos una voz única, pero en contrapartida el tiempo de entrenamiento puede durar menses, no existiendo por parte de Google un tiempo máximo de entrega.

Esta totalmente integrada con otros componentes cognitivos de Google como el ASR o NLU, además de existir una API REST o Grpc, los formatos de audio que soportar son MP3, LINEAR16 y Ogg Opus. Por último, permite adaptar el audio que se está generan al tipo de dispositivo donde se va a reproducir como altavoces, líneas móviles o auriculares.

3.2.2. Cloud Speech-to-Text

Es el servicio de transcripción de Google, está basado en algoritmos de redes neuronales, permitiendo crear modelos propios a partir de uno base. A continuación, vamos a describir las principales características:

- Vocabulario internacional: Permite hasta 125 idiomas y variantes lingüísticas.
- Reconocimiento de voz en *streaming*: La transcripción se puede realizar en tiempo real conforme lleguen los datos de la voz a la API.
- Por lotes: Se pueden realizar transcripciones en *off-line* en este caso el servicio permite tratar un lote de grabaciones al mismo tiempo.
- Adaptación de voz: Dispone de clases que convierten automáticamente números, direcciones, fechas o divisas.
- Speech-to-Text On-Prem: Permite la interconexión de infraestructuras on-premises con el servicio ASR, pudiéndose instalar en los servidores de clientes.
- Tratamiento del ruido: Es muy fiable en ambientes de ruido intrínseco como una fábrica o una tormenta.
- Modelos para dominios específicos: Detecta medios específicos como internet o una llamada por línea móvil, adaptándose el algoritmo para llegar a la mejor transcripción posible
- Filtrado de contenido: Permite un filtrado de contenido como datos sensibles del usuario o palabras inapropiadas.

Además, dispone de las siguientes características que están en Beta:

- Evaluación automática: De forma dinámica inserta puntos, comas o signos de interrogación sin intervención humana.
- Diarización de interlocutores: Es capaz de reconocer interlocutores y poder identificar en la transcripción quien ha dicho una u otra frase.

3.2.3. Google Cloud Functions

No necesitamos servidores para la ejecución de código, esta infraestructura se genera automáticamente cuando estamos ejecutando nuestro código, este se tiene que alojar

previamente en contenedores que se piden bajo demanda también a la infraestructura de Google. Otra de las principales ventajas es que podemos ejecutar código en cualquier parte del planeta sin tener que adquirir o administrar infraestructura.

3.2.4. DialogFlow.

Es la parte de comprensión del lenguaje natural, en la siguiente figura podemos apreciar el flujo de una conversación telefónica que proporciona el fabricante.

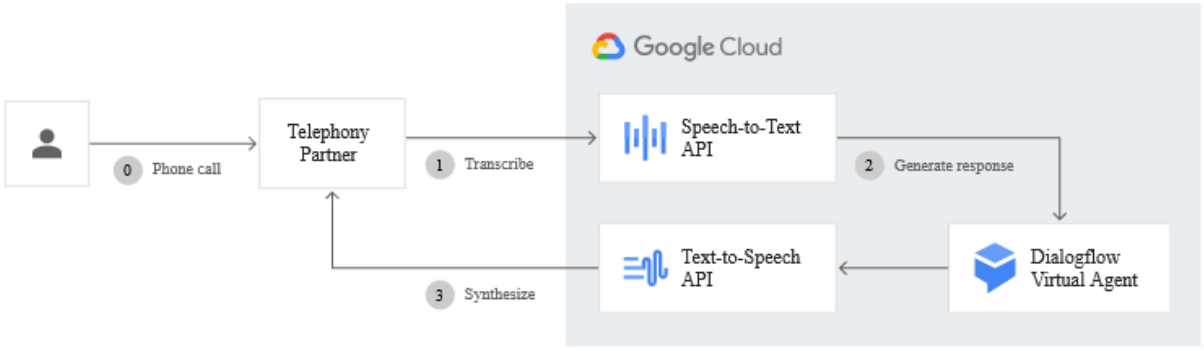

Figura 20. Flujo funcionamiento Asistente virtual Google (Google Dialogflow)

En la figura podemos analizar que el flujo es similar a lo que hemos estudiado en el marco teórico o el funcionamiento de AWS Alexa, la voz de del usuario se transcribe en este caso mediante el servicio Speech-to-Text de Google, una vez que tenemos la conversión en texto esta llega a Dialogflow, en este caso el componente NLU y DM está integrado de forma nativa en Dialogflow, por último se genera una respuesta que tiene que llegar al usuario, en este caso se realiza mediante el servicio Text-to-Speech llegando al usuario el mensaje sintetizado por línea telefónica.

Por otro lado, a nivel de gestión de intenciones también hay un alto grado de similitud con AWS Lex v2 y lo analizado en el marco teórico de TFG. El concepto de clasificar las acciones del usuario es el mismo, además del entrenamiento mediante aprendizaje automático, este concepto como hemos indicado en varios puntos de este de TFG indica que los enunciados que pongamos de ejemplo para la obtención de la intención no tienen por qué hacer *match* sino que el sistema aprende y es capaz de reconocer distintas variaciones, pero con el mismo

sentido. En la siguiente figura podemos apreciar el funcionamiento de las intenciones de Dialogflow.

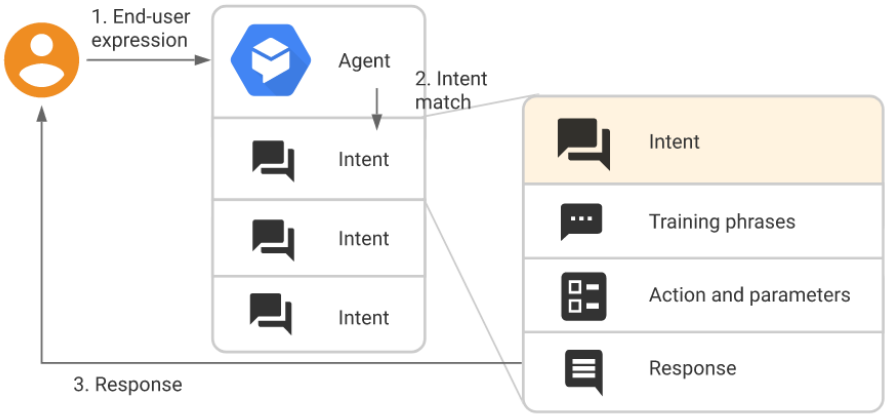

Figura 21. Funcionamiento intenciones (Google Dialogflow)

A continuación, vamos a analizar la figura 21, el usuario expresa una acción, esta llega a DialogFlow en formato texto y se realiza las siguientes acciones:

- El servicio encontrara la intención idónea, hay que destacar que en la imagen aparece el literal *Intent match*, esto significa que busca la coincidencia en la intención, pero no significa que el enunciado proporcionado por el usuario tiene que coincidir exactamente con los configurados por en las distintas intenciones.
- Cada intención tiene unos enunciados de entrenamiento, se podrán capturar datos que ha proporcionado el usuario y generar una acción por ejemplo consultar a un backend un dato que se necesite proporcionar al usuario, por último, indicar que permite generar la respuesta al usuario.

A nivel de intenciones predeterminadas el sistema posee dos:

- Mensaje de bienvenida: Esta intención es la encarga de inicializar la conversación.
- Resguardo: Llegaremos a esta intención cuando no se haya encontrado ninguna coincidencia en las intenciones configuradas.

Las acciones se pueden invocar a un sistema externo o mediante código ejecutado en el servicio Google Cloud Functions. En el siguiente diagrama podemos apreciar el funcionamiento. Una conversación en formato texto no puede llegar por distintos canales,

desde Dialogflow se realiza la parte de reconocimiento de intenciones y datos, después se invoca a cloud Functions donde podemos conectarnos con sistemas externos, el diagrama se conecta con una Api y con Datastore.

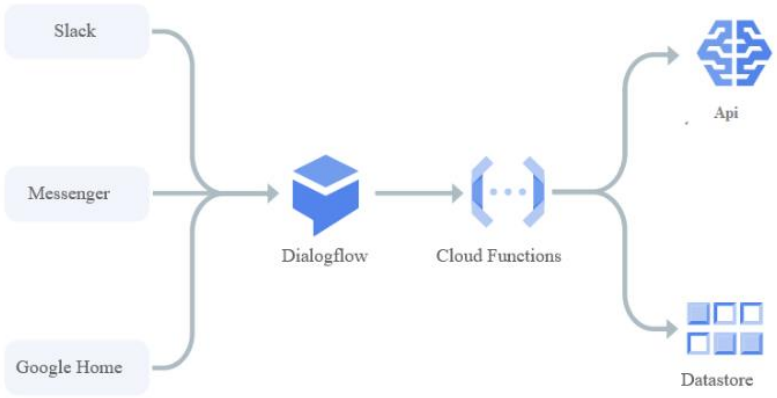

Figura 22. Funcionamiento Google Cloud Functions (Google Cloud)

Respecto a la monitorización dispone de una herramienta de analítica en tiempo real, el principal objetivo de esta funcionalidad es recoger la lista de fallos para mejorar el rendimiento y calidad del sistema, a continuación, pasamos a indicar un par de ejemplos que nos podemos encontrar en esta herramienta:

- Las frases de entrenamiento de las intenciones son similares, nos alerta que las frases que estamos utilizando están provocando un *overfitting*.
- La intención tiene un parámetro que no se ha entrenado. En este caso nos alerta que existe alguna ranura que no estamos realizando ninguna acción para capturar ese dato.

A nivel de métricas disponemos un de un amplio número de vistas que se pueden combinar con datos de intenciones, parámetros o peticiones. En la siguiente figura disponemos de una gráfica proporcionada por la herramienta, donde nos indican en qué estado han terminado las interacciones con DialogFlow a lo largo del tiempo.

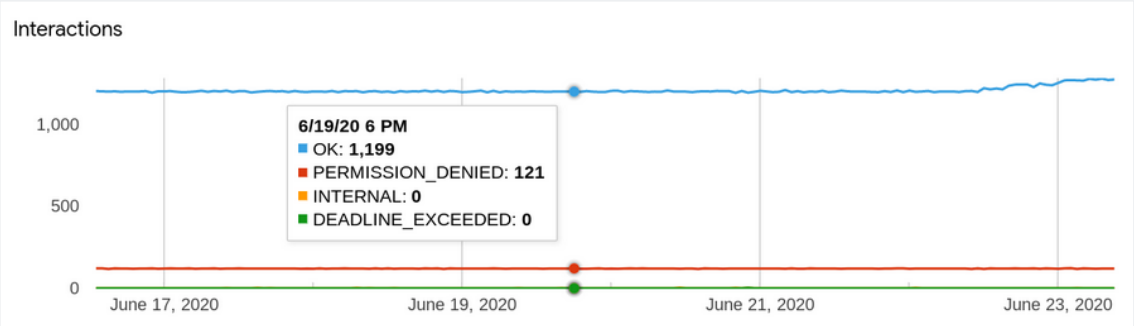

Figura 23. Estado final de las interacciones (Google Dialogflow)

3.2.5. Ok Google

Ok Google, es uno de los asistentes del hogar más populares del mundo, al igual que Alexa se basa en la arquitectura y servicios que hemos visto en puntos anteriores. En la figura se puede ver la arquitectura de Ok Google.

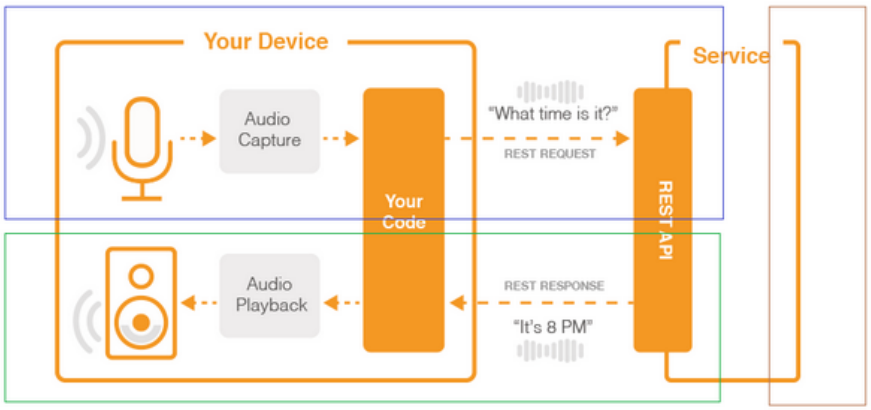

Figura 24. Esquema general de un asistente virtual (Ok Google)

Hemos podido apreciar en la figura 24 que la orden de captura por el micrófono del dispositivo se encuentra en el hogar, ese audio viaja a la nube de Google que lo transcribe a texto mediante el servicio Cloud Speech-to-text, el texto llegará a Dialog Flow en el ejemplo de la figura anterior detectaría que es la intención obtener la hora, realizaría una petición para saber la hora actual mediante cloud Functions y finalmente el resultado es sintetizado por el servicio Cloud Text-to-Speech y viaja al dispositivo que finalmente reproduce el mensaje al usuario.

Por último, hay que destacar que Google está trabajando para que no haya palabra de activación y esta tarea se haga por biométrica de voz o mirando fijamente al dispositivo.

3.3. Conclusiones

En este apartado hemos podido apreciar que las dos infraestructuras cloud para desplegar proyectos de asistentes virtuales disponen de unas características similares y que se basan en lo analizado en el punto del marco teórico de este TFG.

Los factores fundamentales a la hora de poner en marcha un proyecto de estas características son los siguientes:

- Un gran conjunto de entrenamiento y que este actualizado: Tanto Amazon como Google tiene millones de usuarios que diariamente están entrenando a sus sistemas.
- Despliegue de infraestructura bajo demanda: Ambos sistemas se basan en pago por uso y sus sistemas crecen o decrecen de forma automática sin necesidad de una administración de sistemas.
- Retardos muy pequeños: Son dos de los mayores proveedores cloud y disponen de centros de datos en todo el mundo.

Por otro lado, a nivel de gestión de servicios ambas empresas poseen innumerables herramientas y soportan gran parte de los lenguajes de programación que se utilizan en la actualidad.

Ambas soluciones serian idóneas para la puesta en marcha de nuestro proyecto, pero nos vamos a decantar por AWS debido a que la herramienta de generación de plantillas nos ha parecido muy fácil e intuitiva. Hay que indicar que vamos a crear una web responsive con un cajetín de chat, con capacidad de utilizar el micro y altavoces del ordenador o móvil del usuario. En las pruebas de validación la creación de esta interfaz con AWS CloudFormation ha sido inmediata, mientras que con la herramienta Google Cloud Deployment Manager nos ha sido imposible publicar en internet un piloto conceptual de la solución.

4. Diseño de la propuesta

4.1. Objetivos

A continuación, pasamos a describir los objetivos de este TFG.

4.1.1. Objetivo General

Vamos a diseñar una herramienta empresarial, un Asistente Virtual con las siguientes características:

- Capacidad de interactuar mediante voz o texto.
- Operativo 7x24 con capacidad de múltiples idiomas y dar servicio a nivel mundial.
- Resolver los siguientes 3 tipos de tareas en una empresa:
    - Comunicaciones mediante voz o texto para realizar gestiones cotidianas en una empresa.
    - Conversaciones mediante texto que requieran del envió o recepción de ficheros.
    - Alertas a empleados para que sean informados de un hecho relevante.

4.1.2. Objetivos específicos

A continuación, vamos a en enumerar los objetivos para llegar a los objetivos indicados en el punto anterior.

- Modelado de las historias de usuario.
- Diseño infraestructura necesaria.
- Creación de la metodología para el proyecto.
- Elección de roles.
- Diseño de conversaciones.
- Desarrollo de las interconexiones con terceros.
- Gestión de analíticas y desempeño.
- Despliegue de la solución.
- Puesta en marcha.

4.2. Contenidos

Este punto lo hemos dividido en 3 apartados, primero expondremos los diagramas de la solución, después realizaremos las historias de usuario y por último explicaremos como hemos desplegado la solución.

4.2.1. Arquitectura de la solución.

La arquitectura propuesta está totalmente desacoplada esto significa que un futuro se podrían intercambiar componentes de distintos fabricantes. En la siguiente figura podemos apreciar el diagrama de nuestra solución.

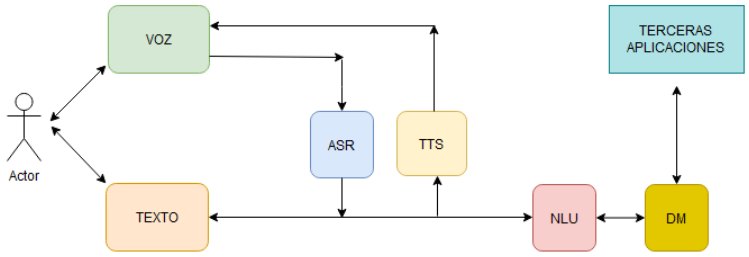

Figura 25. Arquitectura de la solución (Realización propia)

El actor que es el usuario de nuestra solución dispone de una comunicación bidireccional con nuestro sistema ya sea por voz o por canales de texto (Como con un ser humano). En caso de ser un canal de texto este viajara directamente al NLU, mientras que si es por voz tendremos que realizar la transcripción a texto o la síntesis a voz según la dirección de la comunicación. (Elementos ASR y TTS de la figura 25). En el NLU realizaremos la búsqueda de la intención propicia y capturaremos las ranuras que se necesiten en esa acción. En el DM tendremos una interconexión con *backend* de la empresa. Una vez que tengamos la decisión esta volverá al NLU y la enrutará al canal de texto o de voz según como corresponda.

Como hemos indicado en los objetivos generales del TFG nuestro asistente virtual tendrá tres modos de funcionamiento, estos van a depender de los canales ya sea voz o texto, o de quien origina la comunicación el usuario o el sistema. A continuación, vamos a explicar estos 3 tipos de funcionamiento.

4.2.1.1. Tipo A: Comunicaciones mediante voz o texto para realizar gestiones cotidianas en una empresa.

En este modo de funcionamiento el usuario se pondrá en contacto con el sistema tanto por medio de voz como de texto, las comunicaciones no van a requerir él envió o recepción de ficheros.

4.2.1.2. Tipo B: Conversaciones mediante texto que requieran del envió o recepción de ficheros.

En este modo de funcionamiento para que la conversación termine con éxito requiere de envió de ficheros por al menos de una de las dos partes, este requerimiento lleva implícito que la conversación sea únicamente mediante texto.

4.2.1.3. Tipo C: Conversaciones alertas a empleados para que sean informados de un hecho relevante.

En este caso la comunicación empieza desde el asistente virtual, mediante la interconexión con terceras aplicaciones a las funciones Lambda le llegarán evento que va a desencadenar en una comunicación de voz a uno o varios usuarios, una vez que el usuario haya descolgado la llamada comenzará la comunicación.

4.2.2. Historias de usuario.

Se van a realizar 12 historias de usuario que puede realizar nuestro producto, dentro de cada historia vamos a describir los flujos de información de cada componente. Como hemos indicado en el punto anterior tenemos 3 tipos de funcionamiento. Vamos a crear 5 historias de usuario para el tipo A, 5 historias de usuario para el tipo B y las últimas 2 acciones para el tipo C. A continuación, pasamos a describirlas. La nomenclatura que vamos a realizar es primero poner el tipo seguido de un punto y por último el número de la historia. Por ejemplo, la acción A.1 significa que es de tipo A y que es la número 1 de ese tipo. Después de indicar la historia de usuario, vamos a realizar una labor de modelado. Para ello vamos a realizar una tabla donde la primera columna es el componente, la segunda columna son los datos de entrada y la última los datos de salida.

4.2.2.1. Historia de usuario A.1.

El usuario desea realizar una acción de petición de vacaciones, como ejemplo vamos a suponer que el usuario desea tener vacaciones del 24 de diciembre al 4 de enero.

Tabla 2. Modelado de la historia de usuario A1.

Componente	Entrada	Salida
ASR	Voz: Quisiera vacaciones del 24 de diciembre al 4 de enero.	Texto: "Quisiera vacaciones del 24 de diciembre al 4 de enero."
NLU	Texto: "Quisiera vacaciones del 24 de diciembre al 4 de enero."	Intención: Vacaciones. Slots: Desde: 24 de diciembre. Hasta: 4 de enero.
DM	Intención: Vacaciones. Slots: De: 24 de diciembre. Hasta: 4 de enero. Código de empleado.	Enviamos la petición al sistema de gestión de vacaciones y obtenemos si las tiene aceptadas o no. Generamos texto para el siguiente componente.
TTS	Texto de confirmación o denegación de vacaciones.	Voz de confirmación o denegación de vacaciones.

4.2.2.2. Historia de usuario A.2.

El usuario quiere reservar una sala de reuniones para un día determinado, como ejemplo vamos a suponer que nuestro usuario quiere una reserva una sala el próximo 9 de diciembre de 11 a 12 para 4 personas.

Tabla 3. Modelado de la historia de usuario A2.

Componente	Entrada	Salida
ASR	Voz: Necesitaría una sala común para el próximo 9 de diciembre.	Texto: "Necesitaría una sala común para el próximo 9 de diciembre."
NLU	Texto: "Necesitaría una sala común para el próximo 9 de diciembre."	Intención: reservaSala. Slots: Fecha: 9/12/2022 Desde: 11:00 Hasta: 12:00 Personas: 4
DM	Intención: reservaSala. Slots: Fecha: 9/12/2022 Desde: 11:00 Hasta: 12:00 Personas: 4	Petición al sistema de gestión de salas, este nos tendrá que devolver si la reserva esta realizada correctamente o falta algún requisito.
TTS	Texto de confirmación o denegación de la reserva.	Voz de confirmación o denegación de la reserva de la sala.

4.2.2.3. Historia de usuario A.3.

Envió de un paquete por transporte, como ejemplo vamos a suponer que nuestro cliente desea enviar unos documentos legales a un cliente.

Tabla 4. Modelado de la historia de usuario A3.

Componente	Entrada	Salida
ASR	Voz: Quiero enviar unos documentos de forma urgente a cliente.	Texto: "Quiero enviar unos documentos de forma urgente a cliente"
NLU	Texto: "Quiero enviar unos documentos de forma urgente a cliente"	Intención: envioDocumentos. Slots: Tipo: Documento Dirección: Chile 1 pta A Ciudad: Valencia Postal: 46980 Atención: Pepito Perez
DM	Intención: envioDocumentos. Slots: Tipo: Documento Dirección: Chile 1 pta A Ciudad: Valencia Postal: 46980 Atención: Pepito Perez	Petición a la API del sistema de transporte con los datos del paquete, esta nos contestará si se puede hacer cargo, en caso afirmativo fecha y horas de recogida.
TTS	Texto de confirmación con fecha y hora de recogida o mensaje de imposibilidad de tramitar el transporte.	Texto de confirmación con fecha y hora de recogida o mensaje de imposibilidad de tramitar el transporte.

4.2.2.4. Historia de usuario A.4

El usuario desea saber condiciones de un proveedor, como ejemplo nuestro usuario pedirá la forma de pago que tenemos actualmente con un proveedor denominado TFG S.L.

Tabla 5. Modelado de la historia de usuario A.4

Componente	Entrada	Salida
ASR	Voz: Necesitaría la forma de pago que tenemos con el proveedor TFG S.L.	Texto: "Necesitaría la forma de pago que tenemos con el proveedor TFG S.L."
NLU	Texto: "Necesitaría la forma de pago que tenemos con el proveedor TFG S.L."	Intención: informacionProveedor. Slots: Nombre: proveedor TFG S.L Tipo: Forma de pago
DM	Intención: informacionProveedor. Slots: Nombre: proveedor TFG S.L Tipo: Forma de pago	Petición al ERP para obtener la forma de pago de ese proveedor. El ERP nos devolverá la forma o por el contrato nos dará un mensaje de error.

TTS	Texto con la forma actual o mensaje de error.	Voz con la forma actual o mensaje de error.

4.2.2.5. Historia de usuario A.5

Saber si un determinado compañero está trabajando o ha tenido alguna ausencia. Como ejemplo nuestro usuario preguntará si un compañero mañana se encuentra disponible.

Tabla 6. Modelado de la historia de usuario A.5

Componente	Entrada	Salida
ASR	Voz: ¿En estos momentos mi compañero Pepito Pérez está disponible?, es que me intento poner en contacto con él, pero no le localizo.	Texto: "¿En estos momentos mi compañero Pepito Pérez está disponible?, es que me intento poner en contacto con él, pero no le localizo."
NLU	Texto: "En estos momentos mi compañero Pepito Pérez está disponible, es que me intento poner en contacto con él, pero no le localizo."	Intención: localizacionPersonal. Slots: Nombre: Pepito Pérez
DM	Intención: localizacionPersonal. Slots: Nombre: Pepito Pérez	Petición al sistema de fichaje del personal para saber si se encuentra operativo. Devolveremos si se encuentra operativo o si por el contrario ha declaro alguna ausencia.
TTS	Texto con la confirmación o información de ausencia.	Voz con la confirmación o información de ausencia.

4.2.2.6. Historia de usuario B.1

Obtención de nóminas, como ejemplo vamos a suponer que nuestro usuario desea las 3 últimas nóminas. En este caso requiere del envió de ficheros por parte del sistema al usuario.

Tabla 7. Modelado de la historia de usuario B.1

Componente	Entrada	Salida
ASR	Voz: Quiero mis tres últimas nominas.	Texto: "Quiero mis tres últimas nominas."
NLU	Texto: "Quiero mis tres últimas nominas."	Intención: obtenerNominas Slots: Desde: 6/22 Hasta :8/22

DM	Intención: Nominas. Slot: Meses: (8/22, 7/22, 6/22) Código de empleado.	Resultado de obtener las 3 últimas del ERP de la empresa.
TTS	Texto de Confirmación del envió de las nóminas con los ficheros o de que no ha sido posible la operación.	Voz de Confirmación del envió de las nóminas con los ficheros o de que no ha sido posible la operación.

4.2.2.7. Historia de usuario B.2

Dar de alta una incidencia técnica, nuestro usuario ha tenido una incidencia con la licencia del Office 365 por lo que dará de alta el caso. Se construirá una conversación donde el sistema le pedirá al usuario evidencias.

Tabla 8. Modelado de la historia de usuario B.2

Componente	Entrada	Salida
ASR	Voz: No me funciona el office 365 me da error de licencia.	Texto: "No me funciona el office 365 me da error de licencia."
NLU	Texto: "No me funciona el office 365 me da error de licencia."	Intención: incidenciaTecnica Slots: Tipo: Licencia Evidencias: Pantallazos del caso.
DM	Intención: incidenciaTecnica. Slots: Tipo: Licencia Evidencias: Pantallazos del caso.	Inserción en el sistema Helpdesk de la empresa. Información del número de ticket.
TTS	Texto con el número de ticket de la incidencia.	Voz con el número de ticket de la incidencia.

4.2.2.8. Historia de usuario B.3

Obtención del certificado de IRPF, nuestro usuario desea el certificado IRPF del año anterior para poder realizar correctamente la declaración de la renta.

Tabla 9. Modelado de la historia de usuario B.3

Componente	Entrada	Salida
ASR	Voz: Desde la asesoría me piden el certificado del IRPF para hacer la declaración de la renta.	Texto: "Desde la asesoría me piden el certificado del IRPF para hacer la declaración de la renta."

NLU	Texto: "Desde la asesoría me piden el certificado del IRPF para hacer la declaración de la renta."	Intención: obtencionCertificado Slots: Tipo: Renta Año: 2021
DM	Intención: Certificado. Slots: Tipo: Renta Año: 2021	Petición al ERP y obtención del fichero con el certificado. Construcción con el mensaje de envió o de error.
TTS	Texto de Confirmación con el fichero del certificado, en caso de no poderse enviar mensaje con el tipo de error.	Voz de Confirmación con el fichero del certificado, en caso de no poderse enviar mensaje con el tipo de error.

4.2.2.9. Historia de usuario B.4

Justificación de ausencia, el usuario realizará una conversación de chat para justificar que no ha acudido al trabajo en un determinado horario debido a un examen oficial.

Tabla 10. Modelado de la historia de usuario B.4

Componente	**Entrada**	**Salida**
ASR	Voz: Ayer no puede acudir de 15 a 18 debido a que tuve una prueba oficial de inglés.	Texto: Ayer no puede acudir de 15 a 18 debido a que tuve una prueba oficial de inglés.
NLU	Texto: Ayer no puede acudir de 15 a 18 debido a que tuve una prueba oficial de inglés.	Intención: justificacionAusencia. Slots: Fecha:11/08/2022 Desde: 15:00 Hasta: 16:00 Motivo: Prueba Oficial Documento: Fichero con la justificación.
DM	Intención: justificacionAusencia. Slots: Fecha:11/08/2022 Desde: 15:00 Hasta: 16:00 Motivo: Prueba Oficial Documento: Fichero con la justificación	Envió del fichero y los datos al ERP, este confirmara la recepción. Mensaje de confirmación de recepción o problemas en la petición.
TTS	Texto Envió del fichero y los datos al ERP, este confirmara la recepción. Mensaje de confirmación de recepción o problemas en la petición.	Voz envió del fichero y los datos al ERP, este confirmara la recepción. Mensaje de confirmación de recepción o problemas en la petición.

4.2.2.10. Historia de usuario B.5

En este caso nuestro usuario ha realizado un viaje de negocios, tiene que enviar de gastos originados al departamento de contabilidad.

Tabla 11. Modelado de la historia de usuario B.5

Componente	Entrada	Salida
ASR	Voz: La semana pasada estuve de viaje comercial en Turquía y quiero enviar los justificantes de los gastos de mi tarjeta.	Texto: La semana pasada estuve de viaje comercial en Turquía y quiero enviar los justificantes de los gastos de mi tarjeta.
NLU	Texto: La semana pasada estuve de viaje comercial en Turquía y quiero enviar los justificantes de los gastos de mi tarjeta.	Intención: justificacionGastos. Slots: Fecha:11/08/2022 Desde: 01/08/2022 Hasta: 01/08/2022 Motivo: Viaje comercial a Turquía Documento: Fichero con la justificación De Gastos.
DM	Intención: justificacionGastos. Slots: Fecha:11/08/2022 Desde: 01/08/2022 Hasta: 01/08/2022 Motivo: Viaje comercial a Turquía Documento: Fichero con la justificación de Gastos.	Confirmación de la recepción correcta o información de algún problema asociado.
TTS	Texto: Confirmación de la recepción correcta o información de algún problema asociado.	Voz: Confirmación del envió de las nóminas con la nómina o de que no ha sido posible la operación.

4.2.2.11.Historia de usuario C.1

Tabla 12. Modelado de la historia de usuario C.1

Nuestro usuario es el encargado en acudir a la oficina en caso de que suene la alarma del edificio, nuestro usuario recibirá una llamada de nuestro producto notificándole la alerta y si existe algún problema en acudir al edificio.

Componente	Entrada	Salida
DM	La función Lambda recibe un evento por parte del sistema de alarmas.	Envía evento al NLU que se ha producido una alarma y el tipo de alarma.
NLU	Recibe evento que se ha producido una alarma y el tipo de alarma.	Intención: alarmaOficina. Slots: Fecha: 9/12/2022 Hora: 11:20 Tipo: Incendio
DM	Intención: alarmaOficina. Slots: Fecha: 9/12/2022 Hora: 11:20 Tipo: Incendio	Realiza la búsqueda en el ERP para indicar que persona está de guardia y obtener su teléfono. Genera texto para enviar al TTS.

TTS	Texto informativo al usuario y petición de confirmación de que va a acudir a la oficina.	Voz informativa al usuario y petición de confirmación de que va a acudir a la oficina.
ASR	Voz: Confirmo que voy a la oficina lo más rápido posible.	Texto: Confirmo que voy a la oficina lo más rápido posible.
NLU	Texto: Confirmo que voy a la oficina lo más rápido posible.	Intención: confirmacionAlarma. Slots: Fecha: 9/12/2022 Hora: 11:27 Tipo: Positivo .
DM	Intención: confirmacionAlarma. Slots: Fecha: 9/12/2022 Hora: 11:27 Tipo: Positivo	Enviamos mensaje al sistema de alarma que va a ir el usuario y generamos texto para el usuario.
TTS	Texto de confirmación que estamos esperando su presencia en la oficina.	Voz de Confirmación que estamos esperando su presencia en la oficina.

Respecto a los casos anteriores podemos apreciar que en el tipo C hacemos más iteraciones con los componentes del sistema, esto se debe a dos cosas. La primera es que la comunicación la inicia un sistema externo (gestor de alarmas) no el usuario y la segunda es que se requiere que el usuario exprese que confirma que va a acudir a la oficina.

4.2.2.12.Historia de usuario C.2

Nuestro usuario es un técnico de sistemas que está de guardia, en caso de algún problema crítico el sistema le enviará una llamada para que se ponga con el incidente.

Tabla 13. Modelado de la historia de usuario C.2

Componente	Entrada	Salida
DM	La función Lambda recibe un evento por parte del sistema de monitorización de infraestructura.	Envía evento al NLU que se ha producido una alerta en sistema de infraestructura y el tipo de alarma.
NLU	Recibe evento que se ha producido una alerta en sistema de infraestructura y el tipo de alarma.	Intención: alertaInfraestructura Slots: Fecha: 9/12/2022 Hora: 11:20 Tipo: serverDown
DM	Intención: AlertaInfraestructura Slots: Fecha: 9/12/2022 Hora: 11:20 Tipo: serverDown	Realiza la búsqueda en el ERP para indicar que persona está de guardia y obtener su teléfono.

TTS	Texto informativo al usuario y petición de confirmación de que va a acudir a la oficina.	Voz informativa al usuario de que hay una alerta de tipo *serverDown* y petición de que se conecte para verificar el problema.
ASR	Voz: Confirmo que he recibido la alerta y que me conecto.	Texto: Confirmo que he recibido la alerta y que me conecto.
NLU	Texto: Confirmo que he recibido la alerta y que me conecto.	Intención: confirmacionAlerta. Slots: Fecha: 9/12/2022 Hora: 11:27 Tipo: Positivo
DM	Intención: confirmacionAlerta. Slots: Fecha: 9/12/2022 Hora: 11:27 Tipo: Positivo	Enviamos mensaje al sistema de monitorización que un técnico va a gestionar la alerta y generamos texto para el usuario.
TTS	Texto de Confirmación que estamos a la espera de que solucione la alerta.	Voz de Confirmación que estamos a la espera de que solucione la alerta.

4.2.3. Infraestructura

En este punto vamos a exponer todos los elementos necesarios para poner en marcha este proyecto. Como hemos indicado la infraestructura elegida es AWS, iremos describiendo el despliegue y contenidos para poner en producción este proyecto. En cada punto añadiremos un enlace con el código fuente y pantallazos necesarios o ilustrativos para el correcto funcionamiento de la infraestructura.

4.2.3.1. AWS Lex V2.

Como hemos analizado en capítulos anteriores dentro del servicio tenemos integrados de forma nativa los servicios de Polly a nivel de TTS y de Transcribe a nivel de ASR. El primer paso es crear el BOT, en nuestro caso le vamos a poner el nombre TuEmpresaBot que es el nombre comercial del Asistente virtual para la gestión de información de empresas.

Uno de los parámetros más importantes a la hora crear la herramienta es el tiempo de espera de sesión activa, por defecto son 5 minutos y es el tiempo máximo que usuario puede estar inactivo, a partir de ese tiempo el sistema borrará toda la información y cerrará la sesión. Este parámetro puede configurarse entre 1 a 1440 minutos. También hay que destacar que en este paso disponemos de la elección de la voz de nuestro Bot, en nuestro caso hemos elegido “Lucia” debido a que es la única en el idioma español-España que está basada en redes

neuronales, el resto están basados en sistemas expertos que como hemos visto en el punto 2 del TFG el rendimiento es bastante inferior. En la siguiente imagen podemos apreciar un pantallazo de la configuración indicando que la voz Lucia es de tipo neuronal.

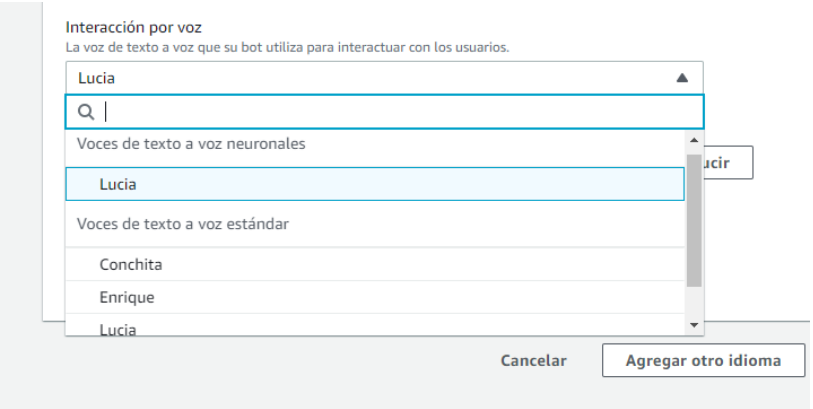

Figura 26. Creación de un bot en AWS Lex (AWS Lex)

Una vez creado el bot en AWS Lex, nos aparece un mensaje de creación y dos intenciones por defecto, una primera denominada *NewIntent* para crear nuestra primera intención y otra llamada *FallbackIntent* que servirá como sumidero en caso de que el sistema no haya encontrado ninguna intención asociada. En la figura 27 podemos apreciar ambas intenciones nada más crear el nuevo bot.

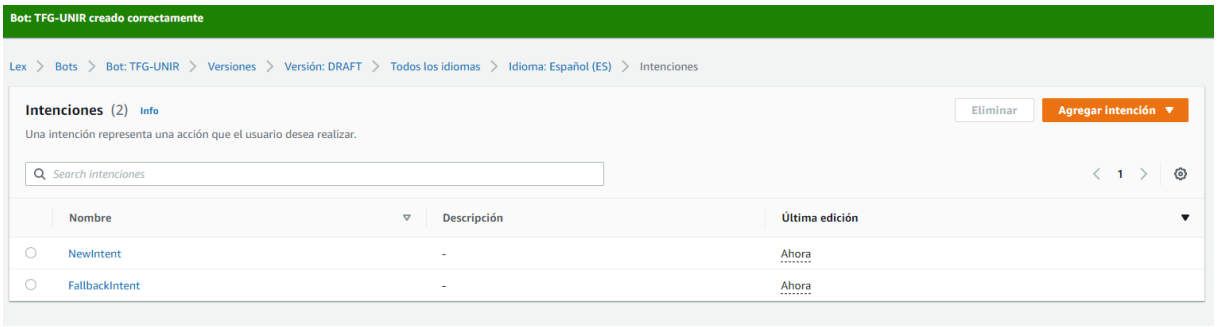

Figura 27. Intenciones por defecto (AWS Lex)

Lo primero que vamos a hacer es crear una intención de bienvenida, cuando el usuario se ponga en contacto con nosotros el sistema le dará un mensaje de bienvenida indicándole que acción desea realizar, en la siguiente imagen podemos un cajetín de prueba de chat que dispone Amazon Lex. En el podemos apreciar por un lado el mensaje de bienvenida y por otro

que el sistema dispone de un sistema de prueba tanto de texto como de voz, este último se activa en el micrófono de la parte inferior izquierda.

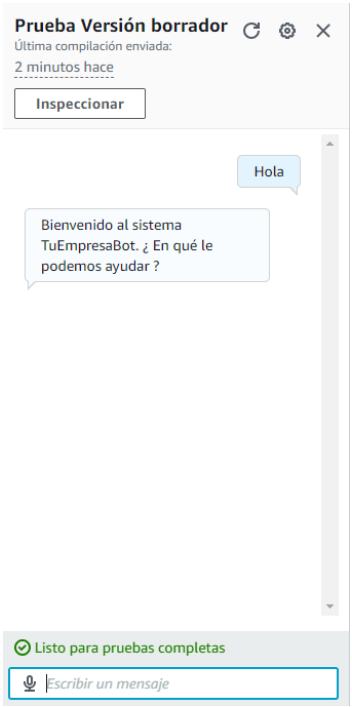

Figura 28. Cajetín de pruebas chat con compatibilidad de voz (AWS Lex)

Además de la intención de bienvenida y las intenciones predeterminadas por la herramienta Amazon Lex V2. Nuestro sistema va a poder realizar 12 historias de usuario que hemos modelado en el punto anterior, a continuación, vamos a exponer la configuración de estas intenciones y slots.

Como ejemplo vamos a analizar la historia de usuario A.1 que es una petición de vacaciones, en esta vamos a desarrollar cómo funcionan los slots o ranuras. Recordemos que estas son variables que vamos a necesitar para la correcta consecución de la intención, en esta denominada “vacaciones” vamos a necesitar la fecha inicial y la fecha final. A continuación, vamos a ver las dos formas de configurar la captación de estos datos en los siguientes párrafos.

Mediante los enunciados de muestra: Con esta acción podemos especificar en el texto del enunciado de muestra, el nombre de la ranura que queremos capturar. Por ejemplo “Quisiera vacaciones del {desde} al {hasta}”, estaríamos capturando el slot “desde” y “hasta “. En el

siguiente pantallazo podemos ver los enunciados de ejemplo, hemos configurado 10 como indica el fabricante hay que destacar que se pueden configurar enunciados con o sin slots.

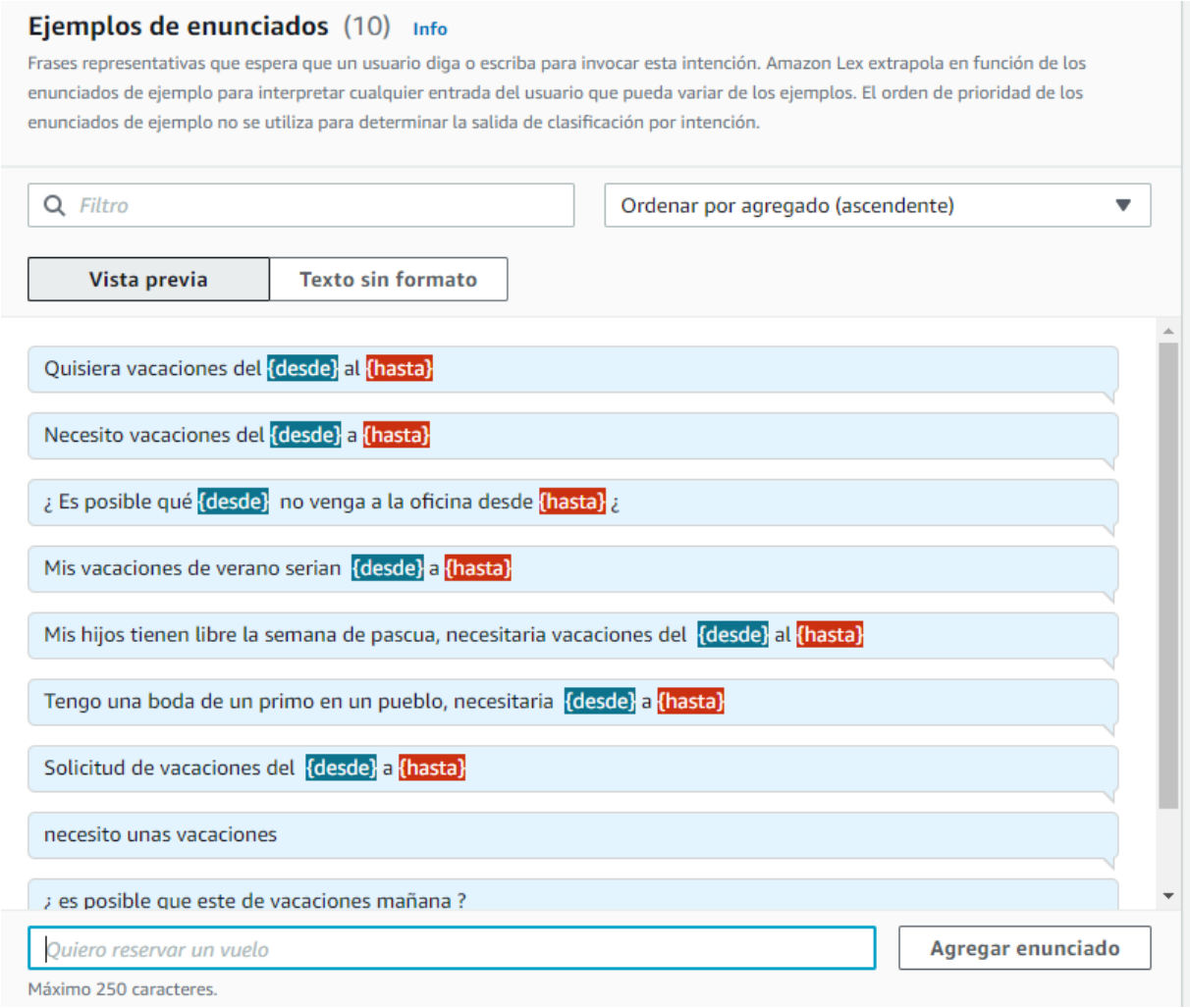

Figura 29. Enunciados configurados de la historia usuario A.1 (AWS Lex)

Mediante preguntas de captación. En este caso en la configuración del slot podemos indicar que mensaje de captación vamos a utilizar en caso de no encontrar la ranura en el enunciado con el que se ha comunicado el usuario. En la figura 30 podemos apreciar cómo se configura, en el apartado solicitudes, hacemos la pregunta: "¿Cuál es la fecha de inicio de sus vacaciones?"

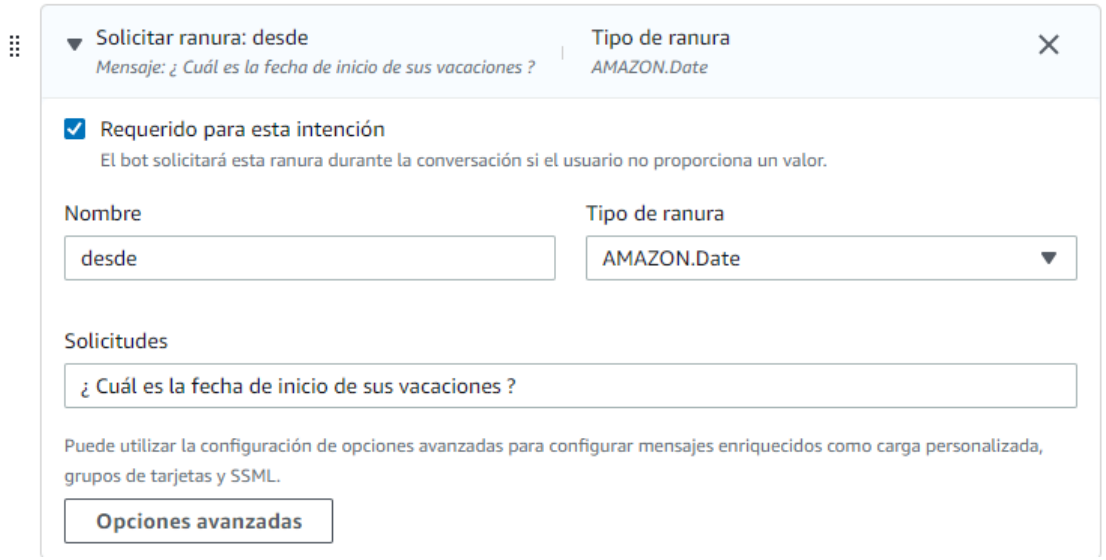

Figura 30. Configuración por preguntas de captación (AWS Lex)

En la imagen anterior también hay que destacar que se puede configurar otros parámetros de la ranura como el tipo, en este caso es tipo AMAZON.Date al ser una fecha y que nos permite configurar el texto con formato SSML para la reproducción enriquecida del mensaje.

A continuación, vamos a mostrar dos conversaciones, la primera donde la captación de las ranuras se realiza mediante la frase del usuario mientras que, en el segundo caso, el usuario al omitir esta información el servicio le ha preguntado explícitamente el inicio y final de vacaciones. También hay que destacar que las frases del usuario no han sido iguales que las configuradas en el enunciado de muestra y pese a eso el sistema ha detectado la intención, eso se debe a que el sistema se basa en entrenamiento no en un sistema de reglas como hemos indicado en el marco teórico de este TFG.

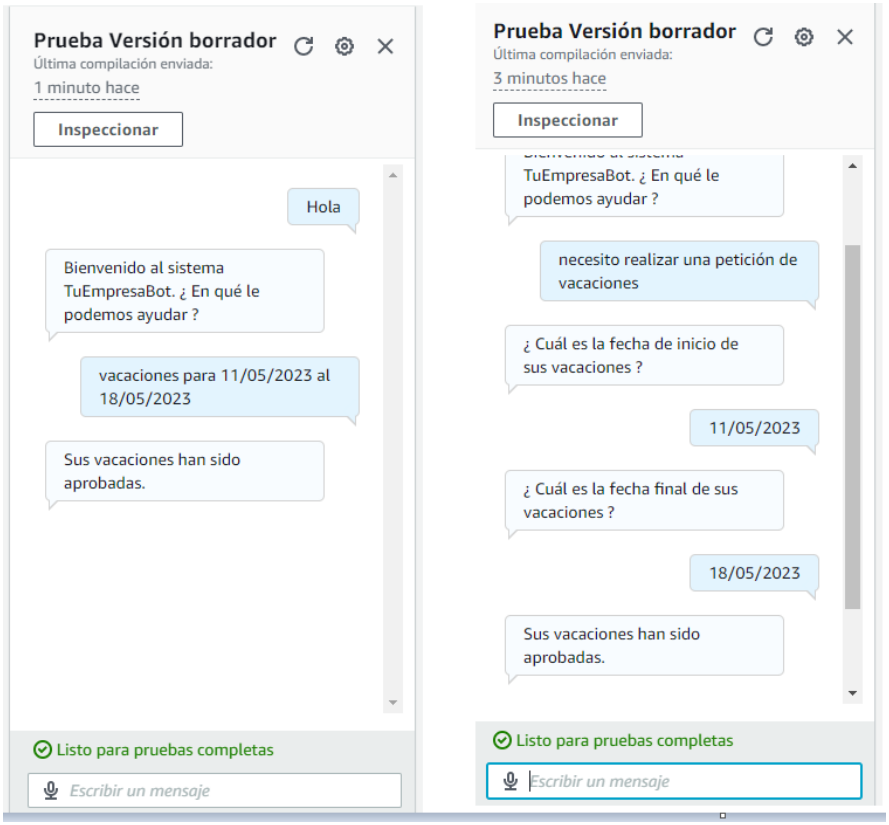

Figura 31. Conversaciones con captación mediante enunciado y por preguntas de captación (AWS Lex)

En las acciones tipo C existen dos intenciones en vez de una, eso es debido a que la acción es generada por el DM, no por el usuario como en las 10 primeras. Como ejemplo en el siguiente pantallazo podemos apreciar esta circunstancia, la acción C1 tiene dos intenciones configuradas que son alarmaOficina y confirmacionAlarma, la acción C2 también tiene dos intenciones configuradas que son alertaInfraestructura y confirmacionAlerta. El resto de las acciones que son las de tipo A y B sólo requieren de una intención. En la figura 32 podemos observar lo que hemos descrito.

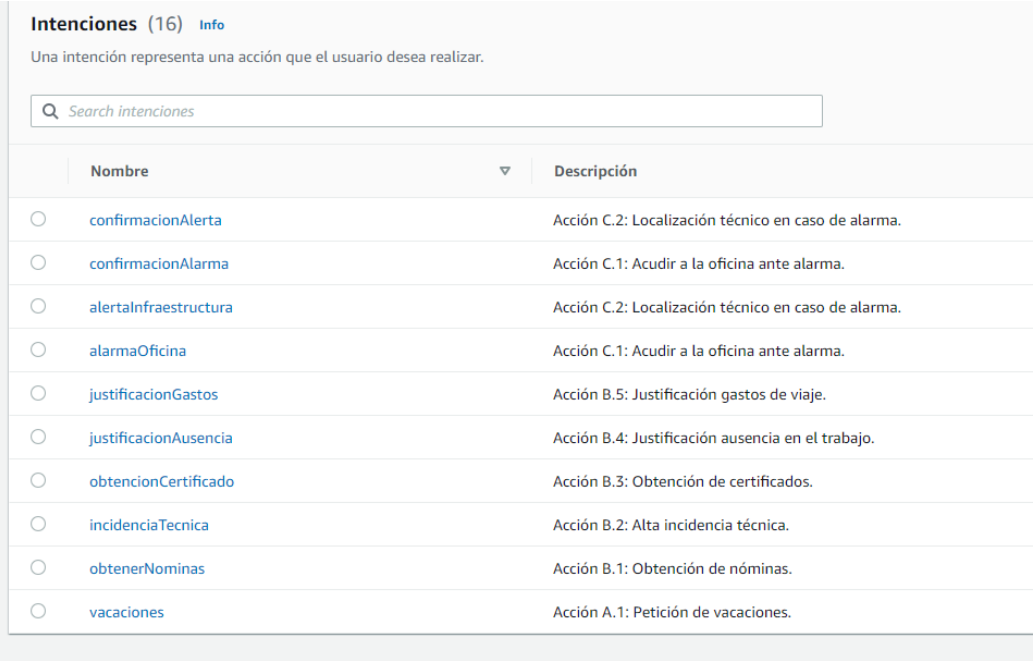

Figura 32. Intenciones configuradas en el servicio (AWS Lex)

Por último, hay que indicar que en el siguiente enlace podemos descargar el código generado y configuración de esta parte del proyecto:

https://github.com/serapde/TFG-UNIR

4.2.3.2. Funciones Lambda

Las funciones Lambda se utilizan como DM en nuestro proyecto. Estas aceptan bastantes lenguajes de programación como podemos apreciar en la siguiente imagen, para nuestro proyecto vamos a utilizar Python 3.8.

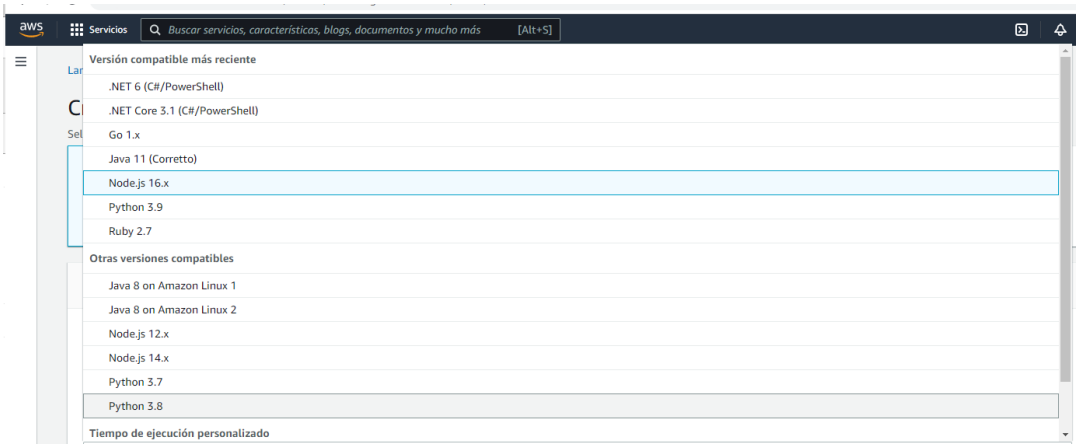

Figura 33. Lenguajes de programación compatibles con funciones Lambda (AWS)

Además de apuntar a versiones, los alias nos sirven para apuntar a las funciones Lambda, concretamente dentro del alias se pueden configurar los idiomas y cada uno de estos puede apuntar a una función Lambda. En nuestro TFG la hemos llamado “botempresa” y hemos indicado que siempre apunten a la última versión, en la siguiente imagen podemos apreciar cómo se configura a nivel de AWS.

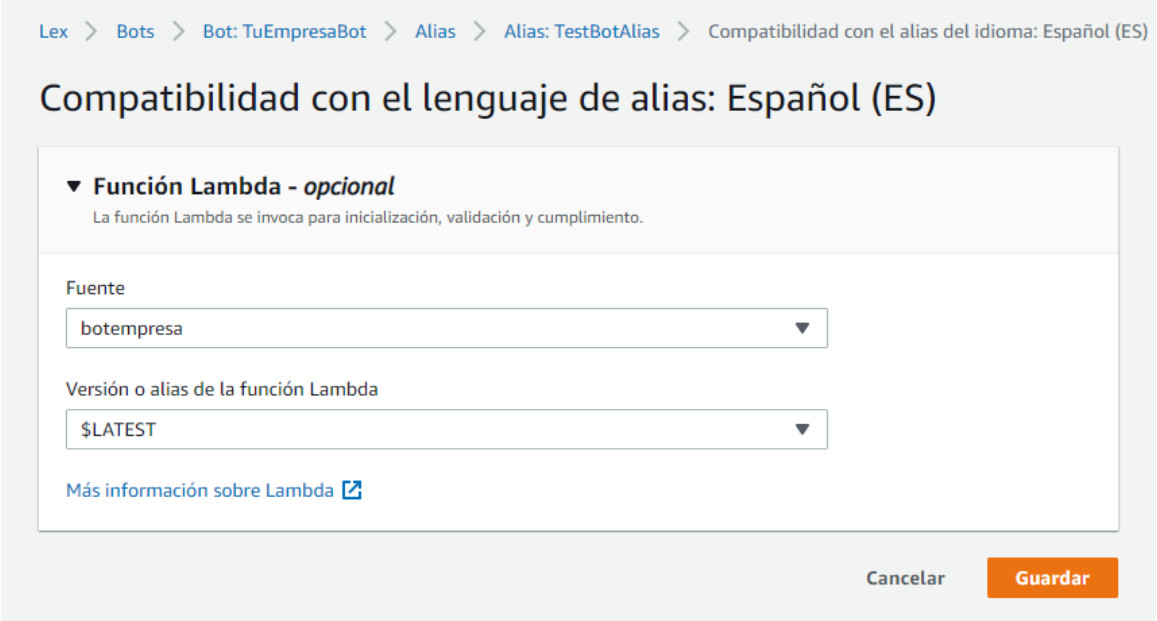

Figura 34. Configuración del alias del bot para que apunte a la función Lambda (AWS Lex)

En cada intención hay que indicar que se necesita una función Lambda en el cumplimiento, así antes de finalizar la intención invocará a la función Lambda. Una vez dentro de la función se invocará el método *lambda_handler* y este obtendrá la respuesta que debe enviar a Lex de un método denominado *dispatch* donde esta codificadas todas las intenciones y la lógica de lo que se debe hacer en cada caso. En la siguiente figura podemos ver un fragmento del código que hemos comentado.

Figura 35. Fragmento de código del proyecto gestión de eventos desde el NLU (Realización propia)

Las acciones tipo C tenían la peculiaridad que el origen de la conversación era la recepción de un evento externo en el DM, para realizar esta función hemos creado otro fichero denominado lambda_events.py, el método *lambda_handler* llamará al método *router* que será el encargado de enrutar al NLU que intención debe de realizar. En la siguiente captura de pantalla podemos apreciar un fragmento de código de esta función.

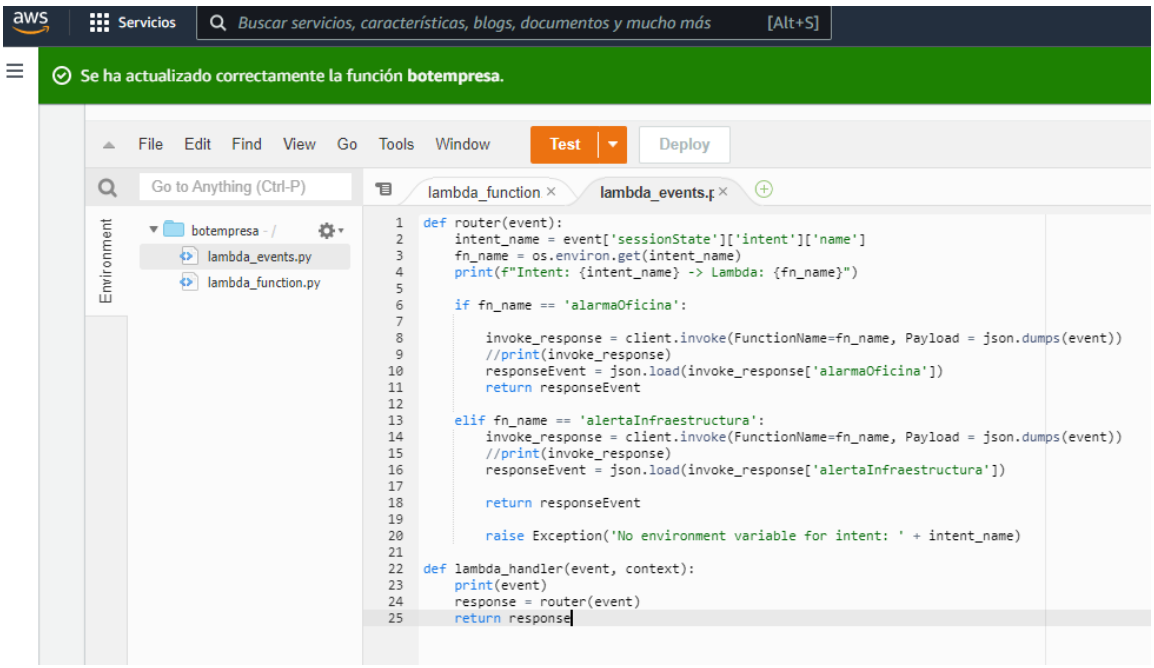

Figura 36. Fragmento de código del proyecto gestión de eventos hacia el NLU (Realización propia)

Por último, hay que indicar que en el siguiente enlace podemos descargar el código generado en esta parte del proyecto: https://github.com/serapde/TFG-UNIR

4.2.3.3. Lex-Web-UI.

Como interfaz de usuario vamos a utilizar Lex-Web-UI, es un proyecto open Source y nos permite una rápida integración en una web. Una vez integrado vamos a poder interactuar tanto con voz como en texto. Las principales características son:

- La interfaz es responsive por lo que se puede adaptar tanto a ordenadores, tablets y móviles.
- Detección de silencios y capacidad de iniciar conversación con el usuario.
- Integración nativa con Amazon Lex.
- Customización mediante JavaScript y CSS.

En el siguiente enlace podemos descargar el código fuente: https://github.com/aws-samples/aws-lex-web-ui

4.2.3.4. CloudFormation.

Esta herramienta nos va a permitir crear una colección de servicios de AWS relacionados. En nuestro caso es una herramienta muy útil para el despliegue de nuestro proyecto, debido a que vamos a poder crear, actualizar y eliminar una pila completa como si fuera una única instancia. Además de los servicios que necesitamos para el proyecto en la plantilla podemos añadir todas las configuraciones de intenciones y ranuras de AWS Lex, de hecho, lo que añadimos todo lo que apunta el alias producción. En la siguiente imagen del entorno de CloudFormation podemos apreciar la pila creada.

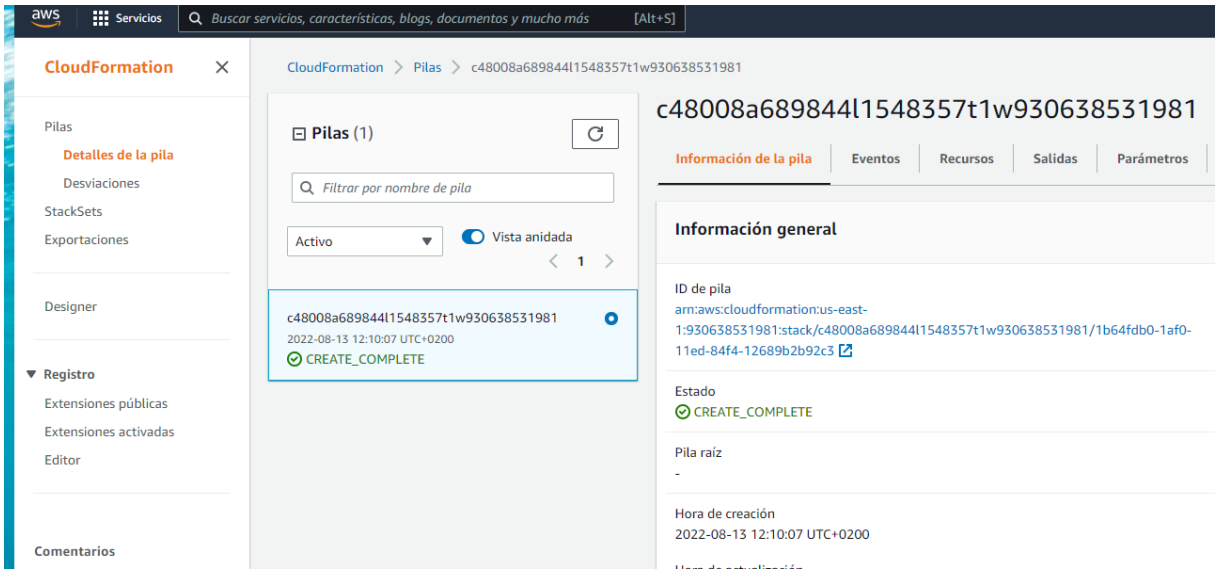

Figura 37. Pantallazo creación PILA (AWS CloudFormation)

4.3. Metodología

En este trabajo vamos a utilizar metodologías agiles. El objetivo va a ser conseguir flexibilidad y rapidez a la hora de acometer posibles cambios que nos encontremos mientras que realizamos nuestra herramienta. Lo primero que vamos a realizar en este apartado es definir los recursos técnicos y materiales. Lo siguiente es la metodología para la generación de versiones y por último a partir de las historias de usuario definidas en el apartado anterior vamos a definir las tareas que debe de tener cada rol para llegar a la consecución de este proyecto.

4.3.1. Recursos personales y materiales

Este es un proyecto multidisciplinar donde vamos a tener varios roles para poder tener éxito en el proyecto. Es muy importante recalcar que nuestro usuario siempre va a demandar los siguientes puntos como hemos visto en distintas partes de este TFG:

- Utilización del lenguaje natural.
- Comprensión de la acción que ha indicado.
- Resolución de la interacción lo más rápida posible.

A partir de estas premisas tenemos que realizar una infraestructura de recursos humanos y técnicos que pueda atender estas premisas del usuario. A continuación, vamos a exponer los roles necesarios y sus principales funciones.

4.3.1.1. Diseñador de conversaciones.

- Diseñar el flujo de conversaciones con el usuario final.
- Configurar intenciones y ranuras.
- Verificar que cada interacción es adecuada con la cultura de la empresa.

4.3.1.2. Desarrollador.

- Funciones Lambda.
- Scripts de monitorización.
- Webchat.

4.3.1.3. Infraestructura.

- Gestión de recursos de Amazon.
- Monitorización de infraestructura.
- Gestión de actualizaciones.

4.3.1.4. Analista de datos.

- Diseño de métricas de desempeño.
- Analizar mejoras en el sistema.
- Reportar errores en el producto.

4.3.1.5. Jefe de proyecto.

- Gestión de costes.
- Coordinación de los distintos roles.
- Comunicación con la empresa.

Para poder acometer este proyecto necesitamos los siguientes recursos materiales dentro del entorno de AWS: Lex v2, Funciones Lambda y CloudFormation. Para la puesta en marcha del TFG hemos dado de alta una cuenta educativa con $ 100 de crédito para poder desplegar la

solución. En la siguiente imagen podemos apreciar en la parte superior derecha que hemos utilizado $ 98,4 para ponen en marcha la solución.

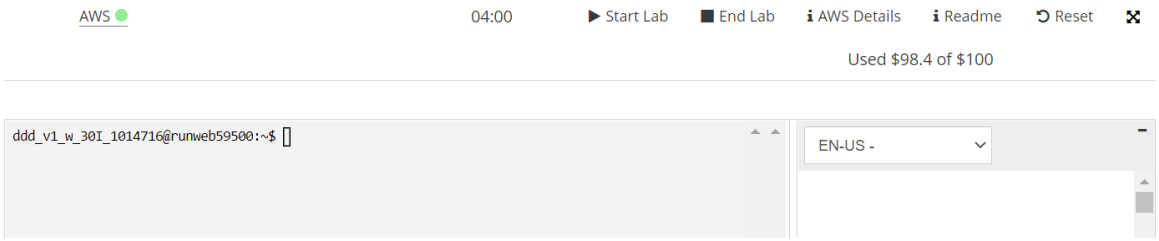

Figura 38. Pantallazo con el costo que ha tenido el TFG en AWS (Servicio educativo AWS)

4.3.2. Control de versiones.

Como hemos visto durante este TFG estos sistemas cognitivos están constantemente cambiando, esto se debe en gran medida por que están destinados para dar servicio interactivo al ser humano. Esto hace que debamos estar liberando versiones con mejoras con lo cual debemos de tener un control del producto con la posibilidad de volver atrás de la forma más fácil posible. El punto crítico de este TFG es el servicio Amazon Lex, debido a que la adición de nuevos enunciados, intenciones o slot podrían tener el riesgo de bajar el desempeño del asistente.

Los bots generados por Amazon Lex permiten la creación de alias, que es un puntero a una versión del bot. Una versión es una imagen funcional del aplicativo, tanto a nivel de configuración y código. En la siguiente imagen podemos apreciar que existen dos alias uno denominado “Prod alias” que apunta a la versión 1 y otro “Beta alias” que apunta a la versión 2, por otro lado, se puede apreciar que el aplicativo (Cliente application) apunta al alias “Prod alias”.

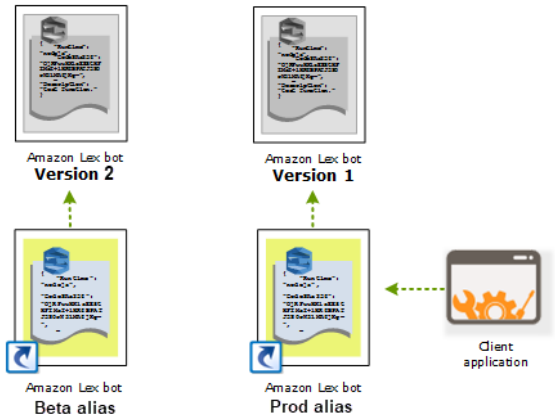

Figura 39. Esquema funcionamiento alias y versiones (AWS Lex)

Gracias a este sistema podemos realizar todas las pruebas contra el alias Beta y una vez que hayan sido satisfactorias cambiar el punto de "Prod alias" para que apunte a versión 2, con esto conseguimos un desacople con otras partes del producto como el Webchat y mayor seguridad ya que en caso de problemas en producción con la nueva versión volver hacia atrás es tan simple como volver a cambiar los punteros.

4.3.3. Tareas para realizar.

En este proyecto tenemos 5 perfiles, el jefe de proyecto como ya hemos indicado en el punto anterior entre otras hace labores de coordinación entre las áreas. Hay unas áreas que tienen tareas iterativas hasta que no se terminen no pueden empezar la siguiente, por otro lado, también hay tareas que dependen de una tarea de otra área, en el diagrama que hemos realizado esto se representa con una flecha. Existen tareas que se deben de realizar en conjunto entre todos los perfiles, por ejemplo, las pruebas ya que las mejoras y recopilación de fallos pueden provenir tanto de la parte de infraestructura, conversaciones, monitorización y desarrolladores. En el siguiente diagrama podemos apreciar cómo se va a realizar el despliegue de la solución. En la parte superior la tarea es "Alta AWS" y la realizará la parte de sistemas siendo la primera a realizar, después podemos ir observando la distribución de tareas hasta llegar el objetivo final que es tener una versión del producto, nodo inferior en el diagrama.

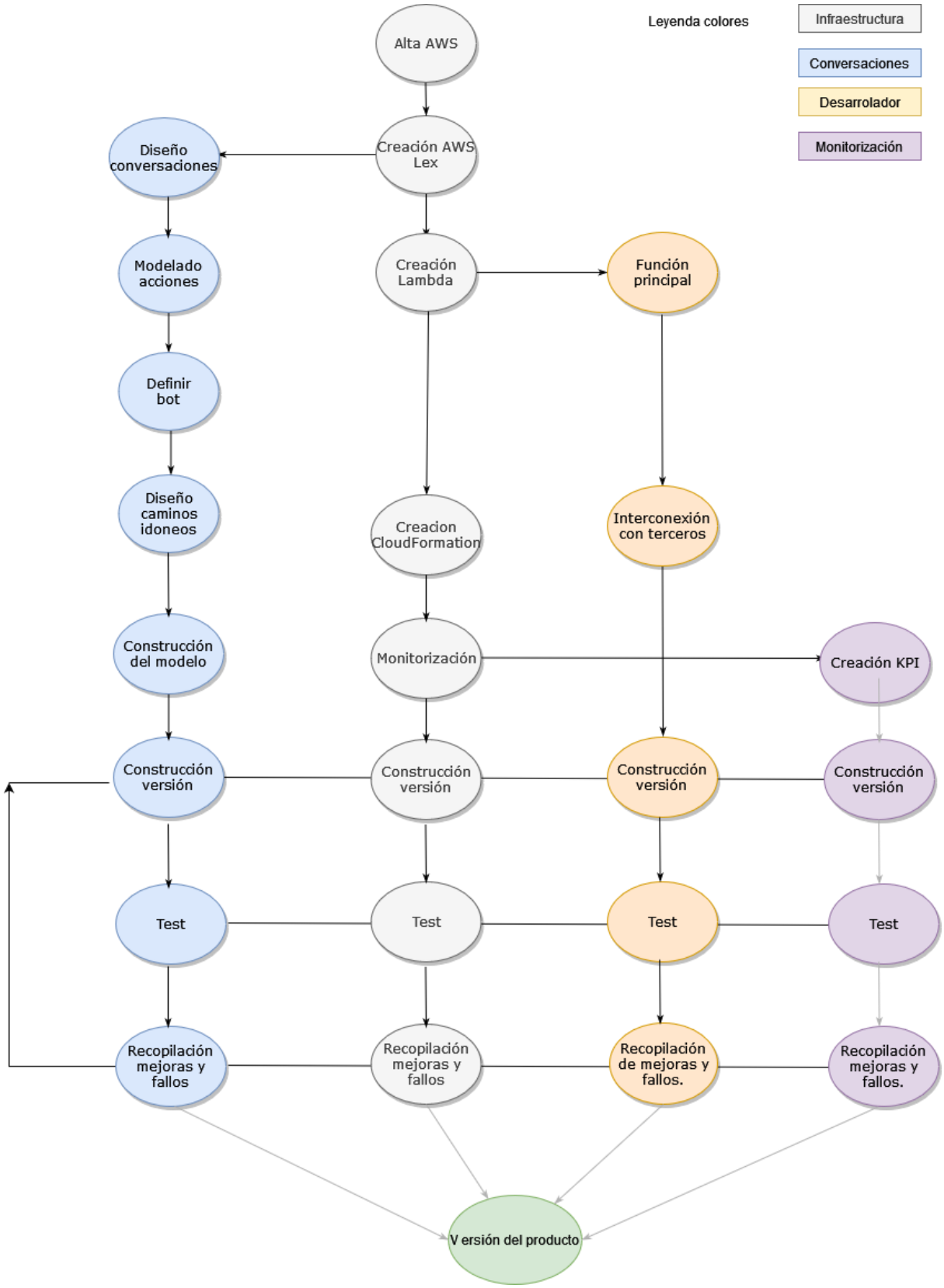

Figura 40. Diagrama despliegue de la solución (Realización propia)

4.3.3.1. Infraestructura.

La primera tarea del proyecto recae en este perfil ya que debe de crear la cuenta de AWS y usuarios en AWS. Después debe de crear el servicio Amazon Lex V2, esta tarea es fundamental ya que una vez terminada los compañeros de diseño pueden empezar a diseñar las conversaciones. Seguidamente crearan los servicios Lambda, la consecución de esta tarea conlleva que los compañeros de desarrollo puedan empezar sus tareas de programación. Acto seguido viene la creación de CloudFormation para tener la interfaz de chat operativa de usuario. Una vez finalizadas estas tareas sólo faltaría finalizar el sistema de monitorización para que los colegas de esta área puedan empezar con sus tareas.

A este perfil le quedaría las tareas en conjunto, construcción de versión, test, recopilación de mejoras y fallos. Donde van a estar muy pendientes en la construcción de alias y versiones, además de aquellos fallos mejoras cuyo origen sean la infraestructura como ejemplo problemas de permisos o rendimiento del aplicativo.

4.3.3.2. Conversaciones.

Este perfil como hemos visto en el apartado anterior está muy ligado a la construcción de las conversaciones del producto. Depende de las partes de infraestructura para la creación del servicio AWS Lex. Por lo que sus tareas van a estar muy enfocadas en la compresión de como hablan los humanos con la IA y la gestión de posibilidades de la conversación. En la primera tarea vamos a diseñar el plan para recopilar datos que vamos a necesitar en la conversación, además de establecer las interfaces conversacionales que vamos a tener con los usuarios. La segunda tarea es fundamental en el proyecto ya que se modelan las historias de usuario una vez terminada configuraremos los parámetros del bot como el tiempo de sesión o el idioma. después empezaremos con la creación de los enunciados y preguntas de captación de ranuras idóneas, donde tanto el usuario como el bot no van a fallar. En la última tarea individual se construirá el modelo en esta tarea ya tendremos todo el diseño de conversaciones preparado para probarlo con las funciones Lambda. En las tareas en común este perfil tiene que estar muy pendiente de que aquellos casos donde no hemos finalizado con éxito la conversación.

4.3.3.3. Desarrollador.

El equipo de Desarrollo tiene como requerimiento principal tener disponible el servicio Lambda (DM). Una vez que tengan esa infraestructura ya pueden diseñan y programar. La primera tarea es construir la función Lambda principal, a través de ellas se recibirán y emitirán mensajes a Lex. También tienen que realizar la infraestructura de interconexión con terceras aplicaciones como el ERP de la empresa o el sistema de Alarmas, esta tarea se realizará mediante funciones Lambda.

Una vez terminadas estás tareas a este perfil le quedará las tareas comunes antes de generar la versión, por lo que tiene que estar muy pendiente a los posibles problemas o mejoras en todo lo referente de la programación, con especial atención que todos los flujos de datos estén correctamente funcionando.

4.3.3.4. Monitorización.

Este perfil está muy unido con el desempeño de la aplicación por lo que la primera tarea que va a tener es definir los KPI específicos de la solución. Por otro lado, tendrá que validar si la versión es acorde a los umbrales mínimos fijados en la primera tarea. Si todos los índices de desempeño están por arriba se podrá liberar la versión, en caso de que estén por debajo significaría que se debe de volver a iterar hasta que se consiga el hito de que todos los umbrales estén por encima.

4.4. Evaluación

La evaluación de este proyecto este muy ligado al rol de monitorización, a continuación, vamos a definir los KPI y los umbrales de funcionamiento, asimismo insertaremos pantallazos a modo ilustrativo de configuración y datos en producción.

- Número de peticiones: Al ser una aplicación basada en pago por uso la monitorización de cada petición es muy importante, esto es debido a que podemos tener un incremento de costos por fallos en la programación o ataques externos. En la siguiente imagen podemos ver la gráfica que proporciona Amazon de este Kpi. El umbral de funcionamiento correcto del aplicativo que hemos configurado es 1.200 peticiones por

minuto, todo lo que esté por encima de este parámetro significara que hay algo anómalo en el aplicativo.

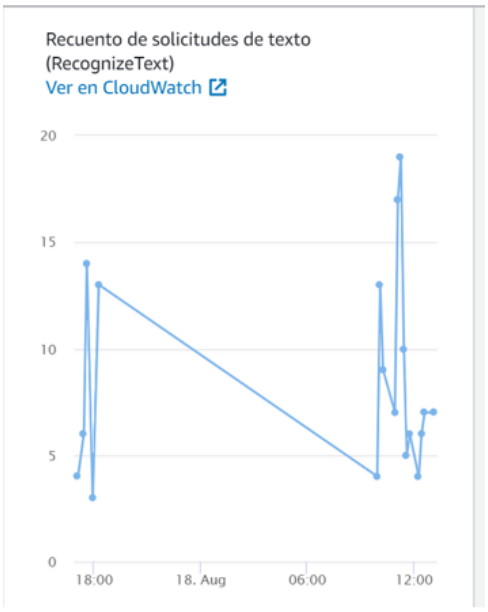

Figura 41. Número de solicitudes de texto (AWS Lex)

- Latencia de las peticiones: Como hemos visto a lo largo de este trabajo la baja latencia es crucial en este tipo de proyectos. El principal motivo es que es un sistema en tiempo real, por lo que cualquier incremento de tiempo en la respuesta del usuario podría hacer que el usuario abandonará la conversación. El umbral de desempeño que hemos configurado es 300 ms, donde el 99 % de las peticiones del último minuto tiene que está por debajo de este umbral. En la siguiente gráfica podemos apreciar datos en real con un caso donde llegamos a superar este umbral, el retardo medio llega a los 800 ms.

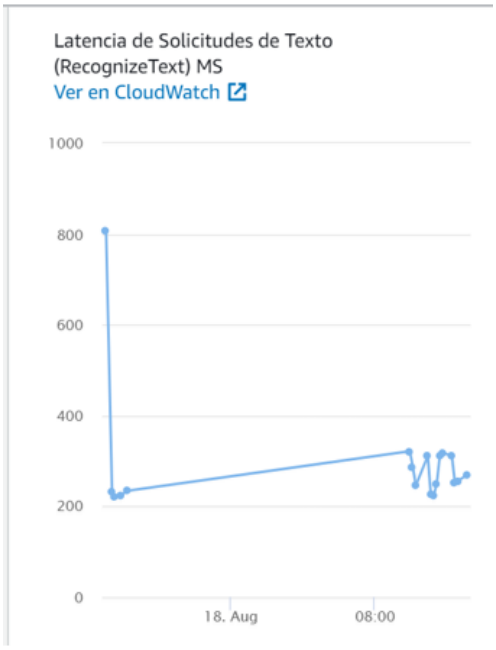

Figura 41. Latencia de solicitudes de texto (AWS Lex)

- Latencia de las funciones Lambda: Este desempeño nos dará más profundidad de donde proviene el retardo. El umbral de desempeño que hemos configurado es 300 ms, donde el 99 % de las peticiones del último minuto tiene que está por debajo de este umbral, en la siguiente gráfica podemos ver el comportamiento de las dos funciones Lambda asociadas a este proyecto, la primera es LightsailMonitoringFunction es propia al sistema de AWS, sirve para recoger y almacenar datos de la monitorización del sistema y la segundo es Botempresa donde están los ficheros Python para poder ejecutar la lógica del DM.

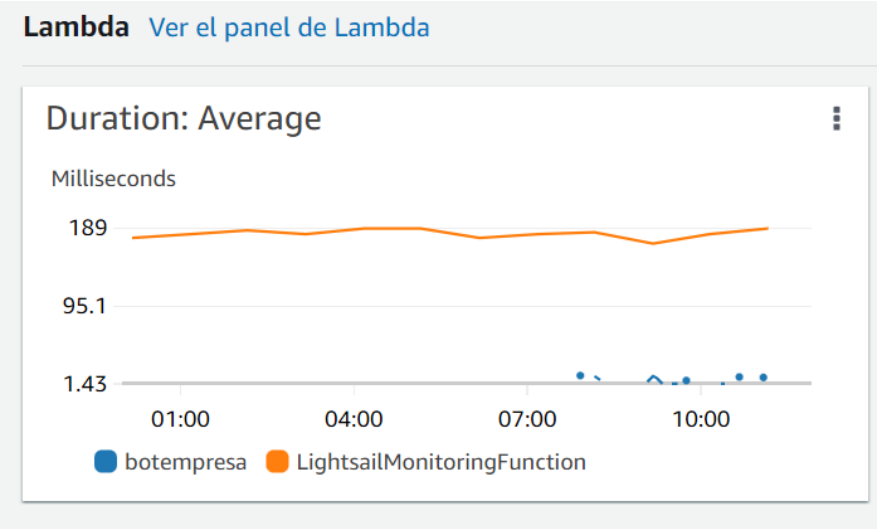

Figura 43. Latencia de ejecución funciones Lambda (AWS Lex)

- Intenciones perdidas: La visibilidad de este dato es muy importante ya que nos va a indicar aquellas conversaciones que se pierden cuando la intención ya está iniciada. El índice de desempeño que hemos configurado es del 95 %, por lo que de cada 100 condiciones que lleguen a una intención más de 95 tienen que terminar con éxito. El sistema te permite exportar a Excel aquellos casos donde la intención ha sido fallida. En el siguiente pantallazo podemos ver un fragmento del Excel que recopila las intenciones fallidas, en la columna G nos indica el punto donde el usuario abandono la conversión.

	A	B	C	D	E	F	G
1	Bot	Fecha	horas	Alias	Intención	Id	Punto
2	TuEmpresaBot	22/08/2022	18:40:11	TestBotAlias	vacaciones	54792517	Captura ranura desde
3	TuEmpresaBot	22/08/2022	18:43:16	TestBotAlias	vacaciones	54792518	Captura ranura hasta
4	TuEmpresaBot	22/08/2022	18:46:38	TestBotAlias	justificacior	54792519	Captura ranura motivoGastos

Figura 44. Ejemplo de intenciones perdidas (AWS Lex)

Independiente del umbral todos los casos perdidos se tienen que analizar y poner medidas correctivas para que en la siguiente interacción sean satisfactorios.

- Enunciados perdidos: En este caso nos dirá aquellos enunciados que ha realizado el usuario y que no hemos podido clasificar en ninguna intención. El índice de desempeño

que hemos configurado es del 90 %, eso significa que el sistema debe de tener más de un 90 % de éxito, todo lo que sea perdido se tendrá que analizar y poner medidas correctivas. En el siguiente pantallazo podemos apreciar cómo se saca ese informe, Amazon te permite segmentar entre enunciados detectados o perdidos, asimismo es posible tener la información por versión o alias.

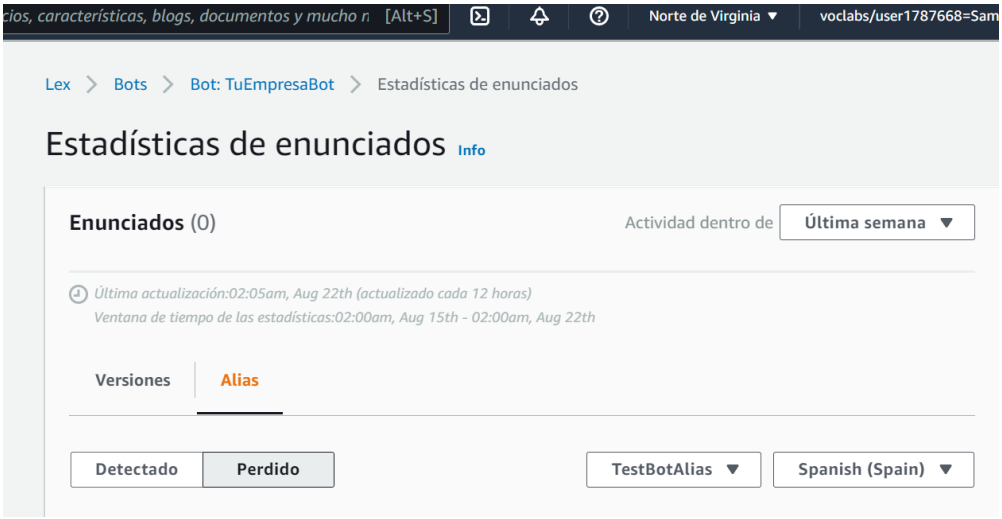

Figura 45. Estadísticas de enunciados (AWS Lex)

5. Conclusiones y trabajo futuro

5.1. Conclusiones

A mitad del siglo pasado empezó una línea de investigación de creación de sistemas que fueran capaces de realizar tareas que sólo saben hacer los humanos. En los orígenes estos sistemas se denominaban “expertos” ya que eran capaces de realizar tareas propias de experto en la materia, los principales problemas eran que estaban basados en reglas por lo que el límite de conocimiento dependía del conjunto de ordenes codificados. A finales del siglo pasado empezó el auge de los sistemas cognitivos, estos están basados en Redes Neuronales Artificiales y su principal característica es que son capaces de aprender mediante un conjunto de datos de entrada, cuantos más datos posea el sistema la precisión será mejor. Esta característica ha hecho que los sistemas cognitivos sean más eficientes que los sistemas expertos, ya que estos últimos llegado a un umbral de datos de entrada su desempeño se

manteniente constante. En contrapartida los sistemas cognitivos requieren de un gran consumo de recursos de infraestructura, tanto a nivel de GPU, memoria y disco duro. En la última década la computación en la nube y el concepto de pago por uso ha hecho que estos sistemas proliferen ya que el despliegue de hardware necesario recae en el proveedor.

Las empresas tienden a generar innumerables datos y procedimientos, por otro lado, tienden a estar departamentadas generándose lo que se denomina silos de información, esto hace que dos trabajadores de una misma empresa puedan realizar procedimientos distintos y ninguno de estos sea el idóneo por la compañía. También existen situaciones cotidianas que requieren de mucho tiempo por parte del parte trabajador, esto hace que la productividad caía ya estas situaciones suelen depender de terceras personas que no pueden estar 100 % disponibles o son procedimientos arcaicos y en farragosos.

Debido al auge de los sistemas cognitivos los asistentes virtuales en el hogar se han hecho cotidianos en el día a día. Esto se debe a que son capaces de interactuar en tiempo real y son comprensibles por el usuario. Otra de las características es que se pueden comunicar en los idiomas más utilizados en el planeta, por otro lado, las fabricantes de estos asistentes virtuales posean infraestructura Cloud a lo largo del mundo, por lo que ha hecho que la difusión a lo largo del planeta haya sido exponencial.

Por norma general en las empresas no se ha extendido el uso de estos asistentes virtuales, por lo que entendemos que un producto como TuEmpresaBot tiene un nicho de mercado ya que puede incrementar sustancialmente la productividad de una empresa, al solventar carencias de comunicación y procedimientos que puede tener cualquier tipo de empresa.

Es relevante como fabricantes de asistentes virtuales en el hogar como Amazon o Google proporcionan acceso a sus servicios de TTS, ASR, NLU o DM. Esto hace que un proyecto características se pueda desplegar sin unos excesivos costos iniciales, ya que la infraestructura Cloud y los servicios necesarios ya entrenados los proporciona el proveedor. Siendo muy novedoso la programación de Funciones Lambda debido a que hemos podido programar la lógica del DM sin necesidad de instalar y configurar un entorno de programación.

A priori se podría pensar que un perfil técnico se podría encargar de la ejecución del proyecto, pero lo novedoso del proyecto es llegar a la conclusión que son multidisciplinares y requieren

que los roles estén totalmente desacoplados y coordinados. Consiguiendo una estructura donde cada perfil tiene sus tareas lógicas asociadas, como ejemplo podemos indicar que no tiene nada que ver el perfil de un diseñador de conversaciones con el de un programador en Python, pero ambos dependen el uno del otro para lograr la consecución del proyecto.

Estos proyectos requieren de una evolución constante debido a que las empresas tienden a estar en constante evolución, el hecho de utilizar metodologías agiles, el pago por uso de los servicios cloud, disponer de perfiles desacoplados y una arquitectura versátil hacen que un producto como TuEmpresaBot pueda evolucionar rápidamente y adaptarse a las necesidades que pueda tener cada empresa.

5.2. Trabajos futuros.

Los asistentes virtuales es un campo novedoso donde los avances son constantes, esto hace que existan varios puntos clave donde hay campo de investigación y desarrollo que a continuación pasamos a desarrollar.

5.2.1. Comparativa de componentes.

Como hemos visto existen servicios como el TTS, ASR, NLU y DM que son un estándar en el sector y pueden estar totalmente desacoplados, en este TFG hemos analizado Amazon y Google, pero existen otros fabricantes y soluciones en el mercado como pueden ser IBM Watson, Microsoft Azure o soluciones Open Source. Un trabajo futuro seria realizar una comparativa a nivel técnico y económico entre los principales actores del sector e implementar la mejor solución posible, teniendo en cuenta que la arquitectura propuesta permite tener por ejemplo un ASR de un fabricante A y un NLU de un fabricante B.

5.2.2. Creación de APP.

En este TFG hemos implementado una solución de Webchat *Open Source* denominado Lex-Web-UI que nos permite publicar tanto a nivel de voz como de texto una solución para navegadores web, otra de las ventajas es que es responsive pudiéndose utilizar tanto en navegadores de ordenadores, tabletas o móviles. Un posible trabajo futuro seria la realización de una app compatible con Android y IOS, para que los usuarios se puedan descargar en su

móvil la aplicación ya sea corporativo o personal y puedan acceder a las funcionalidades sin tener que esta conectados permanentemente a una url mediante un navegador.

5.2.3. Interconexiones con aplicaciones terceras aplicaciones.

En este TFG hemos visto que casi todas las historias de usuario requieren de una interconexión con un aplicativo externo, esto se debe a que la mayoría de las empresas disponen de herramientas específicas para la gestión diaria de la empresa. En nuestro trabajo hemos podido apreciar cómo se necesitan integraciones con ERP, por ejemplo, para información de información de un proveedor, con sistemas de RRHH para la gestión de vacaciones o ausencias, estas herramientas pueden ser un módulo del ERP o ser un software independiente, y por último con sistemas de monitorización de infraestructuras o sistemas de alarmas.

Un campo donde se podría profundizar a partir de este TFG es realizar una integración con las principales herramientas de ERP, RRHH, sistemas de monitorización de infraestructura y gestora de alarmas. La posible línea seria realizar un análisis de las herramientas más utilizadas en el mercado, ver su viabilidad técnica y negocio a la hora de interconexión y finalmente realizar la integración.

5.2.4. Adicción de canales digitales y PBX.

En este trabajo hemos profundizado en la utilización del Webchat, pero en la actualidad existen innumerables canales de comunicación con un usuario. Casi todas las empresas tienen una centralita o PBX, estas sirven para poder recibir, emitir y realizar llamadas entre extensiones. La arquitectura diseñada en este proyecto permite la comunicación por voz, por lo que adicionar una PBX al proyecto puede ser muy beneficioso para el usuario que está muy ligado al mundo empresarial. Por otro lado, desde la pandemia el uso de herramientas colaborativas como Microsoft Teams o Google Workspace han tenido un crecimiento exponencial, por lo que la integración de nuestro producto con los canales de voz o texto de estas herramientas puede ser muy beneficioso para las empresas. Hay que destacar que la arquitectura que hemos diseñado permite la integración con el canal de voz o texto, independientemente del fabricante que haya en el otro extremo.

Por otro lado, el auge de las redes sociales en la última década hace que una posible interconexión con todas estas sea totalmente provechosa para el desarrollo del producto. En la actualidad hay un amplio abanico de redes de sociales como Instagram, LinkedIn, Facebook o TikTok, en este caso habría que realizar un estudio de las ventajas de integrar nuestro producto con una o varias redes sociales y después realizar la integración.

Por último, existen sistemas de chat en la industria como Whatsapp, Slack y Telegram. Por lo que otra línea de trabajo sería la integración con estas aplicaciones de mensajería que son cotidianas en el mundo empresarial.

En los 3 casos la arquitectura diseñada en este TFG permite la integración tanto de los canales de texto como de voz, sin tener que modificar ningún componente o tener que adicional algún servicio que requiera de una reestructuración de los servicios que tenemos en producción.

5.2.5. Adición de historias de usuario.

El objetivo principal de este proyecto es construir una herramienta que permita la interacción mediante voz y datos, que esté operativa 7x24 y pueda dar servicios en varios idiomas a usuarios que pueden estar en cualquier lugar del mundo. Hemos diseñado 12 historias de usuario de 3 tipos diferentes:

- Comunicaciones mediante voz o texto para realizar gestiones cotidianas en una empresa.
- Conversaciones mediante texto que requieran del envió o recepción de ficheros.
- Alertas a empleados para que sean informados de un hecho relevante.

Por lo que una futura línea seria ir añadiendo nuevos tipos de historias de usuario al proyecto e ir desarrollando y poniendo en producción estas historias, para ello hay hacer un estudio de las necesidades de usuario que no se han contemplado en esta primera versión del producto.

5.2.6. Investigación de metodologías.

Hemos visto que estos proyectos no siguen una metodología estándar de Ingeniera de Software. Esto es debido a que una de las partes más importantes del proyecto es el diseño de conversaciones y el entrenamiento a partir de los errores. Estas dependen en gran medida de una viabilidad técnica o que los servicios asociados permitan hacer la historia de usuario.

También de una viabilidad económica, donde tenemos que evaluar si es rentable por un lado el desarrollo de esa historia de usuario y por el otro el mantenimiento de este. Por último, de necesidades reales de negocio, centrarnos en aquellas funcionalidades que realmente utiliza el usuario y hacen que el producto sea competitivo a nivel comercial.

Dentro del Rol Autor existe un gran campo de sofisticación a la hora de construir una conversación. Aspectos como una anticipación a posibles puntos de abandono del usuario, delimitar bien el ámbito de la herramienta o adaptarnos al contexto de la conversación, parecen cruciales para el buen desempeño de la herramienta.

En la parte de indicadores de productividad del aplicativo hay un campo de trabajos futuros. Por ejemplo, el análisis de los índices de desempeño y los umbrales idóneos para poder indicar que el aplicativo esta funcionado con un nivel de calidad satisfactorio para el usuario.

Las metodologías asociadas a la gestión del proyecto, desempeño del aplicativo y gestión de las conversaciones están muy relacionadas en esta clase de proyectos, siendo una vía de posibles trabajos futuros.

Referencias bibliográficas

Acevedo, E., Serna, A., & Serna, E. (2017). Principios y características de las redes neuronales artificiales. Desarrollo e innovación en ingeniería, 173.

Aguado, L. R., Serrano, A. M. G., & Fernández, P. M. (2002). Planteamiento semántico y pragmático para gestión de diálogos en asistentes virtuales. Procesamiento del lenguaje natural, 28.

Arrestegui, L. B. (2012). Fundamentos históricos y filosóficos de la Inteligencia Artificial. UCV-HACER. Revista de Investigación y Cultura, 1(1), 87-92.

Astobiza, A. M. (2021). Inteligencia Artificial para el bien común (AI4SG): IA y los Objetivos de Desarrollo Sostenible. Arbor, 197(802), a629-a629.

Badaró, S., Ibañez, L. J., & Agüero, M. J. (2013). Sistemas expertos: fundamentos, metodologías y aplicaciones. *Ciencia y tecnología*, (13), 349-364.

Balbin, I. (1985). Introductory Papers to Logic Programming and Prolog. Springer.

Cahn, J. (2017). CHATBOT: Architecture, design, & development. University of Pennsylvania School of Engineering and Applied Science Department of Computer and Information Science.

Catalán Ludwig, I. (2011). Robustez a Variabilidad de Locutor en Reconocimiento de Voz con VTLN.

Clips. (1994). Third Conference on CLIPS Proceedings.

Dey, R., & Salem, F. M. (2017, August). Gate-variants of gated recurrent unit (GRU) neural networks. In 2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS) (pp. 1597-1600). IEEE.

Muñoz, J. M. S. (2016). Análisis de Calidad Cartográfica mediante el estudio de la Matriz de Confusión. Pensamiento matemático, 6(2), 9-26.

Galvañ Sala, D. A. (2021). Comparativa de técnicas para la prevención del sobreajuste en redes neuronales.

García Romillo, V. (2019). Adaptación del sistema de conversión de texto a voz extremo a extremo tacotron a otras lenguas.

Gelbukh, A. (2010). Procesamiento de lenguaje natural y sus aplicaciones. Komputer Sapiens, 1, 6-11.

González Ávila, N., López Martínez, I., & Hernández García, N. (2020). Aproximación al Análisis de Benchmark sobre Asistentes Virtuales.

Goodfellow, I., Bengio, Y. y Courville, A. (2016). The Deep Learning Book. Cambridge (Estados Unidos): The MIT Press.

Hochreiter, S. y Schmidhuber, J. (1997). Long Short-Term Memory. Journal Neural Computation

Lillicrap, T. P., Santoro, A., Marris, L., Akerman, C. J., & Hinton, G. (2020). Backpropagation and the brain. Nature Reviews Neuroscience, 21(6), 335-346.

López, R. F., & Fernández, J. M. F. (2008). Las redes neuronales artificiales. Netbiblo.

Macias-Huerta, P. I., Cabañas-Rocha, R., Valdespino, C. N. L., & Santamaría-Bonfil, G. (2021). Aprende en el mundo de CARLA. Algoritmo Memético para el Problema de Ventas por Internet con Costos de Envío (MAIShOP).

Marchena, O. G. (2009). Lingüística española e Inteligencia Artificial Aplicación informática de gramáticas de restricciones para la confección de agentes de diálogo. Interlingüistica, (18), 472-483.

McCarthy, J. (1989). Artificial intelligence, logic and formalizing common sense. In *Philosophical logic and artificial intelligence* (pp. 161-190). Springer, Dordrecht.

Mondéjar, J. A., Mondéjar Jiménez, J., & Vargas Vargas, M. (2007). Docencia virtual en universidades presenciales: experiencia en la Universidad de Castilla la Mancha.

Pohl, A. (2012, November). Classifying the Wikipedia Articles into the OpenCyc Taxonomy. In *WoLE@ ISWC* (pp. 5-16).

Rajagopal, H. (2005). JENA: A Java API for ontology management. *IBM Corporation, Colorado Software Summit*, 23-28.

Rossini, P. (2000). Using Expert Systems and Artificial Intelligence For Real Estate Forecasting. Sixth Annual Pacific-Rim Real Estate Society Conference.

Salas, R. (2004). Redes neuronales artificiales. Universidad de Valparaıso. Departamento de Computación, 1, 1-7.

Sierra Rivera, W. (2020). Comprensión del Lenguaje Natural e Integración de Asistente Virtual para el manejo de cuentas y comercios. *Computer Engineering;*.

Turban, E. (1995). Decision Support and Expert Systems (4ta edición). EE.UU. Prentice-Hall.

Vásquez, A. C., Quispe, J. P., & Huayna, A. M. (2009). Procesamiento de lenguaje natural. Revista de investigación de Sistemas e Informática, 6(2), 45-54.

Zhang, L., & Zhang, B. (1999). A geometrical representation of McCulloch-Pitts neural model and its applications. IEEE Transactions on Neural Networks, 10(4), 925-929.

Índice de acrónimos

ALU: Arithmetic Logic Unit.

API: Application Programming Interface.

APP: Application.

ASR: Audio Speech Recognition.

CPU: Central Processing Unit.

Db: Decibels.

DM: Dialog Manager.

FLAC: Free Lossless Audio Codec.

LSTM: Long Short-Term Memory.

GPU: Graphics Processing Unit.

IPA: International Phonetic Alphabet.

JSON: JavaScript Object Notation.

KPI: Key Performance Indicators.

MP3: Mpeg Audio Layer 3.

MP4: Mpeg Audio Layer 4.

NLU: Natural Language Understanding.

Ogg: Online Gamers Guild.

PBX: Private Branch Exchange

RNA: Redes Neuronales Artificiales.

RNN: Recurrent Neural Networks.

TTS: Text to Speech.

URL: Uniform Resource Locator.

WAV: Windows Wave.

YAML: Yet another markup language.