UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

Creación de chatbots en Amazon Web Services y Google Cloud

Presentado por Mario Lopez Matamala
en Universidad de Burgos — 9 de julio de 2024
Tutor: Jose Manuel Aroca Fernandez

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

Resumen

Los chatbots, también conocidos como bots de charla o bots conversacionales, son aplicaciones de software que surgieron en el siglo pasado. Estos programas simulan una conversación con una persona real, proporcionando respuestas automáticas y previamente establecidas a un conjunto de entradas del usuario. A lo largo de los años, la tecnología de los chatbots ha evolucionado significativamente, desde simples scripts de respuesta hasta sistemas avanzados de inteligencia artificial capaces de comprender y generar lenguaje natural. Hoy en día, los chatbots son utilizados en una variedad de aplicaciones, desde servicios de atención al cliente hasta asistentes personales virtuales.

Amazon Web Services (AWS) y Google Cloud Platform (GCP) son dos de los principales proveedores de servicios de computación en la nube que ofrecen una amplia gama de servicios para el desarrollo y despliegue de chatbots. AWS proporciona Amazon Lex y Google Cloud ofrece Dialogflow, ambas son plataformas sencillas que facilita la creación de interfaces conversacionales, utilizando técnicas de procesamiento del lenguaje natural (NLP) y aprendizaje automático.

El objetivo de este proyecto es explorar estos servicios, examinando sus productos y viendo sus diferencias. AWS y GCP disponen de multitud de servicios, algunos de los cuales son similares en funcionalidad, como los servicios de traducción de texto y detección de lenguaje, mientras que otros presentan enfoques distintos.

Además, el chatbot desarrollado en este proyecto será capaz de responder preguntas sobre cómo crearse a sí mismo. Este podrá guiar a los usuarios a través de los pasos necesarios para configurar y desplegarlo en la nube, utilizando tanto AWS como GCP, a la vez que se configurar el resto de servicios relevantes para su desarrollo.

La herramienta se encuentra disponible en:
TODO

Descriptores

Chatbots, Bots conversacionales, Inteligencia artificial, Procesamiento del lenguaje natural, Amazon Web Services, AWS, Google Cloud Platform, GCP, Amazon Lex, Dialogflow, Computación en la nube, Aprendizaje automático, Servicios de traducción de texto, Detección de lenguaje.

Abstract

Chatbots, also known as conversational bots, are software applications that emerged in the last century. These programs simulate a conversation with a real person, providing automatic and pre-established responses to a set of user inputs. Over the years, chatbot technology has evolved significantly, from simple response scripts to advanced artificial intelligence systems capable of understanding and generating natural language. Today, chatbots are used in a variety of applications, from customer service to virtual personal assistants.

Amazon Web Services (AWS) and Google Cloud Platform (GCP) are two of the main providers of cloud computing services that offer a wide range of services for the development and deployment of chatbots. AWS provides Amazon Lex and Google Cloud offers Dialogflow; both are user-friendly platforms that facilitate the creation of conversational interfaces using natural language processing (NLP) techniques and machine learning.

The goal of this project is to explore these services by examining their products and identifying their differences. AWS and GCP offer a multitude of services, some of which are similar in functionality, such as text translation and language detection services, while others present different approaches.

Additionally, the chatbot developed in this project will be able to answer questions about how to create itself. It will guide users through the necessary steps to configure and deploy it in the cloud, using both AWS and GCP, while also configuring the other relevant services for its development.

The tool is available at:
TODO

Keywords

Chatbots, Conversational bots, Artificial intelligence, Natural language processing, Amazon Web Services, AWS, Google Cloud Platform, GCP, Amazon Lex, Dialogflow, Cloud computing, Machine learning, Text translation services, Language detection.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

En la era digital actual, las empresas están constantemente buscando formas de mejorar la eficiencia y la experiencia del cliente. Los chatbots han surgido como la herramienta esencial capaz de lograr sus objetivos. Desde su aparición en el siglo pasado, los chatbots han evolucionado significativamente, pasando de simples scripts de respuesta a sistemas avanzados de inteligencia artificial capaces de comprender y generar lenguaje natural, como el ofrecido por la empresa OpenAI [22].

Las empresas que implementan estas tecnologías no sólo tienen como objetivo mejorar la interacción con el cliente sino también optimizar los procesos internos y reducir los gastos operativos. Los chatbots se han utilizado en muchas industrias, incluida la atención al cliente, el comercio electrónico, la atención médica y muchas más. Según un estudio de McKinsey & Company, el uso de chatbots puede reducir los costos de atención al cliente en hasta un 30 %, al mismo tiempo que mejora la resolución de problemas en primera instancia en un 20 % [7].

Siguiendo las recomendaciones de expertos en la industria, como las proporcionadas por Gartner, las empresas deben considerar varios factores al elegir una plataforma de chatbot, incluyendo la capacidad de integración, la escalabilidad y el soporte técnico [12].

El desarrollo y la implementación de chatbots avanzados también han sido impulsados por avances en el procesamiento del lenguaje natural (NLP) mediante herramientas como el Stanford CoreNLP y técnicas avanzadas en el reconocimiento de entidades nombradas [25, 37]. Estos avances permiten que los chatbots no solo respondan a consultas básicas, sino que también proporcionen respuestas contextualmente relevantes y complejas.

En este contexto, el presente trabajo se enfoca en el desarrollo de dos chatbots utilizando los servicios en la nube de AWS y GCP. Este proyecto tiene como objetivo comparar la eficacia y eficiencia de las herramientas y servicios proporcionados por ambas plataformas para el desarrollo de chatbots, así como proporcionar una guía detallada y práctica para la implementación de chatbots en entornos empresariales utilizando tecnologías de computación en la nube.

El proyecto integra servicios avanzados como Amazon Lex, Lambda, S3, Textract, Comprehend y Translate en el caso de AWS, y Dialogflow, Cloud Functions y Cloud Storage en el caso de GCP. Esta integración permitirá evaluar las capacidades y limitaciones de cada plataforma en la creación de soluciones de inteligencia artificial conversacional

1.1. Estructura de la memoria

La memoria sigue la siguiente estructura:

- **Introducción:** breve descripción del problema a resolver y la solución propuesta. Estructura de la memoria y listado de materiales adjuntos.
- **Objetivos del proyecto:** exposición de los objetivos que persigue el proyecto.
- **Conceptos teóricos:** breve explicación de los conceptos teóricos clave para la comprensión de la solución propuesta.
- **Técnicas y herramientas:** listado de técnicas metodológicas y herramientas utilizadas para gestión y desarrollo del proyecto.
- **Aspectos relevantes del desarrollo:** exposición de aspectos destacables que tuvieron lugar durante la realización del proyecto.
- **Trabajos relacionados:** Trabajos en los que se basó la realización del proyecto.
- **Conclusiones y líneas de trabajo futuras:** conclusiones obtenidas tras la realización del proyecto y posibilidades de mejora o expansión de la solución aportada.

Junto a la memoria se proporcionan los siguientes anexos:

- **Plan del proyecto software:** planificación temporal y estudio de viabilidad del proyecto.

- **Especificación de requisitos del software:** se describe la fase de análisis; los objetivos generales, el catálogo de requisitos del sistema y la especificación de requisitos funcionales y no funcionales.
- **Especificación de diseño:** se describe la fase de diseño; el ámbito del software, el diseño de datos, el diseño procedimental y el diseño arquitectónico.
- **Manual del programador:** recoge los aspectos más relevantes relacionados con el código fuente (estructura, compilación, instalación, ejecución, pruebas, etc.).
- **Manual de usuario:** guía de usuario para el correcto manejo de la aplicación.
- **Acrónimos:** Índice de acrónimos.

1.2. Materiales adjuntos

Los materiales que se adjuntan con la memoria son:

- **Anexos**: consultar la documentación técnica.
- **Herramienta web**: Visitar página web.
- **Vídeo de presentación**: ver presentación del TFG.
- **Vídeo de demostración**: ver demostración funcional.

Además, el repositorio del proyecto se puede encontrar en el siguiente enlace: Repositorio Github

2. Objetivos del proyecto

A continuación se detallan los objetivos que se persiguen con la realización de este proyecto:

2.1. Objetivos generales

- Desarrollar y comparar dos chatbots utilizando los servicios en la nube de AWS y GCP.
- Evaluar la eficacia y eficiencia de las herramientas y servicios proporcionados por AWS y GCP para el desarrollo de chatbots.
- Proporcionar una guía detallada y práctica para la implementación de chatbots en entornos empresariales utilizando tecnologías de computación en la nube.

2.2. Objetivos personales

- Adquirir conocimientos avanzados sobre el desarrollo e implementación de chatbots en entornos de computación en la nube.
- Desarrollar habilidades prácticas en el uso de servicios de AWS y GCP para proyectos de inteligencia artificial y procesamiento del lenguaje natural.
- Abarcar el máximo número posible de conocimientos adquiridos durante el grado.
- Explorar metodologías, herramientas y estándares utilizados en el mercado laboral.

3. Conceptos teóricos

Los conceptos teóricos de este proyecto residen en el procesamiento del lenguaje natural(NLP en adelante) el cual es una subdisciplina de la inteligencia artificial que se dedica a la interacción entre las computadoras y los lenguajes humanos. Su objetivo principal permitira las computadoras comprender, interpretar y generar lenguaje humano.

3.1. Procesamiento del Lenguaje Natural

El estudio del procesamiento del lenguaje natural tiene sus origen en la década de 1950, cuando Alan Turing propuso el Test de Turing como criterio para la inteligencia artificial. Desde entonces, el campo ha evolucionado pasando por varias fases, incluidas las primeras técnicas basadas en reglas, el auge de los modelos estadísticos en la década de 1990 y el aprendizaje profundo. Los elementos claves para su estudio son:

- **Tokenización**: Consiste en el proceso de dividir el texto en unidades más pequeñas llamadas tokens (generalmente palabras o frases) para su posterior análisis de una manera más sencilla. [22].
- **Etiquetado de Partes del Discurso (POS Tagging)**: Asignar etiquetas gramaticales (como sustantivos, verbos, adjetivos) a cada token. Esto ayuda a entender la estructura gramatical del texto [25].
- **Reconocimiento de Entidades Nombradas (NER)**: Identificar y clasificar entidades mencionadas en el texto, como nombres de personas, organizaciones, lugares, fechas, etc. [37].

- **Análisis Sintáctico y Semántico**: Comprender la estructura gramatical (análisis sintáctico) y el significado (análisis semántico) de las frases. El análisis sintáctico descompone las oraciones en su estructura jerárquica, mientras que el análisis semántico busca comprender el significado y las relaciones entre las palabras [22].
- **Análisis de Sentimientos**: Determinar la opinión o el sentimiento expresado en un texto, que puede ser positivo, negativo o neutral. Esta técnica es ampliamente utilizada en el análisis de redes sociales y opiniones de clientes [32].
- **Traducción Automática**: Convertir texto de un idioma a otro mediante técnicas de NLP. Los modelos modernos de traducción automática utilizan redes neuronales recurrentes y transformadores para generar traducciones de gran precisión [2].

3.2. Técnicas avanzadas del NLP

En los últimos años, los modelos de *deep learning* han revolucionado el campo del NLP. Entre los más destacados, se encuentran:

- **Redes Neuronales Recurrentes (RNNs)**: Son redes neuronales especializadas en procesar secuencias de datos, manteniendo información temporal y dependencias a través de sus conexiones recurrentes. A diferencia de las redes neuronales tradicionales, las RNNs pueden recordar información a lo largo de las secuencias de datos gracias a su estructura recurrente. Esto les permite realizar tareas como la generación de texto, la traducción automática y el reconocimiento del habla [18].
- **Long Short-Term Memory (LSTM)**:Es una variante de las RNNs que aborda el problema del gradiente desaparecido. Las LSTM introducen una estructura de células que permite mantener y actualizar información a lo largo de las secuencias de datos. Esto permite a los modelos capturar dependencias a largo plazo en los datos secuenciales, siendo muy efectivos en tareas de traducción de idiomas y generación de texto [18].
- **Transformadores:** Introducidos por Vaswani et al. (2017), los transformadores han demostrado su efectividad en una amplia gama de tareas de NLP, utilizando mecanismos de atención para procesar secuencias de datos de manera más precisa que las RNNs tradicionales.

BERT (Bidirectional Encoder Representations from Transformers) y GPT-3 (Generative Pre-trained Transformer 3) son solo algunos ejemplos [11, 43].

- **Word Embeddings:** Técnicas como Word2Vec y GloVe transforman palabras en vectores numéricos de alta dimensión que capturan semánticas y relaciones sintácticas [28, 33].
- **Contextualized Word Representations:** Los modelos como ELMo y BERT generan representaciones de palabras que tienen en cuenta el contexto en el que aparecen, clasificando el texto y evitando la desambiguación semántica. [11, 34].

3.3. Comprensión del Lenguaje Natural

La comprensión del lenguaje natural (NLU) es una subdisciplina dentro del NLP que se centra específicamente en la comprensión del significado del lenguaje humano, este se ocupa de interpretar y entender la intención detrás de las palabras y frases.

Si en NLP procesan entradas de texto o voz, realizan tokenización, etiquetado de partes del discurso y análisis sintáctico, NLU interpreta la intención del usuario, extraen entidades relevantes y comprenden el contexto de la conversación.

- **Detección de Intenciones:** Esta tarea implica identificar la intención o propósito del usuario a partir de su entrada. Por ejemplo, si un usuario escribe "¿Cuál es el clima hoy?", la intención es obtener información sobre el clima.
- **Reconocimiento de Entidades:** NLU también se encarga de extraer entidades específicas mencionadas en el texto, como nombres de personas, fechas, ubicaciones y números. Por ejemplo, en la pregunta "¿Cuál es el clima hoy en Burgos?", "Burgos.^{es} una entidad de ubicación que el chatbot debe reconocer para proporcionar una respuesta precisa.
- **Análisis de Contexto:** Los sistemas de NLU deben ser capaces de mantener y utilizar el contexto de la conversación para interpretar correctamente las entradas del usuario. Esto incluye la capacidad de comprender referencias anafóricas (por ejemplo, "él", .^{el}la", .^{es}o") y la información proporcionada en interacciones anteriores (atributos de sesión).

- **Desambiguación Semántica:** Las palabras y frases en el lenguaje natural a menudo tienen múltiples significados. NLU utiliza técnicas de desambiguación para determinar el sentido correcto de una palabra o frase en un contexto dado.
- **Manejo de Variabilidad del Lenguaje:** Los usuarios pueden expresar la misma intención de muchas maneras diferentes. Los sistemas de NLU deben ser capaces de reconocer sinónimos, variaciones gramaticales y diferentes formas de expresiones para comprender la intención subyacente.

Amazon Lex y Google Cloud Dialogflow son herramientas que utilizan técnicas avanzadas de NLP para interpretar y responder al lenguaje humano. Ambas plataformas se basan en la comprensión del lenguaje natural NLU para identificar las intenciones del usuario y proporcionar respuestas contextualmente apropiadas. Como se verá más adelante, estas serán las principales tareas las cuales realizarán los chatbots.

4. Técnicas y herramientas

4.1. Metodología de trabajo

Desarrollo Iterativo con Scrum

Scrum es un marco de trabajo relativamente estructurado con roles específicos dentro de la metodología Agile, entre los cuales destacan el Product Owner, el Scrum Master y el desarrollador. Este marco se puede utilizar tanto para la gestión de proyectos como para el desarrollo de productos, especialmente en el despliegue de software. Con Scrum, los proyectos se dividen en iteraciones cortas llamadas sprints.

Se ha optado por esta metodología debido a su alta adaptabilidad y su capacidad para generar entregas tempranas de valor. Con Scrum, se obtienen productos viables y evaluables por el usuario final desde las primeras fases del proyecto, lo que permite realizar ajustes y mejoras continuas basadas en el feedback recibido. También destaca por su idoneidad a la hora de enfrentarse a nuevos desarrollos.

4.2. Control de versiones

- Herramientas consideradas: Git [13], GitHub Desktop [14]
- Herramienta elegida: Github Desktop

Git y GitHub Desktop son herramientas relacionadas con el control de versiones.

Una de las ventajas de Git es que permite a cada desarrollador tener una copia local del repositorio completo y, aunque puede ser menos eficiente para

proyectos muy grandes, es más sencillo de utilizar para proyectos pequeños. Además, el sistema de ramificación de Git es más intuitivo y facilita la tarea de los desarrolladores. GitHub Desktop, por otro lado, proporciona una interfaz gráfica amigable para interactuar con Git, haciendo que las operaciones de control de versiones sean más accesibles para aquellos que prefieren no trabajar con la línea de comandos.

En mi proyecto, he trabajado con una única rama en el repositorio de control de versiones. Sin embargo, en la mayoría de los proyectos de desarrollo de software se suele trabajar con varias ramas para organizar el trabajo.

- **Rama principal (main o master):** Esta es la rama principal del repositorio. Contiene el código fuente en estado estable y listo para producción. Todo el trabajo que se realiza en otras ramas se integra en esta rama una vez que ha sido revisado y probado.
- **Rama de desarrollo (develop):** Esta rama contiene el código en el que se está trabajando activamente. Es la rama donde se integran todas las nuevas características y correcciones antes de ser fusionadas con la rama principal. Se utiliza para pruebas y revisiones antes de pasar a producción.
- **Ramas de características (feature branches):** Estas ramas se crean a partir de la rama de desarrollo para trabajar en nuevas funcionalidades específicas. Una vez que la característica está completa y ha sido probada, se fusiona de nuevo en la rama de desarrollo.

4.3. Alojamiento del repositorio

- Herramientas consideradas: GitHub [16], GitLab [17]
- Herramienta elegida: GitHub.

GitLab ofrece una solución completa que incluye no solo la gestión de repositorios, sino también un conjunto de herramientas DevOps que abarcan desde la planificación hasta la entrega y monitorización del software.

He decidido utilizar Github dado que lo conozco bastante bien, porque se utiliza en algunas asignaturas del Grado de Ingeniería Informática, su interfaz es muy intuitiva y porque es muy popular, lo que facilita la resolución de problemas gracias a su mayor comunidad

4.4. Comunicación

- Herramientas consideradas: email, GitHub y Microsoft Teams.
- Herramientas elegidas: email y Microsoft Teams.

La comunicación con el tutor se desarrolló mediante reuniones periódicas en Microsoft Teams, así como mediante correos electrónicos.

4.5. Entorno de desarrollo integrado (IDE)

- Herramientas consideradas: Sublime Text [19], Visual Studio Code [27].
- Herramienta elegida: Visual Studio Code.

He utilizado Visual Studio Code para probar a trabajar con un entorno virtual en mi propia máquina, invocando a los servicios en la nube de GCP. Sin embargo, esto no es necesario ya que se pueden utilizar perfectamente dentro de su propia consola.

4.6. Integración y despliegue continuo(CI/CD)

En este proyecto no se ha seguido ningún proceso de automatización durante el proceso de compilación y despliegue del software.

4.7. Documentación de la memoria

- Herramientas consideradas: Texmaker [4] y Overleaf [31].
- Herramienta elegida: Texmaker.

Texmaker es un editor de texto gratuito, multiplataforma y que integra diversas herramientas necesarias para desarrollar documentos LaTeX. *Texmaker* incluye soporte *Unicode*, corrección ortográfica, auto-completado y un visor de PDF incorporado que es realmente útil.

Aunque al principio del desarrollo de la memoria lo empecé con Overleaf, acabé usando Texmaker.

4.8. Calidad y consistencia de código

- Herramientas consideradas: Pylint [24] y Flake8 [8].
- Herramienta elegida: Pylint.

Pylint es una herramienta de análisis de código estático para *Python*, diseñada para detectar errores y mejorar la calidad del código. Este analizador de código se utilizar para verificar sintaxis, semántica y obliga a seguir las convenciones de estilo de *Python*.

Se ha escogido *Pylint* frente a *Flake8* porque, adicionalmente, permite medir la calidad del código en términos de complejidad y legibilidad, lo que favorece un mantenimiento posterior. Además, con *Pylint* podemos hacer informes que señalan todos los fallos, aumentando la productividad.

Durante el desarrollo del proyecto, se utilizaron ciertas exclusiones en Pylint para evitar advertencias que no eran relevantes. Las exclusiones específicas son las siguientes:

- **C0303 (Trailing Whitespace):** Esta advertencia se refiere a espacios en blanco al final de las líneas de código. Se deshabilitó para evitar múltiples advertencias por un detalle menor que no afecta la ejecución del código.
- **C0301 (Line Too Long):** Esta advertencia se genera cuando una línea de código supera la longitud máxima recomendada (normalmente 80 o 100 caracteres). Se deshabilitó porque en ciertos casos las líneas tienen que ser lo bastante largas, en especial en la carga de datos a la base de datos.
- **E0401 (Import Error):** Esta advertencia se genera cuando Pylint no puede encontrar un módulo importado. Se deshabilitó para evitar falsos positivos, especialmente al importar módulos no instalados en mi entorno ya que se estaban usando dentro de los servicios de AWS y GCP.

4.9. Cobertura de código

- Herramientas consideradas: Coverage [3] y Pytest-cov [36].
- Herramienta elegida: Ninguna.

Coverage es una herramienta que permite medir la cobertura de código en *Python*. Uno de los aspectos relevantes de *coverage* es que, con pocos comandos, permite generar un informe *HTML* muy intuitivo que guía al desarrollador hacia los fallos detectados.

Sin embargo, en este proyecto no se han generado tests debido a la naturaleza del proyecto. Con lo cual, no se ha hecho uso de esta herramienta.

4.10. Desarrollo web

HTML

HTML [45] es el lenguaje de marcado de hipertexto estándar para crear páginas web. Es un lenguaje que utiliza etiquetas para definir la esrtructura y el contenido de una página web.

CSS

CSS [44] es un lenguaje de hojas de estilo que se utiliza para dar un aspecto y diseño agradables a una página web. Se utiliza junto con *HTML*.

5. Aspectos relevantes del desarrollo del proyecto

En esta sección se muestra un resumen de los servicios utilizados a lo largo del proyecto en ambas plataformas, así como una descripción detallada de la configuración del entorno.

5.1. Inicio del proyecto

Este proyecto surge con el afán de conocer y comprender las plataformas de AWS y GCP, las cuales ofrecen multitud de servicios en la nube para realizar diferentes tareas.

Inicialmente, consideramos configurar un chatbot simple utilizando OpenAI que pudiera entrenarse en base a los documentos de texto recibidos y responder preguntas derivadas desde esos documentos. Sin embargo, encontramos varias limitaciones, como la incapacidad actual de modelos como ChatGPT para entrenarse directamente a partir de documentos proporcionados por el usuario en tiempo real y responder a preguntas específicas sobre esos documentos.

Finalmente, tras consultar con el tutor Jose Manuel, decidimos explorar una arquitectura más compleja. Así fue como llegamos a evaluar y utilizar las plataformas de AWS y GCP, aprovechando los diversos servicios que ofrecen. Esto nos permitió diseñar y crear un chatbot personalizado, en el cual podemos restringir el flujo de conversación y controlar las respuestas.

Por otro lado, esta decisión se tomó con la intención de comprender como se realizan los chatbots de cero, entendiendo así lo que hay por debajo y de lo que están compuestos mucho de estos.

5.2. Metodologías

Durante el desarrollo del proyecto se ha seguido una metodología ágil, *Scrum*, concretamente. El inconveniente más evidente de esta filosofía en un proyecto educativo es la falta de un equipo de personas que cubran los diferentes roles necesarios. Sin embargo, he intentado ponerme en el papel de cada componente del equipo, haciéndome preguntas constantemente sobre el tiempo disponible, el producto final requerido en cada sprint y qué esperaría un potencial cliente. Cada *issue* se describe detalladamente, especificando el objetivo.

Estos son algunos de los procesos más relevantes que he seguido:

- Realización de iteraciones, *sprints*, de forma constante. Los sprints han tenido una duración aproximada de unas dos semanas, aunque algunos se han incrementado debido a cambios en el objetivo final del proyecto y en la forma de actuar. Por ejemplo, en el *sprints* 6, se cambió la lógica con la que las respuestas se recogían en base a las preguntas. Aquí he intentado realizar un esfuerzo extra en cuanto a lo esperable en las reuniones diarias y, aunque han sido discusiones conmígo mismo y con el tutor, el resultado ha sido satisfactorio.

- Disposición de tareas, *Issues*, en cada uno de los sprints. Estos son tareas específicas que se deben completar dentro del marco temporal del sprint. Esto asegura que todas las actividades necesarias para avanzar en el proyecto estén identificadas y asignadas, facilitando un seguimiento efectivo. Además, cada commit se ha referenciado a su *Issues* correspondiente.

5.3. Extracción de la información

La primera fase de este proyecto consistió en extraer información. Ha sido la más extensa y, aunque ha sido un proceso continuo que se ha llevado a cabo a lo largo de todo el proyecto entendiendo el funcionamiento de los sistemas que proporcionan, ha tenido lugar en el primer *Sprint*.

Para esto, es necesario tener en cuenta varias bases de datos bibliográficas que ofrecen diversos recursos para los investigadores. Entre las más importantes se encuentran Google Scholar y Crossref. En ellas he podido encontrar diferentes artículos científicos de proyectos sencillos utilizando los servicios, aunque ninguno era lo suficientemente complejo, me ha servido para entenderlos lo suficiente como para empezar a trabajar por mi cuenta.

Sin embargo, donde más he podido obtener información ha sido en la documentación de ambas plataformas. Las dos proporcionan una gran cantidad de documentación para todos sus servicios, cubriendo desde conceptos básicos hasta detalles técnicos avanzados.

A pesar de la abundancia de información disponible en ambas plataformas, encontré que la documentación de AWS es mucho más sencilla. Esto fue particularmente útil en las primeras etapas del proyecto, cuando estaba familiarizándome con los diferentes componentes y cómo integrarlos entre ellos.

Por otro lado, la documentación de Google Cloud Platform, aunque igualmente completa y detallada, a veces se adentra en temas muy técnicos y específicos que no eran necesarios para mi proyecto. Esto hizo que tuviera que invertir más tiempo en filtrar la información y extraer solo lo más relevante para las necesidades del desarrollo del chatbot.

5.4. Amazon Web Services

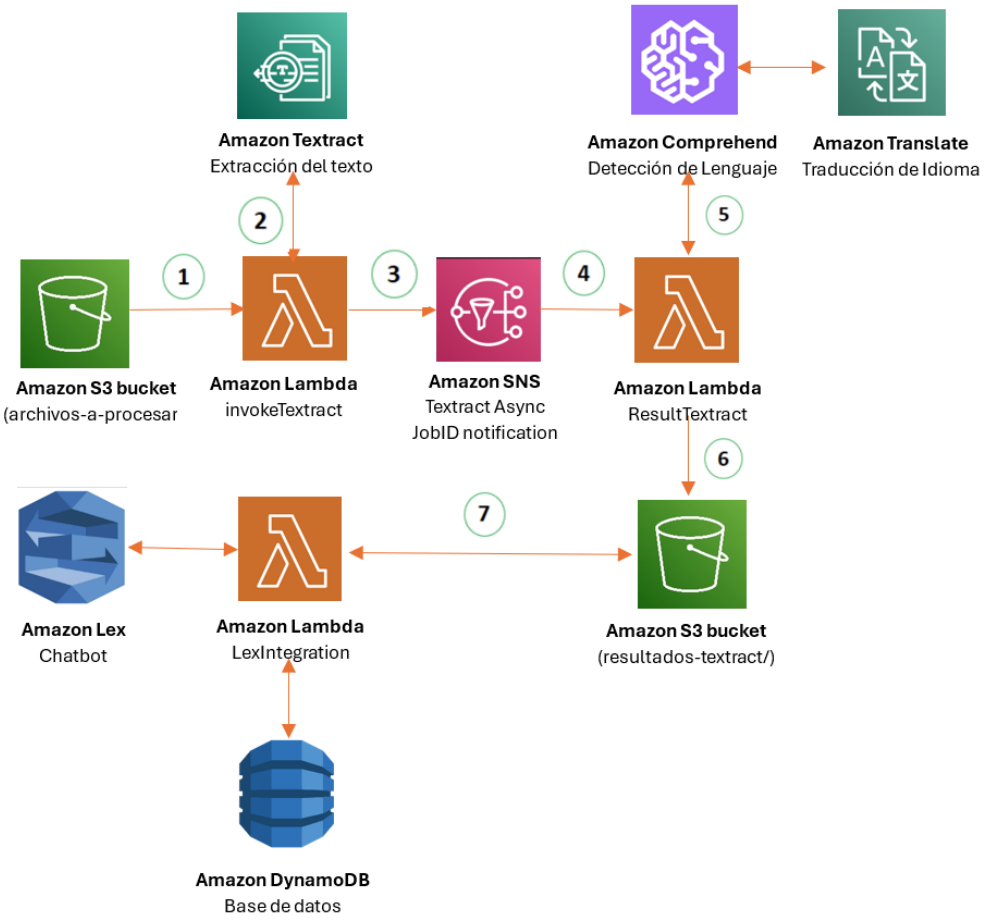

Figura 5.1: Ecosistema AWS. Fuente: elaboración propia

Qué elementos contienen

1. Amazon S3 (Simple Storage Service).

   Es un servicio de almacenamiento de objetos que ofrece escalabilidad, disponibilidad de datos y seguridad. Los objetos se almacenan en contenedores llamados buckets.

   En mi caso, almaceno los archivos de entrada y los resultados del procesamiento. Hay dos buckets: uno para los archivos a procesar **(archivos-a-procesar)** y otro para los resultados procesados **(resultados-textract)**.

2. AWS Lambda.

   AWS Lambda es un servicio de computación serverless que permite ejecutar código en respuesta a eventos sin tener que aprovisionar ni gestionar servidores. Se paga solo por el tiempo de computación consumido.

   Son tres las funciones que utilizo:

   - **invokeTextract:** Esta función invoca Amazon Textract para comenzar la extracción de texto de los documentos almacenados en S3 obteniendo el ID de la extracción.
   - **ResultTextract:** Esta función procesa las notificaciones de finalización de Amazon Textract, obteniendo el texto extraído para luego manejar los resultados, almacenándolos en S3.
   - **LexIntegration:** Esta función integra el chatbot con Amazon Lex y maneja la lógica adicional necesaria, interactuando con DynamoDB.

3. Amazon Textract.

   Amazon Textract es un servicio que utiliza machine learning para extraer texto y datos de documentos escaneados automáticamente. Puede leer archivos escaneados, identificar campos de formulario y tablas.

   Este servicio es invocado por las funciones invokeTextract y ResultTextract para obtener el texto extraído de archivos pdf en formato txt. De esta forma es posible su posterior uso en el bot.

4. Amazon SNS (Simple Notification Service).

   Amazon SNS es un servicio de mensajería que coordina y gestiona la entrega de mensajes a suscriptores en diferentes plataformas y dispositivos. Este envía una notificación cuando Amazon Textract completa el procesamiento de un documento, activando la función Lambda ResultTextract que extrae y procesa los resultados.

5. Amazon Comprehend

   Amazon Comprehend es un servicio de procesamiento del lenguaje natural que utiliza machine learning para encontrar insights y relaciones en el texto. Ofrece capacidades como análisis de sentimientos, detección de entidades, análisis de lenguaje y clasificación de texto. Este es usado por la función ResultTextract para detectar el idioma en el que está escrito el texto.

6. Amazon Translate

   Amazon Translate es un servicio de traducción automática que proporciona traducciones rápidas y de alta calidad entre diferentes idiomas. También utiliza técnicas de machine learning para traducir el texto.

   Este traduce el texto a español en caso de que el idioma detectado por Comprehend sea otro distinto al español, permitiendo así que el bot reciba documentos en cualquier idioma.

7. Amazon Lex

   Amazon Lex es un servicio para construir interfaces de conversación utilizando voz y texto. Utiliza técnicas avanzadas de comprensión del lenguaje natural para entender la intención del usuario y gestionar diálogos de manera dinámica, siguiendo un flujo de conversación entre una persona real y la inteligencia artifical.

   Este da vida al chatbot que interactúa con los usuarios, obteniendo las preguntas que realizan y utilizando los datos procesados y almacenados en DynamoDB para proporcionar las respuestas.

8. Amazon DynamoDB

   Amazon DynamoDB es un servicio de base de datos NoSQL que ofrece escalabilidad automática. En él se encuentra la base de datos con todas las respuestas a las preguntas del usuario. Dispone de tres campos: Nombre del intent reconocido por Lex (IntentName), Texto de la pregunta (Question), Texto de la respuesta (Response).

   También guarda el nombre del documento con la información correspondiente a los pasos de creación del bot, en este caso el atributo Question contiene una clave con el nombre del paso y subpaso. *Funcionamiento*.

9. Amazon IAM

   Amazon IAM es un servicio que permite gestionar el acceso a los servicios y recursos de AWS de forma segura. Con IAM, se pueden crear y gestionar usuarios y grupos, y utilizar permisos para permitir su acceso a los recursos de AWS. En este proyecto, IAM es fundamental para asignar los permisos necesarios a las funciones Lambda y que así puedan interactuar con los otros servicios.

   Se crean tres roles IAM: **AWS_PDF_Textract_Lex_Role** que se usa para las funciones *invokeTextract y ResultTextract*; **AWS_Lambda_LexIntegrati** usado por la función *LexIntegration*; **textract_sns** que se encarga de la notificación SNS.

Funcionamiento

El flujo de trabajo para el desarrollo del chatbot se compone de varias etapas, son dos las secciones que pueden actuar independientemente y que se pueden diferenciar de la siguiente forma:

- **Sección 1: Extracción y Procesamiento de Documentos:** Esta sección se encarga únicamente de la tarea de extracción del texto.
- **Sección 2: Interacción del Chatbot:** Esta sección interactúa con el chabot, reconociendo intents y proporcionando respuestas a las preguntas del usuario. No es necesario que haya texto extraído dentro del bucket para funcionar, ya que actúa independientemente mediante la base de datos.

Cabe destacar que todo este proceso es asíncrono, de forma que se permite cargar y extraer el texto de varios archivos a la vez o tener varios bots funcionando simultáneamente, el sistema es capaz de escalar automáticamente en base a las necesidades.

A continuación se detalla cada paso del proceso. Los pasos de cada sección corresponden con los enumerados en la figura 5.1 *Ecosistema AWS*.

Sección 1: Extracción y Procesamiento de Documentos

1. Carga de Archivos en S3.

   Dentro del bucket y la carpeta *resultados-a-procesar/* se cargan los archivos con extensión .pdf para extraerles el texto y poder trabajar con ellos. Nada más cargarse el archivo pdf, desencadena la función invokeTextract.

   En la carpeta se puede cargar todo tipo de archivos, pero solamente se iniciará el proceso de extracción si es un archivo con extensión .pdf. De lo contrario, no desencadenará a la función lambda.

2. Desencadenación de invokeTextract.

   Una vez que ha recibido la notificación por parte del bucket S3, esta función se activa. Recibe el nombre del bucket y el archivo que se ha cargado. A continuación, utiliza la función de textract start_document_text_detection para obtener el ID del proceso que inicia la extracción.

3. Notificación SNS

   Tras obtener el ID, utiliza el servicio de SNS para notificar a la función ResultTextract que puede continuar con la extracción del texto. Esto activa a la función que recibirá el ID por parte de SNS.

   Los siguientes pasos serán realizados por la función ResultTextract y pueden ser agrupados como el procesamiento de los datos

4. Desencadenación de ResultTextract.

   Esta función lambda recibe la notificación con el ID y comienza la extracción del texto. La forma en la que maneja Textract el texto para extraer admite que se cargue texto ordenado mediante dos columnas, algo típico en ensayos científicos. Por eso se ha implementado una funcionalidad que detecta y agrupa el texto correctamente en caso de que tengo dos columnas.

   En la figura 5.2 *Diagrama de flujo ResultTextract* se muestra el procesamiento de los datos de la función.

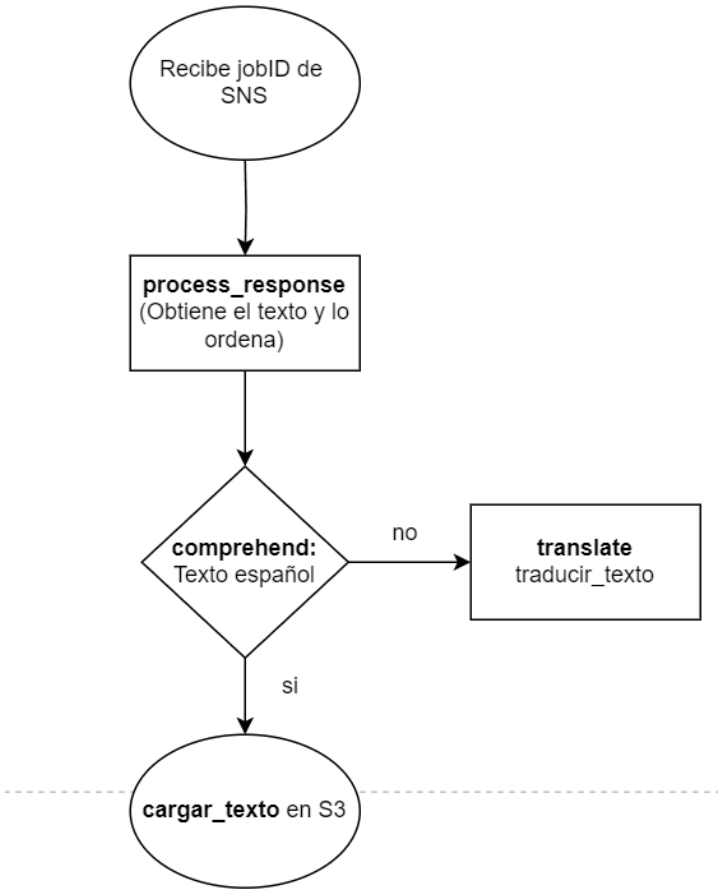

Figura 5.2: Diagrama de flujo función lambda ResultTextract. Fuente: elaboración propia

5. Traducción del texto

   La función ResultTextract utiliza el servicio Amazon Comprehend para detectar el idioma del texto extraído. En caso de que el idioma sea distinto al español, llama al servicio Amazon Translate para traducirlo, por lo que este paso es opcional.

6. Carga de los datos

   Una vez la función ResultTextract ha terminado toda la tarea de procesamiento de texto, carga el archivo en formato txt con el texto extraído dentro del bucket y la carpeta resultados-textract/.

Sección 2: Interacción del Chatbot

7. LexIntegration

   Esta función es la que maneja toda la sección 2. Cada vez que Amazon Lex detecta un intent, llama a la función lambda quien determina como seguir. Los intents que se han añadido son de tres tipos:

   - **Intents de pasos de creación del chatbot**:
     Estos intents se encargan del proceso de creación del chatbot, explicando al usuario todos los pasos. Estos son: *StartTutorial* que se encarga del comienzo del tutorial recibiendo enunciados como çomenzar tutorial"; *NextStep* que permite avanzar entre los pasos del tutorial; *gotostep* que incluye la posibilidad de avanzar a un paso especifico del tutorial.
   - **Intents de respuesta a preguntas de los servicios**:
     Estos intents responden preguntas básicas sobre los servicios utilizados en el proceso de creación del chatbot. Los intents son del estilo de *CreacionRolIAM, CreacionBucketS3, CrearFuncionLambda* y explica conceptos básicos como *"Qué es un rol IAM"*.
   - **Intents de presentación**:
     Estos intents responden preguntas destinadas a la presentación del proyecto, gestionando intents como *ideaTrabajo o Objetivos*.

   La función lambda *LexIntegration* actúa en base al intent que recibe. En cualquiera de los casos, llama a la base de datos de dynamoDB para obtener respuesta a la pregunta del usuario.

   En caso de que haya recibido un intent del paso de creación del chatbot, obtendrá de la base de datos el nombre archivo .txt donde se encuentra la información del paso que tiene que devolver al usuario. Despues de esto, se dirije al bucket a buscar el archivo para terminar por devolver la respuesta al usuario. A lo largo de la sesión, se guardan atributos de sesión con la información del paso actual donde se encuentra el usuario con el fin de poder tener un orden del flujo de creación del chatbot.

   Por ejemplo, si el usuario confirma que ha entendido el paso en el proceso de creación, la función actualiza su registro y en la siguiente interacción devolverá la información correspondiente al siguiente paso. Este atributo también se actualiza en caso de que el usuario quiera saltar a un paso en concreto.

   En los otros dos casos, la función lambda se dirige también a la base de datos para obtener la respuesta. Dado que en la base de datos los intents están agrupados, la función lambda obtiene todos los intents asociados,

analiza su *Question* comparandola con la que ha introducido el usuario y obtiene la respuesta en base a la *Question* que más se asemeje.

5.5. Google Cloud

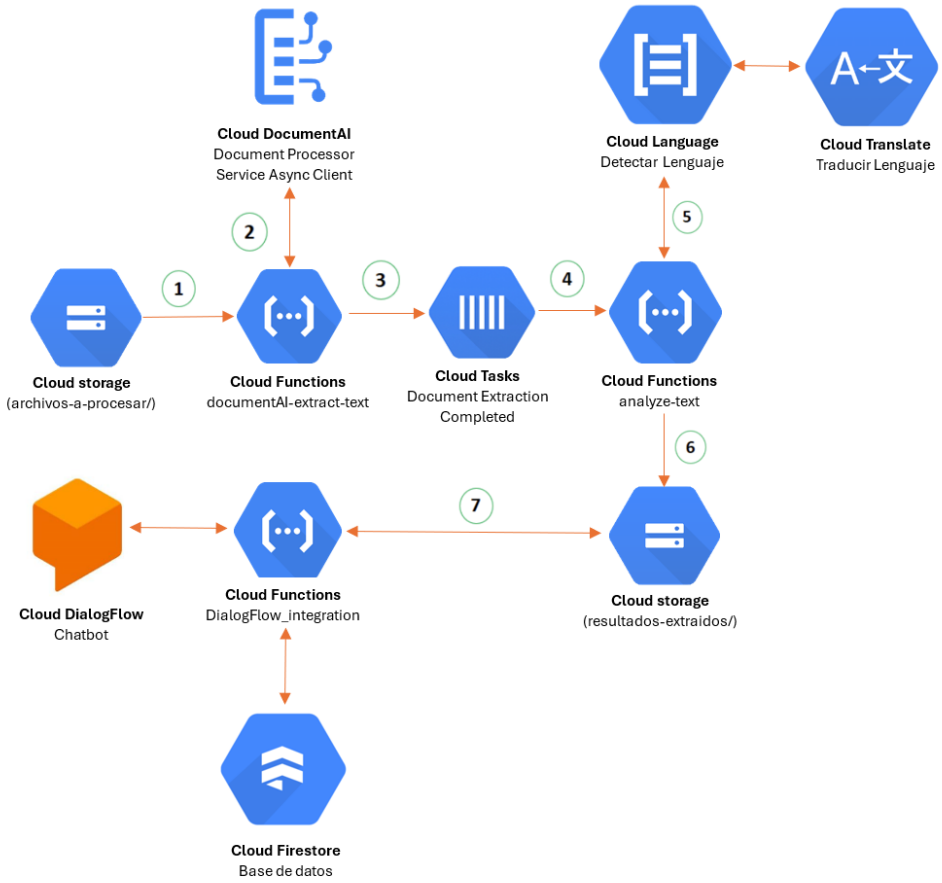

Figura 5.3: Ecosistema GCP. Fuente: elaboración propia

Qué elementos contienen

1. Cloud Storage

   Cloud Storage es un servicio de almacenamiento de objetos de Google Cloud que ofrece escalabilidad, disponibilidad de datos y seguridad. Al

igual que en AWS, los objetos se almacenan en contenedores llamados buckets.

En mi caso, almaceno los archivos de entrada y los resultados del procesamiento. Hay dos buckets: uno para los archivos a procesar **(archivos-a-procesar)** y otro para los resultados procesados **(resultados-extraídos)**.

2. Cloud Functions

   Cloud Functions es un servicio de computación sin servidor que permite ejecutar código en respuesta a eventos sin tener que aprovisionar ni gestionar servidores. Solo se paga por el tiempo de computación consumido.

   Son tres las funciones que utilizo:

   - **documentAI-extract-text:** Esta función invoca Cloud Document AI para comenzar la extracción de texto de los documentos almacenados en Cloud Storage obteniendo el ID de la extracción.
   - **analyze-text:** Esta función procesa las notificaciones de finalización de Cloud Document AI, obteniendo el texto extraído para luego manejar los resultados, almacenándolos en Cloud Storage.
   - **DialogFlow-integration:** Esta función integra el chatbot con DialogFlow y maneja la lógica adicional necesaria, interactuando con Firestore.

3. Cloud Document AI

   Cloud Document AI es un servicio que utiliza machine learning para extraer texto y datos de documentos escaneados automáticamente. Puede leer archivos escaneados, identificar campos de formulario y tablas.

   Este servicio es invocado por las funciones documentAI-extract-text y analyze-text para obtener el texto extraído de archivos PDF en formato TXT, permitiendo su posterior uso en el bot.

4. Cloud Tasks

   Cloud Tasks es un servicio de gestión de colas que coordina y gestiona la entrega de tareas a servicios de back-end de manera asíncrona. Envía una notificación cuando Cloud Document AI completa el procesamiento de un documento, activando la función analyze-text que extrae y procesa los resultados.

5. Cloud Language

   Cloud Language es un servicio de procesamiento del lenguaje natural que utiliza machine learning para encontrar insights y relaciones en el texto. Ofrece capacidades como análisis de sentimientos, detección de entidades, análisis de lenguaje y clasificación de texto. Es utilizado por la función analyze-text para detectar el idioma en el que está escrito el texto.

6. Cloud Translate

   Cloud Translate es un servicio de traducción automática que proporciona traducciones rápidas y de alta calidad entre diferentes idiomas. Utiliza técnicas de machine learning para traducir el texto.

   Este servicio traduce el texto al español en caso de que el idioma detectado por Cloud Language sea otro distinto al español, permitiendo así que el bot reciba documentos en cualquier idioma.

7. Cloud DialogFlow

   Cloud DialogFlow es un servicio para construir interfaces de conversación utilizando voz y texto. Utiliza técnicas avanzadas de comprensión del lenguaje natural para entender la intención del usuario y gestionar diálogos de manera dinámica.

   Este servicio da lugar al chatbot que interactúa con los usuarios, obteniendo las preguntas que realizan y utilizando los datos procesados y almacenados en Firestore para proporcionar las respuestas.

8. Cloud Firestore

   Cloud Firestore es una base de datos NoSQL que ofrece rendimiento rápido y predecible con escalabilidad automática. En él se encuentra la base de datos con todas las respuestas a las preguntas del usuario. Dispone de tres campos: Nombre del intent reconocido por DialogFlow (IntentName), Texto de la pregunta (Question), Texto de la respuesta (Response).

Funcionamiento

El flujo de trabajo para el desarrollo del chatbot se compone de varias etapas, son dos las secciones que pueden actuar independientemente y que se pueden diferenciar de la siguiente forma:

- **Sección 1: Extracción y Procesamiento de Documentos:** Esta sección se encarga únicamente de la tarea de extracción del texto.
- **Sección 2: Interacción del Chatbot:** Esta sección interactúa con el chabot, reconociendo intents y proporcionando respuestas a las preguntas del usuario. No es necesario que haya texto extraído dentro del bucket para funcionar, ya que actúa independientemente mediante la base de datos.

Al igual que en AWS, este proceso es asíncrono, de forma que se permite cargar y extraer el texto de varios archivos a la vez o tener varios bots funcionando simultáneamente. Los pasos de cada sección corresponden con los enumerados en la figura 5.3:

Sección 1: Extracción y Procesamiento de Documentos

1. Carga de Archivos en Cloud Storage.

   Dentro del bucket y la carpeta *archivos-a-procesar/* se cargan los archivos con extensión .pdf para extraerles el texto y poder trabajar con ellos. Nada más cargarse el archivo pdf, desencadena la función documentAI-extract-text.

   Dentro de esta carpeta se pueden cargar todo tipo de archivos. Google cloud desencadenará a la función independientemente del archivo, será la función lambda quién decida empezar o no el proceso de extracción en base al tipo de archivo. A diferencia de AWS, no es posible limitar la desencadenación de la función según el tipo de archivo ni la carpeta donde se introduce, ya que notifica por todos los archivos que se introduzcan en el bucket. Esta notificación sucede cuando se carga el archivo con el texto extraído, pero se descarta por la función lambda.

2. Desencadenación de documentAI-extract-text.

   Una vez que ha recibido la notificación por parte del bucket Cloud Storage, esta función se activa. Recibe el nombre del bucket y el archivo que se ha cargado. A continuación, utiliza el servicio Cloud Document AI para iniciar la extracción de texto. Posteriormente guarda el archivo con el texto extraído dentro del bucket.

   A diferencia de AWS, no es posible obtener el ID de un proceso para continuar la extracción mediante otra función lambda. Sin embargo, al utilizar métodos asíncronos, no es inconveniente para poder tener varios archivos simultáneamente realizando el proceso de extracción.

3. Notificación Cloud Tasks

   Tras cargar el archivo con el texto extraído, utiliza el servicio de Cloud Tasks para notificar a la función analyze-text que puede continuar con el proceso del analisis de texto. Esto activa la función que recibirá el ID del objeto con el texto extraído por parte de Cloud Tasks.

4. Desencadenación de analyze-text.

   Esta función recibe la notificación con el ID del objeto, obtiene el texto del bucket y comienza su análisis. A diferencia de AWS, no permite realizar una correcta extracción en ensayos con textos en columnas.

5. Traducción del texto

   La función analyze-text utiliza el servicio Cloud Language para detectar el idioma del texto extraído. En caso de que el idioma sea distinto al español, llama al servicio Cloud Translate para traducirlo, por lo que este paso es opcional.

6. Carga de los datos

   Una vez la función analyze-text ha terminado la tarea de procesamiento del text, carga el archivo en formato txt con el texto extraído dentro del bucket y la carpeta *resultados-extraídos/*.

Sección 2: Interacción del Chatbot

7. DialogFlow-integration

   Esta función es la que maneja toda la sección 2. Cada vez que Cloud DialogFlow detecta un intent, llama a la función Cloud Functions quien determina cómo seguir. La función interactúa con Cloud Firestore para obtener las respuestas almacenadas en la base de datos y las proporciona al usuario a través del chatbot.

   El funcionamiento es exactamente el mismo explicado en la sección de AWS 5.4

5.6. Comparativa de servicios

En este capítulo, se comparan las características, funcionalidades y experiencias obtenidas durante el desarrollo de los chatbots.

- **Detección de Intenciones:** Durante las pruebas, se observó que ambos chatbots podían proporcionar respuestas adecuadas a las mismas preguntas. Sin embargo, se notaron diferencias en la detección de intenciones. En AWS, la detección de intents requería una mayor cantidad de ejemplos debido a que el servicio de AWS Lex necesitaba una mayor cantidad de datos de entrenamiento para mejorar su capacidad de comprensión. Dialogflow demostró una mayor eficacia en la detección de intents con menos ejemplos.

- **Problemas con Cloud Functions:** Una limitación importante encontrada en GCP fue el manejo de errores en Cloud Functions. Si ocurría un error durante la compilación de una función, el código se perdía y desaparecía del entorno, lo que obligaba a volver a cargarlo. Esto presentó un desafío ya que, sobre todo en los primeros casos, se perdía el código ya escrito, siendo necesario el volver a escribirlo.

- **Gestión de Roles y Permisos:** En términos de gestión de roles y permisos, AWS y GCP utilizan enfoques diferentes. Los servicios de AWS interactúan entre ellos estableciendo permisos mediante IAM, mientras que GCP utiliza la activación de APIs dentro de un mismo proyecto para manejar estos permisos.

- **Costes de los Servicios:** En cuanto a los costos, ambos servicios demostraron ser bastante económicos. Tanto AWS como GCP ofrecen modelos de precios basados en el uso, lo que permite a las empresas escalar sus operaciones sin incurrir en costos fijos significativos. En ambos casos, los costos operativos fueron mínimos, lo que hace que estos servicios sean accesibles incluso para pequeñas y medianas empresas.

En la siguiente tabala se introduce una comparativa de los costos por servicio en AWs y GCP 5.2 y 5.1:

A través de la siguiente tabla 5.3, se puede ver la comparativa de servicios que proporciona GCP y que se ha utilizado a lo largo del proyecto [6].

5.7. Despliegue de los Chatbots

Para el despliegue de los chatbots se ha habilitado una página pública donde cualquier persona pueda acceder y probarlos. Está alojada mediante el servicio de AWS S3. Se valoró la opción de usar otro servicio llamado AWS Amplify que también permite alojar páginas web [40]. AWS S3 también

Servicio GCP	**Coste por Solicitud/Unidad/Almacenamiento**
Cloud Functions	$0.40 USD por millón de solicitudes (primeros 2 millones gratis)
Document AI	$1.50 USD por 1000 páginas
Cloud Storage	• $0.023 USD por GB al mes • $0.005 USD por cada 1000 operaciones
Cloud Tasks	Primer millón gratis al mes
Natural Language API	$0.0020 USD por unidad (1000 caracteres)
Translation API	$20.00 USD por millón de caracteres
Dialogflow	$0.002 USD por solicitud de API
Firestore	• $0.18 USD por GB al mes • $0.02 USD por 100,000 operaciones de lectura • $0.18 USD por 100,000 operaciones de escritura • $0.02 USD por 100,000 operaciones de eliminación

Tabla 5.1: Costes por Solicitud/Unidad/Almacenamiento de los servicios GCP utilizados en el proyecto

permite actuar como lugar de alojamiento estático para sitios web, por lo que cualquiera de las dos opciones es válida.

Me encontré con un problema a la hora de desplegar el chatbot de AWS. Según indica la documentación, es necesario crear una nueva versión desde la versión borrador así como un *alias* asignado a la versión para poder desplegarlo. Sin embargo, no he podido de ninguna de las tres formas que he intentado:

La primera ha sido mediante una interfaz gráfica para el bot que proporciona AWS [42]. La forma de desplegarlo indica que es muy sencilla, ya que esta misma crea una pila y gestiona los accesos para el bot. A la hora de probarlo cuando ya la aplicación lo ha desplegado, el bot no se conectaba con mis servicios. He podido leer en foros de GitHub que se podía deber a la región donde está desarrollado el entorno [15], por lo que procedí a cambiar todo el entorno de la región de Londres a la de Norte de Virginia, tampoco obtuve un resultado satisfactorio.

La otra opción fue configurar yo mismo a mano los pasos que se realizan en la opción anterior. Esto implica incluir los SDK de AWS y Cognito, el servicio que se encarga de gestionar los roles de acceso para usuarios ajenos, pudiendo así hacer uso del bot únicamente para lectura por cualquier usuario.

Servicio AWS	**Coste por Solicitud/Unidad/Almacenamiento**
AWS Lambda	$0.20 USD por millón de solicitudes
Amazon S3	• $0.00045 USD por 1000 solicitudes • $0.023 USD por GB para los primeros 50 TB/mes
Amazon Textract	$1.50 USD por 1000 páginas
Amazon SNS	$0.60 USD por millón de notificaciones
Amazon Comprehend	$0.0001 USD por unidad (100 caracteres)
Amazon Lex	$0.75 USD por 1000 solicitudes
Amazon DynamoDB	• $1.25 USD por millón de escrituras • $0.25 USD por millón de lecturas
Amazon Translate	$15.00 USD por millón de caracteres

Tabla 5.2: Costes por Solicitud/Unidad/Almacenamiento de los servicios AWS utilizados en el proyecto

Servicio	**AWS**	**GCP**
Almacenamiento	Amazon S3	Cloud Storage
Computación Serverless	Amazon Lambda	Cloud Functions
Base de Datos NoSQL	Amazon DynamoDB	Cloud Firestore
Procesamiento de Texto	Amazon Textract	Cloud Document AI
Notificaciones	Amazon SNS	Cloud Tasks
Traducción	Amazon Translate	Cloud Translation
NLP	Amazon Comprehend	Cloud Natural Language
Bots	Amazon Lex	Dialogflow

Tabla 5.3: Comparativa de Servicios AWS y GCP

También desarrollé el código JavaScript para hacer la llamada al servidor de AWS. No fue posible tampoco conectarse con el bot.

La última opción fue utilizar el servicio de Kommunicate [23], el cual es un producto que permite integrar en aplicaciones web y móviles multitud de servicios de manera sencilla. Dispone de una prueba gratuita de 30 días. Para integrarlo, es necesario crear un rol IAM para usuarios, con permiso de acceso a Amazon Lex. Este rol proporciona un ID y una clave secreta que necesitará Kommunicate para acceder al bot. Por último, pide seleccionar

el alias del bot. Un alias en AWS Lex es una etiqueta que apunta a una versión específica del bot.

En esta última parte descubrí el problema: el alias del bot. Volví a seleccionar sin éxito el Alias que apuntaba a la nueva versión. Sin embargo, al seleccionar el Alias borrador con el que he estado trabajando, pude por fin hacer uso del bot sin problema alguno. Seleccionar el Alias borrador es algo que no era posible en ninguna de las otras dos opciones.

Una vez el bot desplegado, Kommunicate proporciona un código script para introducirlo en la página web, pudiendo así integrarse en la parte inferior derecha de la pantalla como un desplegable[9].

En el caso de GCP, el proceso fue mucho más sencillo, pudiendo hacer estos mismos pasos desde la consola de dialogflow- en el apartado de integraciones -para obtener el código script y también así alojarlo en la página web.

5.8. Análisis de Costos y Beneficios

El uso de chatbots puede representar un ahorro significativo en costos para las empresas. A continuación, se detallan los principales aspectos que contribuyen a este ahorro:

Reducción de Costos Operativos

La implementación de chatbots reduce la necesidad de personal para tareas repetitivas de atención al cliente. Esto permite a las empresas disminuir sus costos operativos y redistribuir recursos humanos hacia tareas de mayor valor agregado. De acuerdo con un informe de IBM, los chatbots pueden reducir los costos de atención al cliente hasta en un 30 % [20].

Disponibilidad 24/7

A diferencia de los empleados humanos, los chatbots pueden operar las 24 horas del día, los 7 días de la semana. Esto mejora la disponibilidad del servicio y reduce la necesidad de turnos nocturnos o extras, generando un ahorro adicional.

Escalabilidad

Los chatbots permiten una fácil escalabilidad en función del volumen de interacciones con los clientes. Las empresas pueden ajustar su capacidad

de respuesta sin necesidad de incurrir en costos adicionales de contratación nuevo personal.

Costos de Infraestructura

Utilizar servicios en la nube como AWS y Google Cloud reduce la necesidad de invertir en infraestructura física, mantenimiento y actualización de hardware. Estos servicios ofrecen modelos de pago por uso, lo que permite a las empresas adaptar sus gastos según la demanda real. De esta forma, la infrastructura real no se ve desaprovechada en momentos con poca demanda ni insuficiente en momentos con mucha demanda.

Comparativa de Costos

Inicialmente, el desarrollo e implementación de chatbots puede representar una inversión significativa para las empresas. Sin embargo, con el tiempo, se observa un considerable ahorro en mano de obra y costos operativos. De acuerdo con un informe de McKinsey Global Institute, la automatización basada en inteligencia artificial podría liberar hasta el 20 % del tiempo de trabajo en empleos de conocimiento, lo que equivale a $5 billones en costos laborales a nivel mundial en 2020 [21].

Mantenimiento Predictivo

La inteligencia artificial también puede ayudar a predecir y prevenir fallos en maquinaria y equipos, lo que reduce los costos de mantenimiento. Según un informe de Deloitte, el mantenimiento predictivo basado en AI puede reducir los costos de mantenimiento en un 10 % y aumentar la vida útil de los activos en un 20 % [10].

Casos de Éxito

- **Bank of America**: La implementación de chatbots basados en AI ha permitido reducir los costos de atención al cliente en más de $100 millones al año [29].
- **Sephora**: La empresa de cosméticos y maquillaje implementó un chatbot para interactuar con clientes y ofrecer recomendaciones personalizadas, resultando en un aumento del 11 % en la compra de pedidos, reduciendo los pasos necesarios para reservar una cita y multiplicando por 5 las ventas [39].

- **KLM Royal Dutch Airlines**: KLM utilizó un chatbot para manejar hasta el 50 % de las consultas de los clientes, reduciendo significativamente los costos operativos y aumentando la satisfacción del cliente [1].
- **Domino's Pizza**: Implementó un chatbot para facilitar los pedidos, reduciendo el tiempo promedio de pedido en un 30 %, aumentando así la eficiencia en otro 30 %. Este nuevo bot ahora maneja el 70 % de las consultas. Por otro lado, los costos operativos relacionados con atención al cliente se redujeron un 20 %[35].

6. Trabajos relacionados

En el ámbito del desarrollo de chatbots, existen numerosos trabajos que se centran en la creación de soluciones sencillas utilizando Lex o DialogFlow. Sin embargo, muchos de estos trabajos se enfocan en un subconjunto específico de servicios y no integran de manera completa todas las capacidades disponibles en AWS. Esta fragmentación de soluciones ha llevado a una necesidad de agrupar y combinar varias implementaciones para lograr un sistema robusto y completo, como el desarrollado en este trabajo.

Por ejemplo, un estudio sobre la creación de un asistente virtual interactivo para la programación utilizando Amazon Lex y Lambda proporciona una pequeña guía sobre cómo construir un chatbot con estos servicios. Sin embargo, este trabajo no integra otros servicios esenciales como Amazon Textract, Amazon Comprehend, o Amazon S3 [38].

Otro ejemplo, diría que el más relevante, ha sido el proyecto de github sobre la extracción de *insights* de facturas con Amazon Textract, Amazon Comprehend y Amazon Lex, muestra cómo utilizar estos servicios específicos para automatizar el procesamiento de datos textuales y descubrir insights a partir de ellos [30]. Si bien este proyecto integra un mayor número de servicios para el chatbot, su funcionalidad es muy limitada proporcionando únicamente datos relevantes de facturas.

Sin embargo, la documentación proporcionada por AWS ha sido lo suficientemente detallada para guiarme en la creación e integración de todos estos servicios [41]. Los tutoriales y guías oficiales de AWS ofrecen ejemplos claros y pasos detallados para implementar cada servicio.

Por el otro lado, el caso de GCP ha sido muy similar. Sin embargo, el principal problema ha sido la complejidad de la documentación que proporciona google, ya que tiende a ser más compleja y extensa [5]. A pesar

de este obstáculo, una vez que se comprendió el funcionamiento de los servicios de AWS, la transición y aplicación de conocimientos a GCP resultó ser mucho menos complicada.

Por ejemplo, un trabajo sobre la creación de un chatbot utilizando Dialogflow y Google Cloud Functions sería el siguiente, donde se explica la creación y configuración de Dialogflow, muy similar a Amazon Lex [26].

7. Conclusiones y Líneas de trabajo futuras

7.1. Conclusiones

A lo largo de este proyecto, se ha llevado a cabo el desarrollo de dos chatbots utilizando los servicios en la nube de AWS y GCP. Estos chatbots integran una variedad de servicios avanzados como Amazon Lex, Lambda, S3, Textract, Comprehend, Translate, y DynamoDB en el caso de AWS, y Dialogflow, Cloud Functions, Cloud Storage, y otros servicios en el caso de GCP.

Por otro lado, se ha creado una aplicación web de acceso abierto donde poder desplegar los chatbots, permitiendo que cualquier usuario pueda acceder y utilizarlos.

Durante el desarrollo de este proyecto, se ha adquirido un profundo conocimiento sobre la arquitectura de microservicios, la integración de múltiples servicios en la nube, y las mejores prácticas para el desarrollo y despliegue de chatbots. Además, se han explorado diversos servicios y técnicas como el NLP, el análisis y extracción de texto, la traducción automática, y la orquestación de tareas. Esta última siendo muy importante dentro de este ámbito dada la necesidad de coordinar y gestionar múltiples tareas o servicios que deben trabajar juntos para completar un flujo de trabajo complejo, controlando el orden en que se ejecutan estas tareas, manejando las dependencias entre ellas y asegurando que los datos necesarios estén disponibles en el momento preciso.

Finalmente, frente a las dificultades encontradas, hay que expresar agradecimiento por el respaldo del tutor. Sin su apoyo y orientación, la realización

de este proyecto no habría sido posible en las mismas condiciones de éxito y profundidad alcanzadas.

7.2. Líneas de trabajo futuras

En esta sección se proponen posibles direcciones para continuar avanzando en la comprensión del tema en cuestión y en su aplicación práctica.

1. Creación de Nuevos Intents

   Una de las expansiones más efectivas sería la creación de nuevos intents en los chatbots. Estos nuevos intents permitirían abarcar un mayor número de preguntas y respuestas. Por ejemplo, se pueden añadir intents específicos para un mayor tipo de preguntas, consultas técnicas y preguntas contextuales que requieren respuestas detalladas basadas en otros datos. El abanico de respuestas a distintas preguntas es infinito.

2. Integración de un Mayor Número de Servicios

   Servicios como Amazon Rekognition para análisis de imágenes, Amazon Polly para convertir texto en voz, y AWS Glue para la integración y preparación de datos podrían incorporarse para ofrecer funcionalidades más complejas. Por ejemplo, un chatbot que no solo responde preguntas textuales, sino que también puede analizar y responder a partir de imágenes enviadas por los usuarios. El equivalente en en GCP sería Vision AI, text-to-speech y Dataplex.

3. Mejora de la Lógica en Funciones Lambda

   Actualmente la lógica de las funciones implica sacar la información de los pasos de creación del bot desde varios documentos. Se podría complejizar la lógica de estas funciones para que se pudiera extraer desde un documento todos y cada uno de los pasos.

4. Implementación de Aprendizaje Automático

   La incorporación de modelos de aprendizaje automático avanzados podría mejorar la capacidad del chatbot para entender y responder preguntas de manera más precisa. Utilizar servicios como Amazon SageMaker o google Vertex AI para entrenar y desplegar modelos personalizados de ML podría permitir al chatbot aprender de las interacciones pasadas.

Bibliografía

[1] KLM Royal Dutch Airlines. Klm's bluebot chatbot: Revolutionizing customer service with ai. 2020. https://www.altoros.com/blog/klm-handles-2x-more-customer-requests-with-artificial-intelligence/.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

[3] Ned Batchelder. *Coverage*. https://coverage.readthedocs.io/en/7.4.4/, 2024. Online; Accedido el 09-May-2024.

[4] Pascal Brachet. *Texmaker*. https://www.xm1math.net/texmaker/, 2024. Online; Accedido el 20-Abr-2024.

[5] Google Cloud. Google cloud documentation. Online; Accedido el 22-Abr-2024.

[6] Google Cloud. Aws, azure y google cloud: Comparación de servicios, 2024. Online; Accedido el 22-Abr-2024.

[7] McKinsey & Company. How bots, algorithms, and artificial intelligence are reshaping the future of corporate support functions.

[8] Ian Stapleton Cordasco. *Flake8*. https://flake8.pycqa.org/en/latest/, 2024. Online; Accedido el 09-May-2024.

[9] Cumulus Cycles. Amazon lex deployment video, 2020. https://www.youtube.com/watch?v=cI1NAjLE_I8&ab_channel=CumulusCycles.

[10] Deloitte. Predictive maintenance with ai: Reducing costs and increasing asset life. 2020. https://www2.deloitte.com/us/en/pages/consulting/articles/using-ai-in-predictive-maintenance.html.

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2019.

[12] Gartner. Magic quadrant for enterprise conversational ai platforms.

[13] Git. *Git*. https://git-scm.com/, 2024. Online; Accedido el 12-Abr-2024.

[14] GitHub. Github desktop. https://github.com/desktop. Online; Accedido el 12-Abr-2024.

[15] GitHub. Issue 71: Aws lex web ui. https://github.com/aws-samples/aws-lex-web-ui/issues/71.

[16] GitHub. *GitHub*. https://github.com/, 2024. Online; Accedido el 12-Abr-2024.

[17] GitLab. *GitLab*. https://about.gitlab.com/, 2024. Online; Accedido el 12-Abr-2024.

[18] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[19] Sublime HQ. *Sublime Text*. https://www.sublimetext.com/, 2024. Online; Accedido el 19-Abr-2024.

[20] IBM. How ai-powered chatbots can reduce customer service costs. 2020. https://www.ibm.com/watson/ai-customer-service-smartpaper/#:~:text=IBM%20Watson%20Assistant%20brings%20the,agents%20that%20are%20available%2024x7.

[21] McKinsey Global Institute. The future of work: Automation, ai, and digital transformation. 2020. https://www.mckinsey.com/mgi/overview/in-the-news/automation-and-the-future-of-work.

[22] Daniel Jurafsky and James H Martin. *Speech and Language Processing*. Pearson, 2020.

[23] Kommunicate. Amazon lex integration. https://www.kommunicate.io/product/amazon-lex-integration.

[24] Pylint Logilab. *Pylint*. https://www.pylint.org/, 2024. Online; Accedido el 09-May-2024.

[25] Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. The stanford corenlp natural language processing toolkit. *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60, 2014.

[26] Mattermost. How to create a chatbot with dialogflow and google cloud functions. Online; Accedido el 22-Abr-2024.

[27] Microsoft. *Visual Studio Code*. https://code.visualstudio.com/, 2024. Online; Accedido el 19-Abr-2024.

[28] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2013.

[29] Bank of America. Reducing customer service costs with ai chatbots. 2020. https://newsroom.bankofamerica.com/content/newsroom/press-releases/2022/10/bank-of-america-s-erica-tops-1-billion-client-interactions--now-.html.

[30] User on GitHub. Extraction of conversational insights from invoices with amazon textract, amazon comprehend, and amazon lex. Online; Accedido el 22-Abr-2024.

[31] Overleaf. *Overleaf*. https://www.overleaf.com/, 2024. Online; Accedido el 20-Abr-2024.

[32] Bo Pang and Lillian Lee. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1–135, 2008.

[33] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[34] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, 2018.

[35] Domino's Pizza. Domino's ai chatbot: Simplifying pizza ordering and reducing wait times. 2020. https://messagemind.ai/case-studies/dominos/.

[36] pytest-cov contributors. *Pytest-cov*. https://pytest-cov.readthedocs.io/en/latest/readme.html, 2024. Online; Accedido el 09-May-2024.

[37] Lev Ratinov and Dan Roth. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL)*, pages 147–155, 2009.

[38] Saketh Reddy Regatte. An amazon lex based interactive virtual assistant for c programming. Online; Accedido el 22-Abr-2024.

[39] Sephora. How sephora is transforming customer experience with ai chatbots. 2020. https://www.cut-the-saas.com/ai/beauty-and-the-bot-how-sephora-reimagined-customer-experience-with-ai.

[40] Amazon Web Services. Aws amplify. https://us-east-1.console.aws.amazon.com/amplify/apps.

[41] Amazon Web Services. Aws documentation. Online; Accedido el 22-Abr-2024.

[42] Amazon Web Services. Deploy a web ui for your chatbot. https://aws.amazon.com/es/blogs/machine-learning/deploy-a-web-ui-for-your-chatbot/.

[43] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[44] Wikipedia. *CSS*. https://es.wikipedia.org/wiki/CSS, 2024. Online; Accedido el 07-Jun-2024.

[45] Wikipedia. *HTML*. https://es.wikipedia.org/wiki/HTML, 2024. Online; Accedido el 07-Jun-2024.