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 — 30 de junio de 2024
Tutor: Dr. Jose Manuel

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.

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 % (McKinsey, 2020).

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 (Gartner, 2022).

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:** estado del arte en las aplicaciones y sitios web de bolsa y finanzas.
- **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:

- **todo**
  todo

1.2. Materiales adjuntos

Los materiales que se adjuntan con la memoria son:

- **Anexos**: consultar la documentación técnica.

Además, los siguientes recursos están accesibles a través de internet:

- **Repositorio**: visitar 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 Google Cloud Platform.
- 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 más destacables 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.

- **Tokenización**: El proceso de dividir el texto en unidades más pequeñas llamadas tokens (generalmente palabras o frases). [4].
- **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 [5].
- **Reconocimiento de Entidades Nombradas (NER)**: Identificar y clasificar entidades mencionadas en el texto, como nombres de personas, organizaciones, lugares, fechas, etc. [10].
- **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 [4].

- **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 [7].
- **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 mejorar la precisión y fluidez de las traducciones [1].

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)**: Especialmente útiles para datos secuenciales, como texto y habla. Las RNNs pueden recordar información a lo largo de las secuencias de datos[3].
- **Long Short-Term Memory (LSTM)**: Una variante de las RNNs que aborda el problema del gradiente desaparecido, permitiendo a los modelos capturar dependencias a largo plazo en los datos de secuencia [3].
- **Transformadores:** Introducidos por Vaswani et al. (2017), los transformadores han demostrado ser altamente efectivos en una amplia gama de tareas de NLP. Estos modelos utilizan 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 claros ejemplos de estos modelos [2, 11].
- **Word Embeddings:** Técnicas como Word2Vec y GloVe transforman palabras en vectores numéricos de alta dimensión que capturan semánticas y relaciones sintácticas [6, 8].
- **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. [2, 9].

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 procesamiento del lenguaje natural (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.

4. Técnicas y herramientas

4.1. Técnicas metodológicas

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.

4.2. Control de versiones

- Herramientas consideradas: Git [**?**], GitHub Desktop [**?**]
- 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.

4.3. Alojamiento del repositorio

- Herramientas consideradas: GitHub [**?**], GitLab [**?**]
- 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.

4.5. Entorno de desarrollo integrado (IDE)

- Herramientas consideradas: Sublime Text [**?**], Visual Studio Code [**?**].
- 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. Documentación de la memoria

- Herramientas consideradas: Texmaker [**?**] y Overleaf [**?**].
- 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.

4.7. Calidad y consistencia de código

- Herramientas consideradas: Pylint [**?**] y Flake8 [**?**].
- 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.

4.8. Cobertura de código

- Herramientas consideradas: Coverage [**?**] y Pytest-cov [**?**].
- Herramienta elegida: Coverage.

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.

4.9. Desarrollo web

HTML

HTML [**?**] 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 [**?**] 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

Básicamente como he hecho las cosas, por que las he hecho y en que orden. Que me costó mas de mi aprendizaje, menos, por que fueron distintos, cuales son equivalentes y conceptos que no son lo mismo realmente. Documentar por qué configuro, que configuro, que fin tiene documentar la otra parte.

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 Dr., 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.

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 recoger las respuestas 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 ha sido la de 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 por encima.
TODO insertar referencias.

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

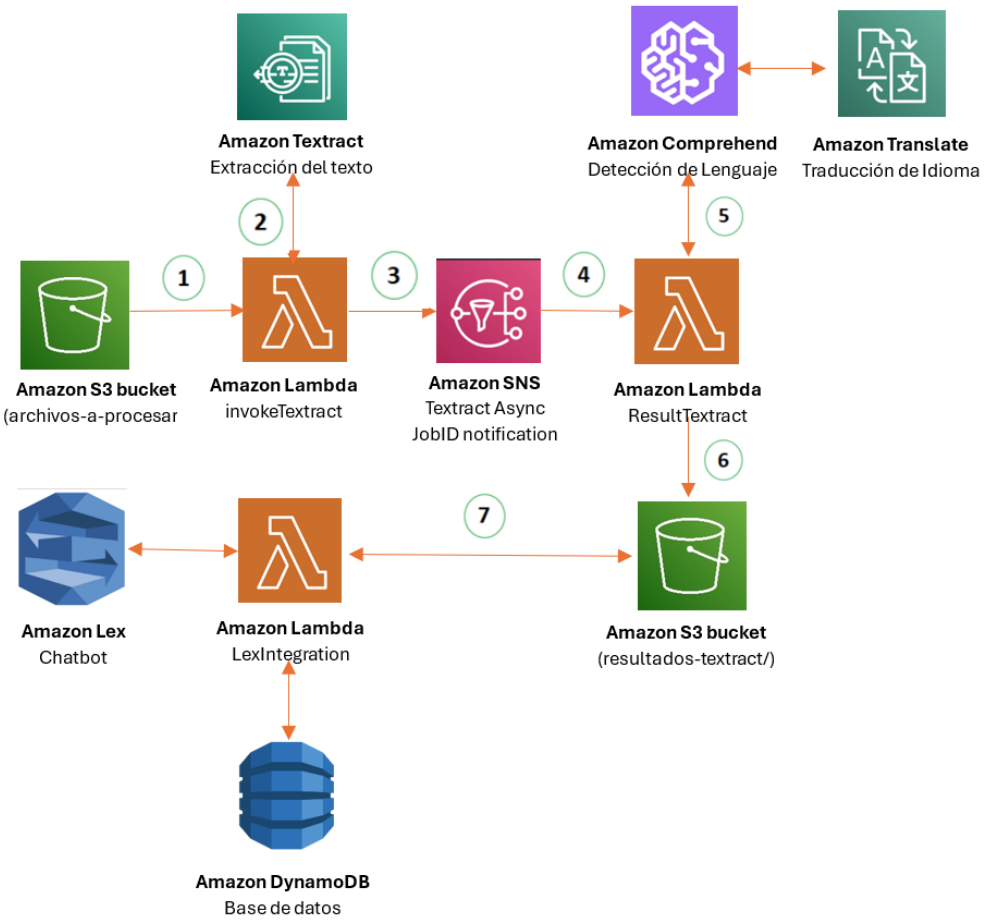

Figura 5.1: Ecosistema AWS. Fuente: elaboración propia

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. 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 (NLP) 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 usado por la función ResultTextract para detectar el idioma 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. 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 (NLU)
   TODO poner referencia. para entender la intención del usuario y gestionar diálogos de manera dinámica.

   Este desarrolla el 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 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 Lex (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 se encarga únicamente de la tarea de extracción del texto.
- Sección 2: Interacción del Chatbot: No es necesario que haya texto extraído dentro del bucket para funcionar, ya que actúa independientemente mediante la base de datos.

A continuación se detalla cada paso del proceso:

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.

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, comienza la extracción del texto.
   TODO introducir diagrama de flujo y explicar mejor

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 preprocesamiento de datos, 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.
TODO continuar explicando

Montaje

En esta sección, se detallará cómo se ha montado todo el entorno para el desarrollo del chatbot utilizando los servicios de AWS.

Creación del Rol IAM

Para comenzar, es necesario configurar un Rol IAM en AWS. Este rol define los permisos que las funciones Lambda necesitarán para interactuar con otros servicios de AWS.

Primero, busca el servicio Identity and Access Management (IAM) y crea un nuevo rol en la sección Roles. Yo le he puesto el nombre "AWS_PDF_Textract_Lex_Role". En el Paso 1, configura el tipo de entidad de confianza como Servicio de AWS, con el caso de uso Lambda. En el Paso 2, asigna los siguientes permisos: `AmazonS3FullAccess`, `AmazonTextractFullAccess`, `AWSLambdaExecute`, `AWSLambdaSNSFull` `ComprehendFullAccess` y `TranslateFullAccess`. En el Paso 3, elige un nombre para el rol y revisa el resumen de configuración. Finalmente, crea el rol.

Repite estos pasos para crear otro rol con los siguientes permisos, en mi caso lo llamé "AWS_Lambda_LexIntegration": `AmazonDynamoDBFullAccess`, `AmazonS3FullAccess`, `AWSLambdaBasicExecutionRole` y `AmazonLexFullAccess`.

Creación de un Bucket S3 con dos Carpetas

El siguiente paso es crear un bucket en Amazon S3 para almacenar los archivos que se procesarán y los resultados. Busca el servicio Amazon S3 y crea un nuevo bucket, eligiendo un nombre y dejando todas las configuraciones por defecto para mayor seguridad. Dentro de este bucket, crea dos carpetas: una para cargar los PDF y otra para almacenar el texto extraído de los PDF en formato TXT.

Crear un SNS

Para la mensajería y las notificaciones, utiliza Amazon Simple Notification Service (SNS). Busca el servicio SNS y crea un nuevo tópico, asignándole un nombre y seleccionando el tipo estándar. Deja todas las configuraciones por defecto y crea el tópico.

Creación de las Funciones Lambda

En este paso, se crearán tres funciones Lambda que interactuarán con S3, Textract y Lex.

Primero, busca el servicio AWS Lambda y crea una nueva función desde cero. Asigna un nombre a la función y selecciona Python como el lenguaje de tiempo de ejecución, con arquitectura x86_64. En la sección de permisos, añade el rol IAM "AWS_PDF_Textract_Lex_Role". Después de crear la función, ve a la configuración básica y ajusta la memoria a 1GiB, el almacenamiento efímero y el tiempo de espera a 15 minutos. Configura un desencadenador (trigger) para el bucket S3 que hemos creado, especificando el prefijo de la carpeta donde se cargarán los PDF y el sufijo `.pdf`.

La segunda función Lambda será similar a la primera, pero el desencadenador será el servicio SNS configurado anteriormente. Utiliza el mismo rol IAM que la primera función.

La tercera función Lambda no tendrá un desencadenador y puedes usar la misma configuración básica de las funciones anteriores, exceptuando el rol, que será el "AWS_Lambda_LexIntegration".

Creación de la Base de Datos

Ahora vamos a configurar la base de datos donde el chatbot buscará respuestas a las preguntas. Busca el servicio Amazon DynamoDB y crea una nueva tabla. Introduce el nombre de la tabla y establece la clave de partición como `IntentName`. Esto permitirá recuperar el conjunto de preguntas asociadas a cada intent y proporcionar respuestas precisas. Deja el resto de las configuraciones por defecto y crea la tabla.

Después, crea una función Lambda para cargar los datos en la tabla DynamoDB. El código necesario para esta función se puede encontrar en un enlace de GitHub especificado. Una vez creada la función, realiza una prueba para asegurarte de que los datos se cargan correctamente.

Creación del Bot

Para configurar el chatbot, accede al servicio AWS Lex y crea un nuevo bot desde cero. Asigna un nombre al bot y deja la configuración de permisos predeterminada, ya que las funciones Lambda previamente creadas serán las que interactúen con los servicios necesarios.

Crea una lista de intents y añade ejemplos de enunciados para cada uno. Configura los hooks de código utilizando las funciones Lambda creadas

anteriormente. Los intents iniciales que debes crear incluyen `StartTutorial`, `RepeatStep`, `NextStep` y `GoToStep`. Añade frases de entrenamiento como "crear un chatbot" o "siguiente paso".

A continuación, configura los intents correspondientes al banco de preguntas. Asegúrate de utilizar los nombres de intents que coincidan con los utilizados en la función Lambda que carga los datos en DynamoDB, para garantizar la correcta detección y respuesta. Añade también las frases de entrenamiento similares a las preguntas que has introducido en la base de datos.

Implementación del Chatbot

TODO

Costos y escalabilidad

5.5. Google Cloud

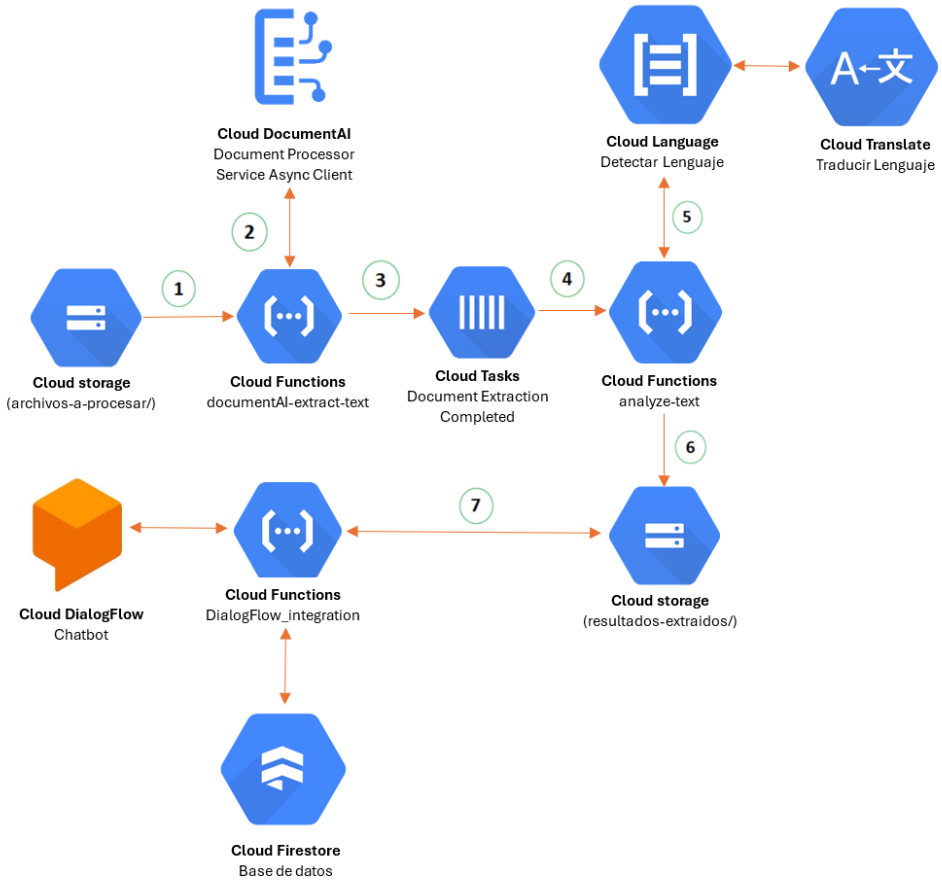

Figura 5.2: Ecosistema GCP. Fuente: elaboración propia

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. 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 (NLP) 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 (NLU) para entender la intención del usuario y gestionar diálogos de manera dinámica.

   Este servicio desarrolla el 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 se encarga únicamente de la tarea de extracción del texto.
- Sección 2: Interacción del Chatbot: No es necesario que haya texto extraído dentro del bucket para funcionar, ya que actúa independientemente mediante la base de datos.

A continuación se detalla cada paso del proceso:

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.

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 y obtener el ID del proceso.

3. **Notificación Cloud Tasks**

   Tras obtener el ID, utiliza el servicio de Cloud Tasks para notificar a la función analyze-text que puede continuar con la extracción del texto. Esto activa la función que recibirá el ID por parte de Cloud Tasks.

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

4. **Desencadenación de analyze-text**.

   Esta función recibe la notificación con el ID, comienza la extracción del texto utilizando Cloud Document AI.

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 toda la tarea de pre-procesamiento de datos, 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.

   Esta integración permite que el chatbot responda preguntas utilizando la información preprocesada y almacenada, asegurando que las respuestas sean precisas y relevantes, independientemente de si hay texto recién extraído en el bucket.

Montaje

Para comenzar a trabajar con los servicios de Google Cloud y crear el entorno del chatbot, es necesario seguir una serie de pasos detallados a continuación.

Creación del Proyecto

Lo primero que se debe hacer es crear un proyecto nuevo en la consola de Google Cloud. Al acceder a la consola, se presenta la opción de crear un nuevo proyecto. Es importante guardar el ID del proyecto, ya que puede ser necesario más adelante para configurar el entorno virtual.

Paso 1: Creación del Storage

El primer paso es crear un bucket en el servicio Cloud Storage. Se debe configurar el nombre del bucket y la región deseada, dejando el resto de las configuraciones en su estado predeterminado. Una vez creado el bucket, se deben crear dos carpetas dentro de él: una para los archivos PDF que se desean procesar y otra para almacenar los archivos TXT con el texto extraído.

Paso 2: Creación de Document AI

A continuación, se configura el servicio Document AI. Al acceder a este servicio, se solicita habilitar la API correspondiente. Después, se procede a crear un procesador, eligiendo el tipo Document OCR para realizar la extracción de texto. Se le asigna un nombre y una región al procesador. Es

recomendable mantener la misma región para todos los servicios relacionados con el proyecto. Una vez configurado, se puede probar subiendo un documento de prueba para verificar su correcto funcionamiento.

Paso 3: Creación de las Cloud Functions

En este paso, se crean tres funciones Cloud Functions, cada una con un propósito específico: extraer texto, procesar el texto y gestionar la interacción con DialogFlow.

La primera función se encarga de la extracción de texto. Se selecciona el entorno de 1ª generación por su estabilidad. Se configura el nombre y la región, y se establece como activador Cloud Storage con tipo de evento "finalized". Se ajustan los recursos de memoria, CPU y tiempo de espera para asegurar la finalización de la tarea. También se asigna la cuenta de servicio del proyecto.

Antes de configurar las otras funciones, se crea una cola en el servicio Cloud Tasks para gestionar las notificaciones. Se habilita la API de Cloud Tasks y se configura el nombre y la región de la cola.

La segunda función se activa mediante una notificación HTTP enviada por la primera función. Esta función realiza la tarea de procesar el texto, para lo cual necesita habilitar las API de Cloud Natural Language y Cloud Translate.

La tercera función se encarga de la integración con el bot de DialogFlow. Esta función se configura de manera similar a las anteriores, pero con el propósito específico de gestionar la interacción con DialogFlow.

Paso 4: Creación de la Base de Datos

Para almacenar el banco de preguntas y respuestas del chatbot, se utiliza Firestore. Se selecciona el modo Nativo y se asigna un nombre, dejando el resto de configuraciones por defecto. Una vez creada la base de datos, se añaden dos colecciones: una para el banco de preguntas y respuestas (`chatbotresponses`) y otra para los pasos de creación del chatbot (`chatbotsteps`).

Paso 5: Configuración de DialogFlow ES

Se utiliza DialogFlow ES (Essentials) para configurar el bot. En la consola de DialogFlow, se crean los intents necesarios, añadiendo frases de

entrenamiento y habilitando la opción .[En]able webhook call for this intent.[en] el fulfilment.

Los intents iniciales incluyen `StartTutorial`, `RepeatStep`, `NextStep` y `GoToStep`, con frases de entrenamiento como "crear un chatbot" o "siguiente paso". También se configuran los intents correspondientes al banco de preguntas, utilizando los nombres de intents que recibe la Cloud Function y añadiendo frases de entrenamiento similares a las preguntas almacenadas en la base de datos.

Paso 6: Implementación del Chatbot

Finalmente, se implementa el chatbot en una página web. En la consola de DialogFlow, se navega a la configuración del bot y se añade una descripción. En el apartado de "Integrations" -> "Web demo", se copia el código `<iframe>` y se inserta en la página web.

Costos y escalabilidad

5.6. Creación de la aplicación web

5.7. Comparativa de servicios

Problemas encontrados, soluciones, comparativa de respuestas?, cual me resultó mas facil, diferencias, Experiencia del Usuario (UX)

6. Trabajos relacionados

7. Conclusiones y Líneas de trabajo futuras

7.1. Conclusiones

7.2. Líneas de trabajo futuras

Bibliografía

[1] 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.

[2] 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.

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

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

[5] 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.

[6] 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.

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

[8] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Confe-*

rence on Empirical Methods in Natural Language Processing (EMNLP), pages 1532–1543, 2014.

[9] 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.

[10] 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.

[11] 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.