MCDI

MAESTRÍA EN CIENCIA DE DATOS E INFORMACIÓN

INFOTEC CENTRO DE INVESTIGACIÓN E INNOVACIÓN EN TECNOLOGÍAS DE LA INFORMACIÓN Y COMUNICACIÓN

DIRECCIÓN ADJUNTA DE INNOVACIÓN Y CONOCIMIENTO
GERENCIA DE CAPITAL HUMANO
POSGRADOS

“ChatBot de dominio específico para atención a usuarios”

Implementación de proyecto laboral
que para obtener el grado de MAESTRO EN CIENCIA DE DATOS E INFORMACIÓN

Presenta:

Lic. Víctor Eduardo Pimentel Avilés

Asesor:

Dr. Sabino Miranda Jiménez

Ciudad de México, Agosto, 2022.

CONACYT

INFOTEC POSGRADOS

Autorización de impresión

INFOTEC
POSGRADOS

AUTORIZACIÓN DE IMPRESIÓN Y NO ADEUDO EN BIBLIOTECA

Maestría en Ciencia de Datos e Información, MCDI.

Ciudad de México, 06 de mayo de 2022

La Gerencia de Capital Humano / Gerencia de Investigación hacen constar que el trabajo de titulación intitulado

"ChatBot de dominio específico para atención a usuarios".

Desarrollado por el alumno: ***Víctor Eduardo Pimentel Avilés.*** y bajo la asesoría del **Dr. Sabino Miranda Jiménez** cumple con el formato de Biblioteca. Por lo cual, se expide la presente autorización para impresión del proyecto terminal al que se ha hecho mención.

Asimismo, se hace constar que no debe material de la biblioteca de INFOTEC.

Vo. Bo.

Lic. Juan Ramón Abarca Damián
Coordinador de Biblioteca

Anexar a la presente autorización al inicio de la versión impresa del trabajo referido que ampara la misma.

C.p.p Servicios Escolares

Tabla de contenido

Presentación **8**

Capítulo 1. Introducción **15**

1.1 Planteamiento del Problema 16
1.2 Objetivo General 16
1.3 Objetivos Específicos 16
1.4 Metas 17
1.5 Hitos 17

Capítulo 2. Marco Teórico **20**

2.1 Historia de los chatbots 20
2.2 Machine Learning: Revisión de literatura 23
2.2.1 Words Embeddings 23
2.2.2 Latent Semantic Analysis 27
2.2.2 Recurrent Neural Network 29
2.2.3 Latent Dirichlet Allocation 30

Capítulo 3. Metodología **32**

3.1 Bases de datos 32
3.2 Limpieza y preprocesamiento 34
3.3 Construcción del modelo 36
3.3.1. Vectorización 36
3.3.2 Selección de número de tópicos 37
3.4 Medidas de Similitud 39
3.5 Arquitectura del chatbot 40
3.4.1 Esquematización y funcionamiento de la implementación 42

Capítulo 4. Resultados **46**

4.1 Generación de Tópicos 46
4.2 Desempeño 49
4.3 Evaluación 51
4.4 Tiempos de respuesta de interfaz 54
4.5 Aplicación 56

Conclusiones **62**

Bibliografía **64**

ANEXO **68**

Índice de figuras

Figura 1. Esquema de CBOW ……..……... 25

Figura 2. Esquema de Skip Grams …… 25

Figura 3. Esquema de WordToVec …….…..……. 26

Figura 4. Top 10 compañías con mayores registros …….…… 33

Figura 5. Preprocesamiento ……. 34

Figura 6. Distribución de datos por tópicos (3 clases) ……. 38

Figura 7. Distribución de datos por tópicos (4 clases) ……. 38

Figura 8. Distribución de datos por tópicos (5 clases) ……..... 38

Figura 9. Distribución de datos por tópicos (6 clases) ……...…. 38

Figura 10. Distribución de datos por tópicos (7 clases) …….. 38

Figura 11. Implementación de chatbot ……. 41

Figura 12. Flujo de trabajo de chatbot ……. 42

Figura 13. Definición gráfica de F1 Score ……... 53

Figura 14. Distribución por sexo …… 55

Figura 15. Distribución por edad …… 55

Figura 16. Distribución por idioma ……...…… 56

Figura 17. Satisfacción ……...……...……. 57

Figura 18. Dificultad de la pregunta ..……...…… 57

Figura 19. Utilidad ……...……...…… 58

Índice de cuadros

Cuadro 1. Cronograma de actividades …… 17

Cuadro 2. Información disponible ……...……. 33

Cuadro 3. Preguntas Originales ……..... 35

Cuadro 4. Métricas de desempeño ……... 51

Cuadro 5. Ejemplo de preguntas y respuestas ……. 53

Cuadro 6. Desempeño por tópico ……..... 53

Glosario

“D”

Dataset: conjunto de datos

“K”

Kaggle: una subsidiaria de Google LLC, es una comunidad en línea de científicos de datos y profesionales del aprendizaje automático.

“T”

Tech: Hace referencia al término *technology* y se usa para denotar empresas que están desarrolladas sobre una base tecnológica y que sus actividades pueden estar alojadas en servicios en la nube.

Presentación

El auge de las actividades a través de internet ha provocado una creciente demanda de personal que permita atender todas las solicitudes que los usuarios hacen en medios electrónicos. Dentro de estas podrían encontrarse la solicitud de pagos, ordenes de servicio, obtención de información e incluso agendar eventos. Dicho de otra forma Peters, F menciona que: "*usualmente los usuarios encuentran ciertos problemas contactando al servicio al cliente a través de los medios comunes de comunicación (E-mail, social media, website chat) creando lo que es llamado un ticket*" [23]

Al mismo tiempo, algoritmos avanzados de inteligencia artificial en combinación con machine learning y deep learning han hecho un considerable progreso en la mejora de la calidad del entendimiento del proceso cognitivo; aunado a la creciente capacidad computacional para el manejo de grandes volúmenes de información [3]. Parte considerable de lo que significa el proceso cognitivo es la comunicación, ya sea escrita o verbal, donde la ciencia de datos ha hecho avances importantes a través de técnicas de Análisis de Lenguaje Natural. Surge de esta forma, la pregunta inherente: ¿Es posible construir un sistema inteligente a través de Machine Learning que permite emular la conversación humana?

Mayormente conocidos como chatbots, estas herramientas permiten mostrar una interfaz a un cliente o usuario final que simule la atención de un humano a través de comunicación escrita. Es posible encontrar diferentes tipos de chatbots cuando nos enfocamos en sus objetivos [16]:

1. Question-Answering: el agente necesita proveer respuestas concisas y directas a preguntas que hacen los usuarios basado en conocimiento proveniente de varias fuentes de información y entrenamiento
2. Task Completion: El agente tiene como objetivo cumplir con una serie de tareas específicas. Éstas pueden ir desde la reservación en un restaurante hasta planear un viaje de negocios

3. Social Chat: el agente necesita conversar de manera apropiada con los usuarios, como un humano en la prueba de Turing, además deberá poder ser capaz de proveer recomendaciones.
4. Goal Based:el objetivo del agente es el cumplimiento de una meta en específico. Son diseñados para el cumplimiento de una tarea. Donde la comunicación con el usuario es usualmente muy corta, suficiente para obtener información para el cumplimiento de la actividad. Un ejemplo claro sería un chatbot en una página web que ayude en el servicio al cliente y la atención de las preguntas además del direccionamiento de problemas y solicitudes [24].

Algunas de las limitaciones que tienen este tipo de herramientas están incrustadas básicamente desde su construcción [24]. Es decir, problemas relacionados con el hecho de que el *output* de un chatbot está basado en reglas estándares, los errores gramaticales de los usuarios que hacen uso de la herramienta, la definición preestablecida de los dominios de atención, la ambigüedad de las preguntas o solicitudes, y la estructura misma del lenguaje: además, debe mencionarse incluso que tópicos como el análisis de sentimientos donde los chatbots no son capaces de definir cuál es el sentido de la oración así como la misma exactitud de los modelos con que se entrenan las herramientas amplían el campo de acción en el camino de mejoras a los chatbots.

Capítulo 1
Introducción

Capítulo 1 Introducción

El uso de recursos de inteligencia artificial para la solución de problemas cotidianos e incluso para la mejora de procesos dentro de las empresas y en las organizaciones es cada vez más recurren te. Sobre todo cuando hablamos de tareas fácilmente esquematizables, lo que significa que el conjunto de variables que influyen en el resultado o desarrollo de una actividad permite definir el proceso completo en un conjunto relativamente pequeño de pasos.

Ejemplos claros de lo anterior son herramientas como los filtros de SPAM en los correos electrónicos, la personalización de anuncios en una red social, dispositivos en el hogar que ofrecen información o soluciones a pequeños problemas, el asistente personal que "vive" en nuestro smartphone; así como chatbots, videojuegos, vehículos autónomos y el manejo de agendas personales.

El objetivo del presente es la implementación de uno de estos tipos de herramientas: el chatbot, para su uso en herramientas virtuales de servicio al cliente.

La palabra Chatbot proviene de ChatterBot, que fue un juego de calabozos multiusuario donde la principal tarea era la de responder preguntas a los usuarios para la navegación simulando habilidades conversacionales por simples reglas asignadas. El ChatBot puede definirse como una herramienta de software que tiene como objetivo la simulación de una conversación para el cumplimiento de tareas o la solución de preguntas, según la definición de Bayan Abu: "*son programas de computadoras que pueden interactuar con usuarios utilizando lenguaje natural*" [2]; así mismo, Den Allison menciona que "*los chatbots son agentes conversacionales, entidades artificiales de conversación capaces de imitar la personalidad humana, interactuar, y responder en sentencias para rastrear una conversación en forma significante para el ser humano*"[7]

Con lo anterior, podemos entonces definir al chatbot como una herramienta de conversación que permite mantener un intercambio de información en lenguaje natural, que coadyuva en el cumplimiento de tareas y resolución de preguntas utilizando algoritmos de Inteligencia Artificial.

1.1 Planteamiento del Problema

Con el crecimiento de los servicios brindados a través de internet y la cantidad de usuarios que al mismo tiempo están utilizando herramientas tecnológicas, las empresas tech entendiéndose como aquellas enfocadas en la tecnología y mayormente instaladas en servicios en la nube, necesitan enfocarse en la diferenciación competitiva a través de un mejor servicio al cliente. En este sentido tener un sistema de Peticiones, Quejas, Reclamos y Sugerencias (PQRS) automatizado; es decir, una herramienta que permita conocer las inquietudes y las manifestaciones de los usuarios de forma automática y que al mismo tiempo se convierta en el medio y la solución podría significar un crecimiento sostenido. Adicionalmente y como lo menciona Morgan, B[4]: “los chatbots trabajan a través de aplicaciones de mensajes que la mayoría de los consumidores (clientes) ya tienen en sus teléfonos, lo que los hace el siguiente paso natural en la interacción con los usuarios”

1.2 Objetivo General

Elaborar un Bot para la atención del chat de servicio al cliente enfocado a resolver preguntas de los usuarios por medio de respuestas automáticas.

1.3 Objetivos Específicos

Los objetivos específicos están delimitados en dos secciones: la primera de ellas hace referencia a la construcción las bases de conocimiento o conjuntos de datos para el entrenamiento y evaluación del chatbot y, por otro lado, el funcionamiento mismo del sistema:

1. Conjunto de datos:
    - Recopilación de información de preguntas y respuestas
    - Construcción de conjuntos de datos de trabajo para modelo
2. Modelado y evaluación:
    - Clasificación de solicitudes de información

- Clasificación y ordenamiento de posibles respuestas
- Creación de Interfaz de trabajo.

1.4 Metas

Cuando hablamos de la capacidad de los sistemas de atención a usuarios o clientes debemos tener en cuenta el objetivo de la herramienta. Es decir, la clasificación que reciba entre: Question-Answering, Task-Completion, Social Chat y Goal Based. El objetivo principal es poder ofrecer orientación a usuarios sobre sus requisitos de información, a través de la solución de un problema del estilo pregunta respuesta. Por ello, los resultados esperados de la implementación del chatbot de dominio específico para la atención de usuarios es poder ofrecer una interfaz de consulta a través de servicio web o API que sea capaz de identificar el tópico del cual el usuario esta solicitando información y ofrecer como respuesta las más probables soluciones relacionadas.

1.5 Hitos

1. Revisión del estado de arte: Documentación histórica y conceptual de las características de las implementaciones para chatbots, así como de los modelos de Inteligencia Artificial más utilizados para el entrenamiento. Esto se llevará a cabo en el segundo capítulo cuando demos pie a la revisión del estado del arte.
2. Bases de Conocimiento y material para clasificación: recopilación, limpieza y puesta a punto de bases de datos para entrenamiento del modelo. Lo anterior, se presentará junto con las pruebas y análisis estadístico exploratorio durante el capítulo 3
3. Selección de modelo y pruebas de consistencia: Comparación entre resultados de desempeño, precisión, valor de recall y medida F1 para los distintos tópicos. El objetivo es poder obtener información que nos permita la definición del mejor modelo para lo cual encontraremos evidencia en el capítulo 3, sección 3.

4. Parametrización: Puesta a punto del modelo seleccionado. También dentro del capítulo tres encontraremos los resultados de la parametrización y la comparación de desempeño de las diferentes versiones del mismo modelo.
5. Redacción de informe final: Este se llevará a cabo durante todo el proceso de implementación.

Capítulo 2
Marco Teórico

Capítulo 2 Marco Teórico

En este capítulo revisaremos la historia del surgimiento de los chatbots desde los primeros ejercicios de simulación de conversaciones hasta los representantes actuales con características que les permiten ser reconocidos como verdaderas herramientas de conversación automática.

Posteriormente haremos una revisión de los algoritmos de inteligencia artificial que permiten el funcionamiento de los chatbots. Así como el impacto que las Redes Neuronales han tenido en el desarrollo de sistemas de conversación.

2.1 Historia de los chatbots

Según Nuruzzaman y Khadeer [24] podemos definir un chatbot como un sistema computacional de conversación que es diseñado para emular las habilidades de comunicación de un ser humano y que interactúa con un usuario. El uso extenso de chatbots para comunicación, mayoritariamente aplicado entre empresas y sus usuarios o clientes, tiene poco más de 50 años desde la primera aparición de una inteligencia artificial capaz de responder a preguntas.

El primer ejemplo remonta al año 1966 cuando un equipo del MIT desarrolla ElizaBot [30]. Este primer acercamiento tenía como objetivo, en primera instancia, demostrar que la comunicación entre maquinas y humanos era posible, además, el probar que era factible utilizar una máquina para proveer psicoterapia. Según la definición de Weizenbaum, autor del artículo, ELIZA " *es un programa operativo que permite que cierto tipo de conversación entre humanos y máquinas pueda ser llevado a cabo*" lo que obliga a pensar en primera instancia, en una inherente incapacidad para hablar de cualquier tema y en cualquier sentido.

Este sistema utilizaba el siguiente esquema de pasos:

1. Identificación de la palabra clave
2. Descubrimiento del contexto
3. Elección de la transformación adecuada
4. Generación de la respuesta en ausencia de la palabra clave

5. Aprovisionamiento de cierta capacidad de editarse a sí mismo.

Como podremos observar más adelante, la delimitación de reglas específicas de funcionamiento, sobre todo para la generación de respuestas automáticas provoca cierto acartonamiento de la capacidad de responder de manera adecuada. Esto es bien explicado cuando entendemos que preestablecer las condiciones de funcionamiento, o delimitar a cierto tópico o formato provoca que los chatbots solo puedan responder si el usuario muestra un patron léxico que permita ser identificado dentro de la base de conocimiento con la cual el modelo fue entrenado [1].

La incapacidad de ELIZA para responder a cuestionamientos más allá de solo encontrar palabras clave dentro de las oraciones usadas como *inputs*, provoca de manera directa la búsqueda de una arquitectura un tanto diferente: ALICE.

Definido como Artificial Linguistic Internet Computer Entity [20], ALICE , está basado en la arquitectura de ElizaBot con la principal diferencia de que ésta no utilizaba reglas preestablecidas, sino que trata de identificar patrones dentro de las conversaciones para la construcción de las preguntas y las posibles soluciones. El principal inconveniente es justamente el modelado de la personalidad que es necesario para el correcto funcionamiento del chatbot, como rasgos, actitudes, estado de ánimo, emociones y estados físicos.

El siguiente ejemplo se llama Mitsuku [8], que inicialmente fue diseñado para solo funcionar a través de conversación mecanografiada con base en las reglas de Artificial Intelligence Mark-up Language que es un lenguaje de programación basado en XML. Lenguaje que fue pensado para ayudar en la creación de la primera entidad informática de lenguaje en línea (ALICE). Tiene como objetivo la integración en una red de bots y fungir como una capa de personalidad [7]. Lo interesante de este ejemplo es que no utiliza técnicas de Procesamiento de Lenguaje Natural sino que trabaja a través de patrones heurísticos y está alojado en Pandorabot.

Pandorabot es una compañía de Inteligencia Artificial que funciona como un hosting de servicios web para el desarrollo y el despliegue de chatbots. La cual

ofrece dentro de su infraestructura, la descarga completamente gratuita del código fuente para ALICE y Mitsuku [17].

CleverBot es uno de los chatbots de entretenimiento más populares, además su implementación no está basada en técnicas de inteligencia artificial. Ha sido desarrollado para recolectar una larga cantidad de datos basados en intercambios conversacionales con personas a través de un *crowdsourcing*. La principal diferencia con otros chatbots es que éste no tiene respuestas preprogramadas. En lugar de eso, es capaz de simular una conversación natural a través de aprender de las entradas del usuario, y su retroalimentación con el uso constante de la herramienta.

IBM ha desarrollado un chatbot al igual que otras compañías grandes de servicios en la nube. Watson es un chatbot basado en reglas que fue diseñado para ofrecer información y responder preguntas sencillas a través de un modelo de análisis de Lenguaje Natural y Métodos de Machine Learning Jerárquicos [13]. Lo interesante de este ejemplo es que a lo largo de tres días en febrero de 2011, para probar sus capacidades reales, participó en un partido especial de dos juegos en el concurso Jeopardy ganando un total de $1,000,000 de dólares que fueron donados a caridad[1].

Google DialogFlow[2], mejor conocido como API.ai y desarrollado por Google como parte de las herramientas de Google Cloud Platform. Permite a las aplicaciones de los desarrolladores permitir a sus usuarios interactuar con interfaces a través de voz e intercambio de texto.

Por último, Amazon Lex[3] es un servicio de AWS para la construcción de interfaces de conversación dentro de las aplicaciones usando voz y texto. Provee algunas funciones de aprendizaje profundo y la flexibilidad del Entendimiento de Lenguaje Natural (NLU), además de la automatización del reconocimiento de speech para la construcción de experiencias altamente atrayentes que brinden la sensación de ser conversaciones de verdad.

---

[1] Recuperado de : https://es.wikipedia.org/wiki?curid=4411054
[2] Recuperado de : https://cloud.google.com/dialogflow/cx/docs
[3] Recuperado de : https://aws.amazon.com/es/lex/

2.2 Machine Learning: Revisión de literatura

En este apartado revisaremos algunos de los modelos más usados para la construcción de chatbots, así como una visión rápida a la teoría que los fundamenta, desde el análisis de textos hasta algunas aplicaciones de redes neuronales y aprendizaje profundo.

2.2.1 Words Embeddings

Puede definirse a Word Embeddings como al conjunto de técnicas de procesamiento y modelado de lenguaje natural, donde se implica el encaje matemático de un espacio de una dimensión por palabra, a un espacio vectorial continuo con menos dimensiones [13]. Lo anterior refiere al procesamiento de un conjunto de documentos a través de las palabras que los componen donde los documentos o las frases son representadas como un conjunto de números reales. Esta asignación de vector a cada una de las palabras guarda siempre un poco de información gramatical y contextual, lo que permite mantener dentro del preprocesamiento información sobre las palabras que se encuentran alrededor. Una vez se hubiesen terminado el cálculo y asociación de cada uno de los vectores, entonces es posible realizar operaciones con ellos, el ejemplo más sencillo es aquel en donde se tiene el vector correspondiente a la palabra “Rey”. Este está asociado al vector de la palabra “Hombre”. Así, si se le resta el vector “Hombre” y se le suma el vector “Mujer”, quedaría entonces el vector “Reina”[4].

Fue explicado y utilizado por primera vez por Tomás Mikolov, Kai Chen, Greg Corrado y Jeffrey Dean, todos del equipo de investigación de Google en su artículo seminal “Efficient Estimation of Word Representations in Vector Space” [28]; donde se establecen dos arquitecturas de modelaje de palabras (y oraciones) que permiten la representación en vectores continuos de grandes conjuntos de datos. En este sentido, el cálculo de los valores dentro de los vectores se lleva a cabo con información del contexto de las palabras.

La construcción de dichos vectores de palabras puede ser abordado de varias maneras. De entre las más comunes se encuentran el uso de Análisis de

[4] Recuperado de: https://blog.enzymeadvisinggroup.com/natural-language-processing

Semántica Latente[5], Modelos de Gramas[6], Latent Dirichlet Allocation[7] además de Common Bag of Words[8].

Según el artículo “Comparative study of LSA vs Word2vec embeddings in small corpora: a case study in dreams database” [5] esta metodología es una de las más utilizadas en la representación de palabras. Utiliza como insumo un corpus de entrenamiento. Este corpus puede estar definido por un conjunto de documentos y permite la construcción de una matriz de co-ocurrencia (que es el conteo de cuántas veces una palabra aparece en todo el corpus por documento) [5]. Posteriormente, se le aplica una normalización que permite el tratamiento uniformado de cada palabra por sí misma para finalmente llegar a la construcción de los vectores a través de Singular Value Decomposition[15], que proyecta cada palabra en un subespacio predefinido de dimensiones [5].

Hablar de la aplicación de las redes neuronales al trabajo de procesamiento semántico, nos obliga a referir al trabajo realizado por Tomas Mikolov *et al* [15, 17] el cual consiste en un modelo llamado “Word2vec” que esta construido a traves de dos modelos de redes neuronales: Continious Bag of Words y Skip Gram[16].

En el caso de Continious Bag of Words. El objetivo es la predicción de la palabra objetivo a través de las palabras del contexto. Por ejemplo, de la oración “El rey David tiene un gran reino”, siendo la palabra objetivo “David”, el input para el entrenamiento sería:

$$x = [El,\ rey,\ tiene,\ un\ ,\ gran,\ reino] y = [David]$$

[5] El análisis semántico latente (LSA) es una técnica en el procesamiento del lenguaje natural, en particular la semántica distributiva, para analizar las relaciones entre un conjunto de documentos y los términos que contienen mediante la producción de un conjunto de conceptos relacionados con los documentos y términos. LSA asume que las palabras que tienen un significado cercano aparecerán en fragmentos de texto similares (la hipótesis de distribución).

[6] Un modelo de n-grama es un tipo de modelo probabilístico que permite hacer una predicción estadística del próximo elemento de cierta secuencia de elementos sucedida hasta el momento. Un modelo de n-grama puede ser definido por una cadena de Márkov de orden n-1.

[7] Latent Dirichlet Allocation (LDA) es un modelo generativo que permite que conjuntos de observaciones puedan ser explicados por grupos no observados que explican porqué algunas partes de los datos son similares. LDA es un ejemplo de modelo de categorías y fue presentado como un modelo en grafo para descubrir categorías por David Blei, Andrew Ng y Michael Jordan en 2002.

[8] El modelo de bolsa de palabras es una representación simplificadora que se utiliza en el procesamiento del lenguaje natural y la recuperación de información (IR). En este modelo, un texto (como una oración o un documento) se representa como la bolsa (multiset) de sus palabras, sin tener en cuenta la gramática e incluso el orden de las palabras, pero manteniendo la multiplicidad.

Al final del entrenamiento, se utilizan los parámetros de la capa oculta como una representación del vector de salida. El nombre “Continious Bag of Words” se esquematiza toda vez que dependiendo del tamaño de las ventanas es el número de x’s que estarán asociadas a la palabra objetivo.

Dado que el modelo Word2vec, del cual CBOW es tan solo uno de los componentes, pertenece a la familia de aprendizaje no supervisado, es que podemos llevar a cabo una implementación sin tener que proveer al modelo del etiquetado del corpus. Por ello, “podemos construir la arquitectura del modelo CBOW como un algoritmo de clasificación de deep learning toda vez que podemos tomar al conjunto de las variables de contexto como los valores de input, y tratar de predecir el valor de la palabra objetivo Y” [9]

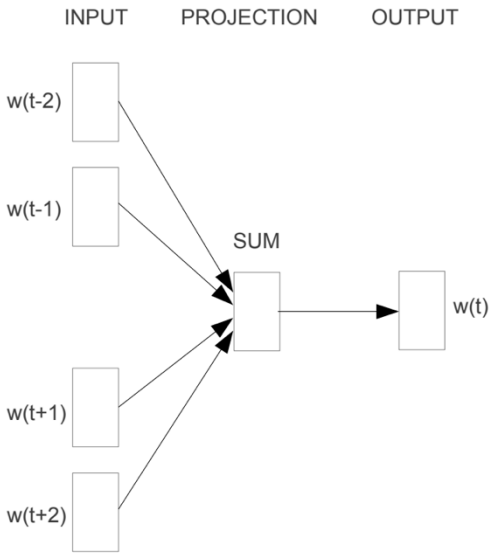

Figura 1: Esquema de CBOW
Fuente: Tomas Mikolov *et al. 2013*

Por otro lado, Skip Gram Model, arquitectura completamente nueva que hace su primera aparición con Tomas Malkovi [28], mantiene, sin embargo, una estructura lógica muy simple. Con el objetivo de la predicción de las palabras de contexto, dada una palabra central como entrada. Florian [14] lo esquematiza de la siguiente manera:

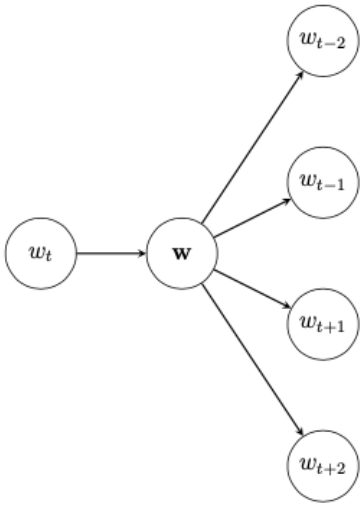

Figura 2: Esquema de Skip Grams
Fuente: Florian, Peters. *2017*

Como podemos observar, CBOW y Skip Grams funcionan como dos piezas de un solo modelo, donde la salida de uno es la entrada del siguiente. De esta forma el entrenamiento de la arquitectura conjunta, estará enfocada en la obtención de las *inner layers* y su utilización como los vectores asociados a la sentencia gramatical. La representación esquemática de estas dos arquitecturas es la siguiente:

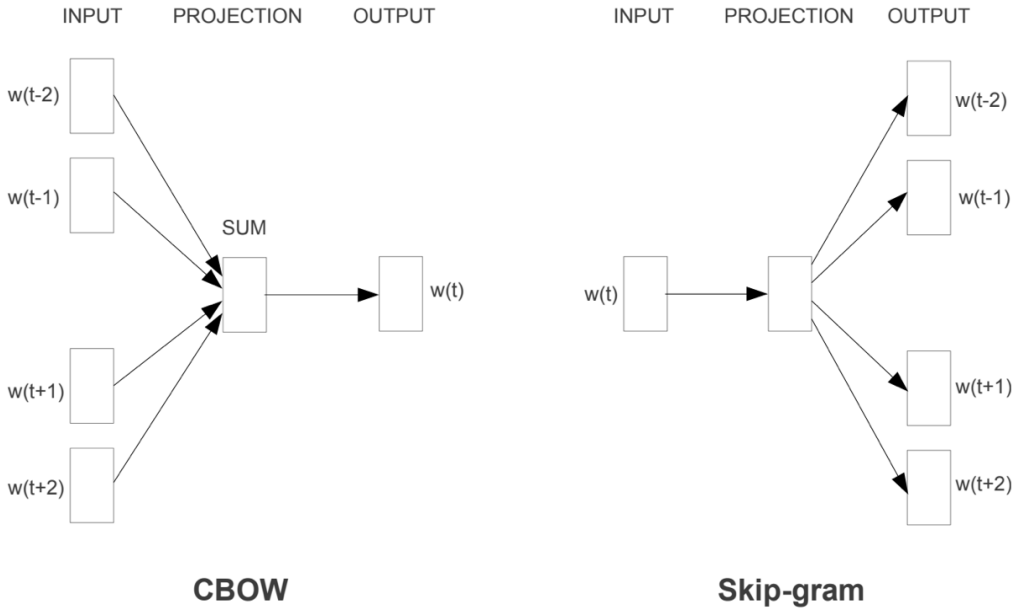

Figura 3: Esquema de Word2Vec
Fuente: T. Mikolov. 2013

2.2.2 Latent Semantic Analysis

Podemos definir al *Latent Semantic Analysis* (LSA) como un conjunto de algoritmos y herramientas que a través del análisis del lenguaje natural logran una representación en un espacio vectorial previamente definido de las relaciones entre las sentencias gramaticales. Según Wiemer-Hastings [31], este algoritmo fue originalmente pensado como un indexador que permitiera la búsqueda rápida de documentos dentro de un texto. Refiere, además, que han existido algunas otras formas de aproximación al análisis de semántica, sobre todo cuando hablamos de *keyword-matching, weighted-keyword-matching* y *vector based representation* que tenían como fundamento el número de ocurrencias de una palabra en el texto.

Sin embargo, una nueva aproximación a LSA, refiere que la principal diferencia de estos algoritmos es que éste último utiliza una función de descomposición que permite identificar aquellos componentes de los vectores que mejor funcionen en la representación. Utilizar *Singular Value Decomposition* (SVD) para una especie de reconfiguración de los datos se basa en el supuesto de que dentro de las variables que explican otra relacionada, siempre se encuentra un conjunto de estas que son latentes y que agrupan, o dicho de otra forma, los factores comunes que unen a todas las características asociadas las cuales se asumen independientes y ortogonales en el espacio del vector de salida.

Normalmente, el número de variables latentes que se intenta extraer de cualquier corpus, relacionadas con una palabra, son 300. Ésta es una suposición que se ha hecho normal cuando se trabaja con SVD. Los pasos básicos para el análisis a través de esta herramienta son:

- Separar el corpus de entrenamiento en una serie de subconjuntos llamados “documents”
- Crear una matriz de coocurrencias de los documentos, los términos y las palabras. Cada una de las celdas en esta matriz corresponde con un documento x. No se recomienda el uso de steamming o lematización para el tratamiento previo de las palabras. Si existen, por ejemplo, una cantidad *m* de términos y una cantidad *n* de documentos, la matriz podrá ser vista como una representación que

tiene una dimensión de *m* para cada documento y de *n* para cada uno de los términos.

- La matriz debe normalizarse para evitar que la mayor ocurrencia de palabras mueva los valores de los vectores asociados.
- Por último, el algoritmo necesita de un parámetro K que determina el tamaño del vector que es el número de dimensiones deseadas. En principio, SVD deberá ser calculado con todas las dimensiones para crear tres matrices que, cuando se multipliquen entre ellas, deberán crear la matriz inicial, por ello se presupone la ortogonalidad de las características latentes. Sin embargo, debido a la cantidad de memoria necesaria para llevar esto a cabo, se utilizan matrices dispersas donde sólo se seleccionan los principales componentes de éstas.

Wiemer-Hastings [31] refiere que el resultado del procesamiento anterior son tres matrices. Una de ellas es la representación k-dimensional para cada uno de los documentos, la siguiente es k-dimensional para cada uno de los términos, y la siguiente mantiene solo el valor de los valores únicos. El resultado final de la aplicación, es el obtenido de la multiplicación de la representación k-dimensional de los documentos y el vector de los valores singulares.

Este algoritmo presupone que las palabras que están más cercanas entre ellas, en cuanto a significado semántico, ocurren en lugares cercanos del texto, bajo la hipótesis de distribución[9], por lo que su principal uso es para la comparación a través de medidas de similitud como la similitud coseno; que se calcula para cada conjunto par cualesquiera de la matriz de salida [25].

[9] La distribución semántica es un área de investigación que desarrolla estudios, teorías y métodos para cuantificar y categorizar la similitud semántica entre diferentes ítems lingüísticos basados en sus propiedades de distribución en largos conjuntos de datos. Recuperado de: https://en.wikipedia.org/wiki/Distributional_semantics

2.2.2 Recurrent Neural Network

Como se ha mencionado durante el texto, la mayoría de los desarrollos para chatbots están construidos y pensados a través de una metodología enfocada en clasificación. Es decir, se le presenta un diálogo al algoritmo y se califica la capacidad que este tiene para poder clasificar de manera correcta las intenciones o el tema relacionado. Dicho de otra manera, después de enfocarse en el manejo de textos a través de vectores, los modelos clásicos generalmente consisten en la proyección a través de capas de neuronas de estos mismos vectores para lograr encontrar la clasificación más próxima a las intenciones del texto. Sin embargo, el principal problema de la estructura "clásica" es la rigidez del proceso de aprendizaje, sobre todo cuando se habla de las características de los valores de entrada.

Podemos definir a las redes neuronales recurrentes como aquellas que integran bucles de realimentación, permitiendo a través de ellos que la información persista durante algunas fases o épocas de entrenamiento a través de conexiones desde las salidas de las capas, que incrustan sus resultados en los datos de entrada. Las conexiones entre nodos forman un gráfico dirigido a lo largo de una secuencia temporal. Dicho de otra forma, funciona como una red con múltiples copias de sí misma, cada una con un mensaje a su sucesor [26].

Las principales diferencias con las redes neuronales clásicas es que tratan datos secuenciales de forma eficiente, recuerdan las salidas anteriores como entrada y pueden tratar secuencias muy largas, elemento a elemento; además, son capaces de procesar una secuencia de tamaño arbitrario sobre los vectores utilizados como inputs utilizando una función de transición sobre el vector de cada uno de los steps [22].

Las diversas metodologías que pueden ser aplicables para la implementación de un chatbot son bastante variadas como hemos podido observar. Lo interesante de la selección es que debe estar enfocada en resolver el problema que se pretende atender; es decir, dependiendo de los objetivos, y las características de las preguntas y necesidades de los clientes.

En nuestro caso, la implementación estará encaminada en la caracterización de una solicitud de información. Esta solicitud puede incluir desde la respuesta a una pregunta en específico, tanto como a la creación de un ticket de servicio al cliente para una acción específica por parte del equipo de desarrollo. Esto se verá en el siguiente capítulo.

2.2.3 Latent Dirichlet Allocation

Los autores describen éste método como un modelo probabilístico generativo para colecciones de información discreta como textos. Es un modelo Bayesiano jerárquico de tres niveles en donde cada entrada de una colección, en este caso un documento, es modelado como una mezcla finita sobre un conjunto de tópicos no definidos. Dicho de otra forma, permite que los conjuntos de observaciones puedan ser explicados por grupos no observados que puedan explicar porque algunas partes de los datos son similares. Es decir, en el caso del análisis de lenguaje natural o de textos, permite saber cómo el hecho de pertenecer al mismo documento, para dos oraciones completamente diferentes, coadyuva en la probabilidad de que el texto se trate de cierto tópico. [6]

La clave, como se menciona en [10] es que las palabras siguen una hipótesis de bolsa de palabras o mejor dicho, que el orden no importa, que el uso de una palabra es ser parte de un tema y que comunica la misma información sin importar dónde se encuentra en el documento. Eso genera un problema en el momento de frases en particular. Es decir, dado que el cálculo de la probabilidad de pertenecer a un tópico se hace sobre la ocurrencia de palabras en una bolsa de palabras, no importando el orden de aparición de éstas, es que es posible que tome como del mismo tópico dos oraciones semánticamente diferentes, como:

1- Juan contrató a Pedro, Pedro es electricista
2- Pedro contrató a Juan, Juan es albañil.

Dadas las reglas establecidas para el cálculo de las probabilidades, para el caso del modelo LDA ambas son iguales y pertenecen al mismo conjunto. Sin embargo

lo anterior, es posible verificar el buen comportamiento de este algoritmo en contraposición del análisis probabilístico de semántica latente o (pLSA) [10] .

Capítulo 3
Metodología

Capítulo 3 Metodología

En el presente capítulo abordaremos los enfoques a través de los cuales se llevará a cabo el diseño de la infraestructura del modelo y su posible aplicación a un herramienta completamente administrada en la nube. Comenzaremos por la revisión de las bases de conocimiento sobre las cuales se ejecutarán los procesos de entrenamiento y predicción. Así mismo, construiremos con esta misma revisión las posibles aplicaciones de los diferentes algoritmos, revisando y comprobando la factibilidad técnica de su usabilidad, así como de la implementación y el desempeño.

Posteriormente explicaremos el funcionamiento básico de una aplicación de chatbot de dominio específico y los flujos de información a través de los cuales los clientes o usuarios envían los mensajes para ser atendidos. Recordemos, además, que el principal enfoque del presente trabajo es la aplicación de un Chatbot para servicio al cliente, dicho objetivo está delimitado por el conjunto de datos que hemos elegido.

3.1 Bases de datos

Las bases de datos que hemos elegido para la construcción y entrenamiento del algoritmo contiene las conversaciones a través de Twitter que algunas empresas de tecnología mantienen con los usuarios de servicio al cliente, está disponible en Kaggle[10]. Ésta es una plataforma de competencias y aprendizaje online las cuales están enfocadas en la búsqueda de las mejores versiones de algoritmos y soluciones completas de Inteligencia Artificial.

La información está disponible para: Apple, Amazon, Uber, American Airlines, Virgin entre otros, y cuenta con 794,335 registros de entrada. Además de contar con referencias de los tweets de preguntas y su respuesta codificada.

La distribución del conjunto de entrenamiento no es homogénea entre las distintas compañías. En la figura número 4 encontramos dicha distribución donde

[10] El conjunto de datos de entrenamiento es público y puede ser descargado directamente desde la página oficial de Kaggle : https://www.kaggle.com/thoughtvector/customer-support-on-twitter

podemos observar que los registros para Apple rebasan en casi el doble de items a la empresa en segundo lugar; Uber.

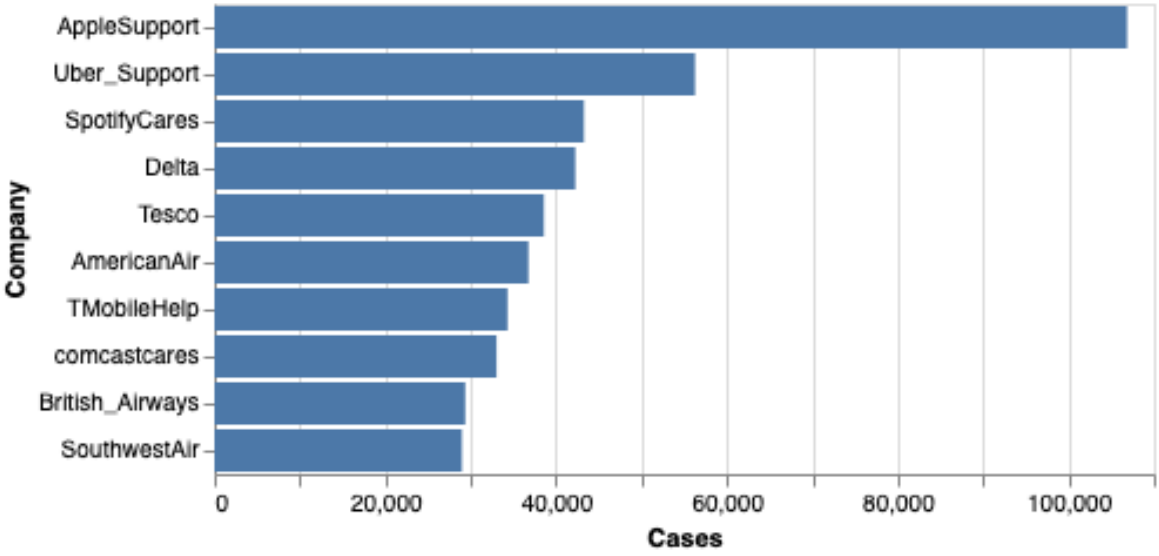

Figura 4: Top 10 compañías con mayores registros
Fuente: Elaboración Propia

El conjunto de datos cuenta con información individual por tweet, por lo que tendremos dos entradas diferentes: una para el usuario y otra para la compañía. Mismas que están relacionadas a través de un tweet_id. Cuidando el resguardo de información confidencial y para evitar poder identificar los usuarios a través de sus cuentas, a las entradas "respuesta" se le cambió el id del usuario por un indexado numérico que permitiera su identificación con la empresa a la que solicitan información. Una muestra de los datos puede verse en la siguiente tabla:

tweet_id	author_id	inbound	created_at	text	response_tweet_id	in_response_to_tweet_id
1	sprintcare	FALSO	Tue Oct 31 22:10:47 +0000 2017	@115712 I understand. I would like to assist you. We would need to get you into a private secured link to further assist.	2	3.0
2	115712	VERDADERO	Tue Oct 31 22:11:45 +0000 2017	@sprintcare and how do you propose we do that		1.0
3	115712	VERDADERO	Tue Oct 31 22:08:27 +0000 2017	@sprintcare I have sent several private messages and no one is responding as usual	1	4.0
4	sprintcare	FALSO	Tue Oct 31 21:54:49 +0000 2017	@115712 Please send us a Private Message so that we can further assist you. Just click 'Message' at the top of your profile.	3	5.0
5	115712	VERDADERO	Tue Oct 31 21:49:35 +0000 2017	@sprintcare I did.	4	6.0

Cuadro 2. *Información Disponible.*
Fuente: Elaboración Propia

La columna tweet_id es un identificador construida con el indexado de los registros, mientras que el author_id es el identificador del usuario que

mencionamos anteriormente. Los nombres sólo están disponibles para los usuarios que representan el servicio al cliente de las compañías.

La columna inbound refiere al tipo de texto desde la perspectiva de servicio al cliente, de esta forma un inbound toma el valor de true cuando es un texto de entrada o solicitud de información de un cliente y de false en el momento en que es una respuesta por parte del área de servicio al cliente.

Es necesario puntualizar que el conjunto de datos está en el idioma inglés, por lo que la interfaz final de cara al cliente podrá aceptar sólo comunicación en este lenguaje. Para una posterior ampliación del sistema de preguntas y respuestas, es necesario poder contar con un dataset del idioma objetivo.

3.2 Limpieza y preprocesamiento

El preprocesamiento, la limpieza y la preparación de los datos para su inclusión en el modelo puede verse ejemplificada en el siguiente esquema:

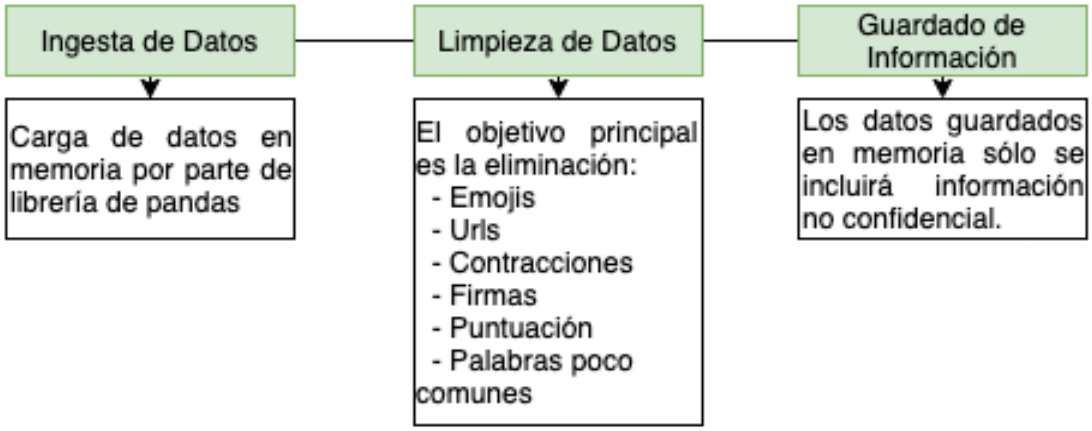

Figura 5: Preprocesamiento
Fuente: Elaboración Propia

La carga de los datos en memoria, permite el tratamiento de la información mucho más rápido una vez que se particiona y se trabaja por conjuntos de datasets. La principal idea de la carga de datos es justamente poder garantizar el correcto tratamiento de los datos en los subsecuentes procedimientos. Por ejemplo, se ha separado en dos conjuntos de datos completamente diferentes para las preguntas

y las respuestas. Posteriormente se realiza la limpieza por separado pero a través de herramientas llamadas *pipelines* que permiten repetir procesos en diferentes conjuntos de datos.

El proceso de limpieza de información comienza con la conversión de la información en sólo texto, es decir: los emojis son limpiados en su totalidad, así como las direcciones urls. Por otro lado, la eliminación de las palabras relacionadas con chats y mayormente visualizadas en plataformas de comunicación también son eliminadas. Por último, se eliminan las palabras poco frecuentes o fuera de vocabulario y los *stopwords*.

El último paso del preprocesamiento es el guardado de la información resultante. Esto es importante para poder garantizar el correcto funcionamiento del algoritmo y disminuir el tiempo de respuesta a nuestros clientes. De lo anterior debemos resaltar que como en la mayoría de los sistemas de información, un problema importante es la utilización de los recursos. Por ello, los conjuntos de datos de salida, después del preprocesamiento y limpieza se guardarán fuera de memoría del entorno de ejecución. Debido principalmente al gran tamaño del conjunto de salida, el cual tiene el formato siguiente:

ID	question	response
94	changing showing correctly social medium platform	lets take a closer look into this issue select the following link to join us in a message dm and we will go from there
95	need answer annoying question mark	we would like to look into this with you which model do you have and is ios 111 installed any steps tried so far
96	sent message dm hey anyone else upgraded io yall issue capital mail app put	hey let us work together to figure out what is going on meet us in message dm and we will continue from there
97	tf wrong keyboard	fill us in on what is happening then we can help out from there
98	hello line closed tonight help	what is going on we are hapy to help if we can

Cuadro 3. *Preguntas originales*
Fuente: Elaboración Propia

Como podemos observar el conjunto de datos de salida sólo contiene un id, que permite identificar un intercambio de información, o una conversación. De esta forma, la pregunta siempre será hecha por parte del cliente y la respuesta es siempre dada por el personal de atención al cliente.

Cómo hemos referido anteriormente, el conjunto de datos se encuentra desbalanceado respecto de las empresas que lo conforman. Además, el hecho de que tengan actividades tan dispares podría obstaculizar el proceso de entrenamiento de un modelo cuyo objetivo sea el de caracterizar, por ejemplo, los tópicos dentro de un conjunto de documentos. Por ello, hemos cortado el dataset de entrenamiento para que esté constituido tan sólo por las entradas referentes a la empresa Apple, que es la que cuenta con poco más de 140,000 registros entre preguntas y respuestas.

3.3 Construcción del modelo

La solución propuesta para la implementación del “ChatBot de dominio específico para atención a usuarios” se compone de dos procesos diferentes. Por un lado, un algoritmo de clasificación que permita asignar clases o tópicos a las diferentes entradas o preguntas que se le puedan hacer al modelo. Para ello se retoma de la revisión de literatura el algoritmo LDA, que, como se ha mencionado es un modelo generativo que permite que conjuntos de observaciones puedan ser explicados por grupos no observados que explican porqué algunas de las partes de los datos son similares. Es decir, si lo que se busca es modelar la relación entre palabras dentro de varios documentos, se presupone que cada documento es una mezcla de un pequeño número de categorías, y la aparición de cada palabra dentro de un documento se debe a una de las categorías a las que el documento pertenece [10].

3.3.1. Vectorización

La vectorización del texto supone la separación de las oraciones entre las palabras que las componen. Respecto del marco teórico que se propone, cada una de las preguntas funcionará como un documento. De esta forma, la extracción del tópico de las preguntas estará determinado por estas características escondidas dentro de las oraciones.

El proceso de vectorización se lleva a cabo a través del paquete count_vectorizer de la librería sklearn ya que no sólo permite la vectorización de

los documentos de entrada sino también es posible pasarla como objeto dentro de una clase para poder ser utilizada en la vectorización de la pregunta sobre la cual se hará la predicción.

Esta librería regresa un vector tal que cada uno de las posiciones está relacionada con una palabra en específico. De esta forma, tendremos al final una matriz con muchos ceros, sobre todo atendiendo a que el diccionario completo de todas las posibles palabras utilizadas en twitter para la comunicación con los clientes puede ser más bien extenso.

3.3.2 Selección de número de tópicos

Para la selección del número de tópicos se han comparado los resultados de Accuracy[11] y Recall[12] para diferentes modelos. En el caso del entrenamiento de un modelo con cinco tópicos, lo que se ha podido resaltar es que la distribución entre estos no es similar. Así, por ejemplo, tenemos dos tópicos con menos de 10 mil registros, mientras que tres de ellos tienen alrededor de veinte mil.

Las figuras 3, 4 y 5 muestran la relación de la distribución entre los diferentes tópicos que el modelo LDA encontró como relevantes. La homogeneidad de las clases que son generadas en el preprocesamiento es importante debido a que éstas hacen referencia al número de comparaciones que el modelo tendrá que hacer para regresar cuál es la pregunta más parecida dentro del tópico. Adicionalmente a que es una muestra gráfica sobre la efectividad del modelo para predecir nuestra pregunta puesto que demostraría que no existe sesgos entre las diferentes clases y que por lo tanto las métricas de evaluación podrían ser mucho más consistentes.

---

[11] La métrica Accuracy mide el porcentaje de casos que el modelo ha acertado y puede definirse como la suma de los *true positives* entre el total de todos los registros. Recuperado de: https://deepai.org/machine-learning-glossary-and-terms/accuracy-error-rate

[12] Definido como la capacidad que tiene un modelo para predecir correctamente los valores que han sido clasificados como true positives. En el contexto de los problemas de analisis de textos esta definición se entiende como la cantidad de documentos relevantes que han sido exitosamente devueltos por el algoritmo. Recuperado de: https://en.wikipedia.org/wiki/Precision_and_recall#Recall

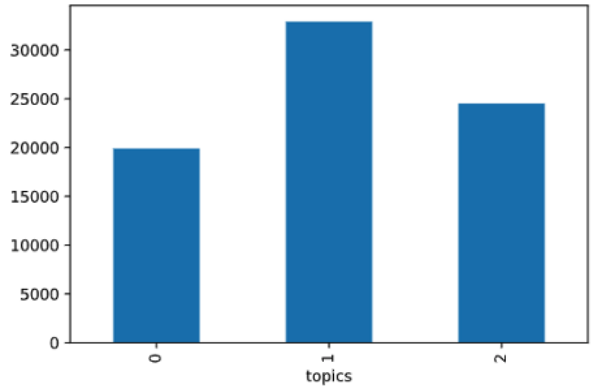

Figura 6. Distribución de datos por tópicos (3 clases)
Fuente: Elaboración propia

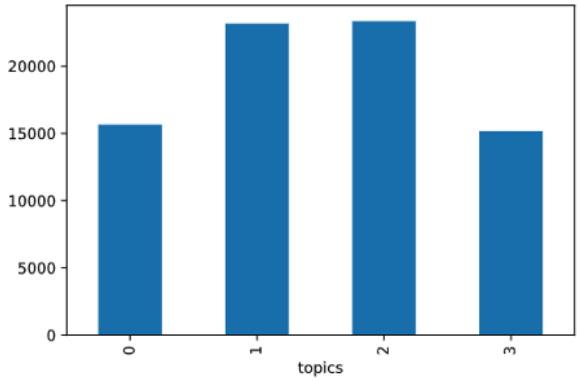

Figura 7. Distribución de datos por tópicos (4 clases)
Fuente: Elaboración propia

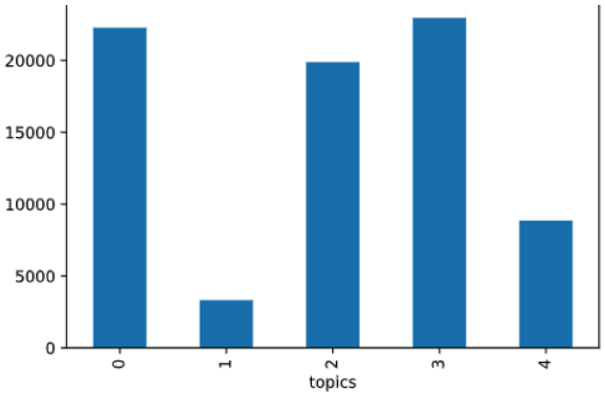

Figura 8. Distribución de datos por tópicos (5 clases)
Fuente: Elaboración propia

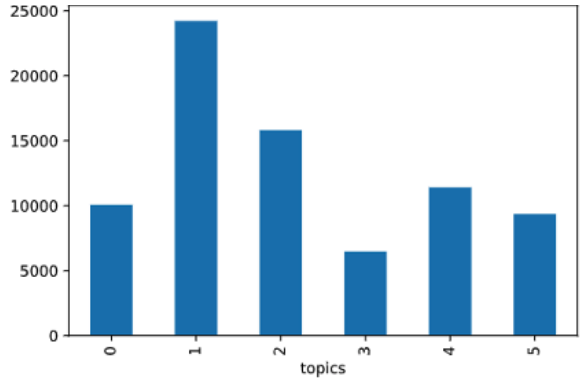

Figura 9. Distribución de datos por tópicos (6 clases)
Fuente: Elaboración propia

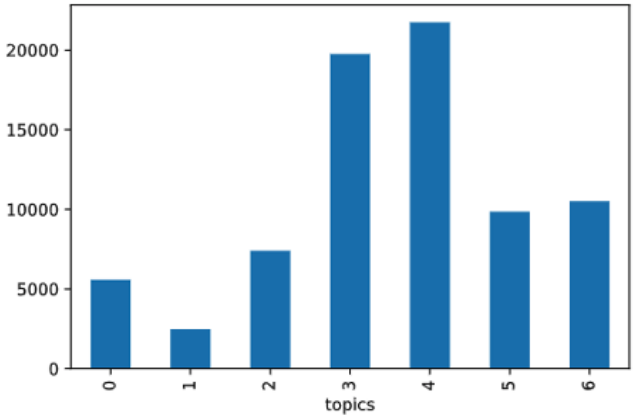

Figura 10. Distribución de datos por tópicos (7 clases)
Fuente: Elaboración propia

Derivado de lo anterior podemos observar que la figura 7, que refiere el ejercicio para el modelo con 4 clases es mucho más homogénea respecto de la cantidad de registros que el modelo LDA logra clasificar dentro de cada uno de los tópicos. Sin embargo, es fundamental llevar a cabo pruebas y experimentos con los modelos entrenados para poder medir su capacidad de responder correctamente a una pregunta que no se encuentra en el dataset, o dicho de otra manera, su capacidad de generalización.

3.4 Medidas de Similitud

Como parte fundamental de la arquitectura de pregunta – respuesta sobre la que está basada el presente, se encuentra la capacidad del modelo para poder encontrar la similitud entre los diferentes registros que componen el universo de búsqueda. En nuestro caso, es primordial que las preguntas que el usuario genere sea posible relacionarlas con aquellas que el modelo utilizó como conjunto de entrenamiento y creación de tópicos.

Una de las principales medidas que se utilizan para comparar la distancia entre dos objetos es la similitud coseno[13] que puede definirse como:

> "…una medida de la similitud existente que refiere la distancia entre dos vectores en un espacio que posee un producto interior con el que se evalúa el valor del coseno del ángulo comprendido entre ellos. Esta función trigonométrica proporciona un valor igual a 1 si el ángulo comprendido es cero, es decir, si ambos vectores apuntan a un mismo lugar…" [27]

donde la fórmula es expresada como sigue:

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

[13] Esta distancia se emplea de manera frecuente para el cálculo de la similitud entre dos textos, sobre todo en minería de datos y procesamiento de grandes conjuntos de información.

Estos vectores son construidos a través de *word embeddings*[14] que podría definirse como un conjunto de técnicas de modelado y aprendizaje automático enfocado en el procesamiento del lenguaje natural.

Existe también como principal medida de similitud el coeficiente de correlación de pearson el cual puede definirse como la división entre la covarianza de dos variables y la raíz cuadrada de la multiplicación de las varianzas. Permite conocer el nivel de relación lineal entre dos variables continuas. Sin embargo, el coeficiente de correlación de pearson sólo es capaz de detectar dependencias lineales entre variables. Cómo se refiere Sara Dorado Alfaro [10] un claro ejemplo de las limitaciones es cuando podemos considerar dos variables aleatorias continuas tales de X1 sea igual al seno de X2.

$$\rho_{X,Y} = \frac{\sigma_{XY}}{\sigma_X \sigma_Y} = \frac{\operatorname{Cov}(X,Y)}{\sqrt{\operatorname{Var}(X)\operatorname{Var}(Y)}}$$

Por último mencionaremos a la entropía e información mutua, la cual puede definirse como una medida de similitud que permite caracterizar la independencia entre variables aleatorias. Donde trata de medir la incertidumbre de una variable aleatoria y la cantidad de incertidumbre remanente cuando es conocida [10]. Esta medida de similitud fue por primera vez propuesta en *Feature selection for high-dimensional data: A fast correlation-based filter solution* de Lei Yu y Huan Liu [27].

3.5 Arquitectura del chatbot

El funcionamiento del chatbot toma como principal insumo los datos resultados de la aplicación del modelo LDA al conjunto de entrenamiento. Sin embargo, es

[14] Conversión de palabras en su representación vectorial de números reales. El cual puede realizarse a través de procedimientos matemáticos como la reducción de dimensionalidad, redes neuronales o incluso matrices de coocurrencia. Recuperado de: https://es.wikipedia.org/wiki/Word_embedding

importante resaltar el enfoque teórico sobre el cual estaría basado el comportamiento de nuestro sistema de conversación: question-answering[15].

Este sistema tiene como fundamento la búsqueda de preguntas planteadas en lenguaje natural para posteriormente relacionarlas entre sí y regresar al usuario una respuesta asociada a su pregunta. En su funcionamiento podría considerarse parecido a los motores de búsqueda donde la estructura no es capaz de almacenar algún tipo de información dentro de las interacciones pero que es capaz de resolver preguntas específicas.

Para el caso de nuestra implementación el enfoque que estamos proponiendo recibe una pregunta por parte del usuario, la clasifica dentro de los cuatro posibles tópicos en los que el conjunto de datos fue separado. Una vez seleccionado el tópico más probable al que podría pertenecer se utiliza una medida de distancia respecto de las preguntas dentro de la clase, seleccionando aquella con la similitud más grande, o con la distancia más pequeña.

El sistema deberá ser capaz de ofrecerle una respuesta que es seleccionada como la respuesta a la pregunta con mayor similitud del conjunto dado del tópico al cual la solicitud de información esté asociada. La similitud se mide a través de la distancia de los vectores asociados a las preguntas, la cual se calcula a través de la similitud de coseno que hemos revisado en el apartado 3.3 sobre medidas de similitud.

La figura número 11 representa el esquema general de funcionamiento de la implementación. Que se encuentra dividido en dos componentes principales:

1. Implementación o deployment
2. Interfaz

La implementación hace referencia al conjunto de herramientas o servicios que permiten el correcto funcionamiento del chatbot. Algunos ejemplos podrían ser servidores dedicados en la nube, instancias de procesamiento e incluso trabajos de entrenamiento. Por otro lado, cuando se habla de la interfaz, se presupone es un conjunto de herramientas enfocadas en el uso del usuario final, como una página web o un servicio como una API de consulta.

[15] Referencia: https://es.wikipedia.org/wiki/Búsqueda _de_respuestas

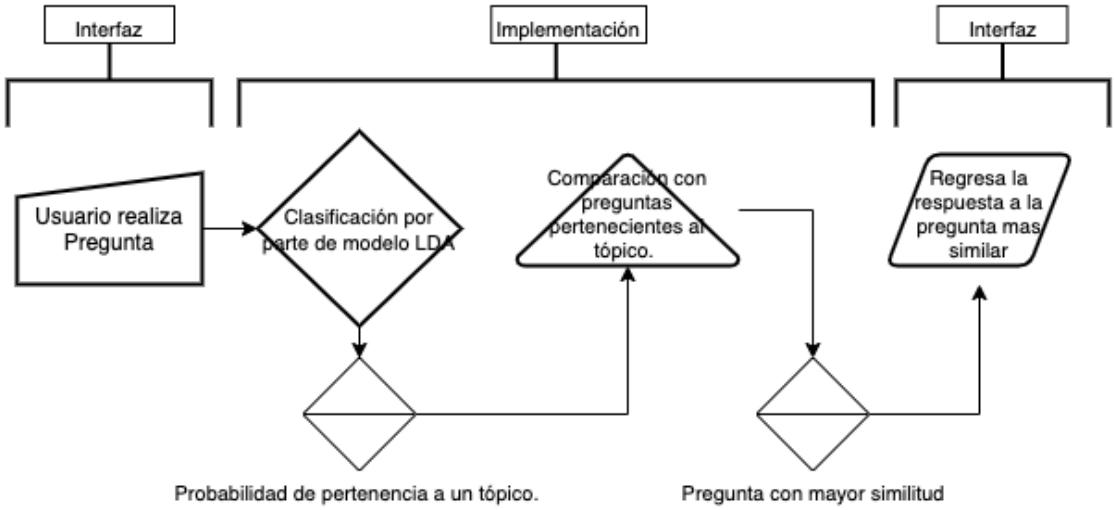

Figura 11. Implementación de chatbot
Fuente: Elaboración Propia

Debemos resaltar que aunque la definición de un chatbot es clara, la ejemplificación de los procesos que se encuentran inmiscuidos en cada uno de los pasos para poder llevar al cliente la respuesta a su solicitud no siempre lo es. De esta forma, es mucho más claro si se piensa en este programa como un framework completo donde diferentes plataformas y algoritmos se entrelazan para cumplir un objetivo: procurar al cliente una respuesta que satisfaga sus necesidades.

3.4.1 Esquematización y funcionamiento de la implementación

De manera general se presenta el formato de funcionamiento de un servicio de chatbot. Esto nos permitirá definir los pasos dentro del funcionamiento los procesos de optimización y selección de preguntas para su comparación con los vectores.

La figura 12 define el flujo de decisiones del sistema completo.

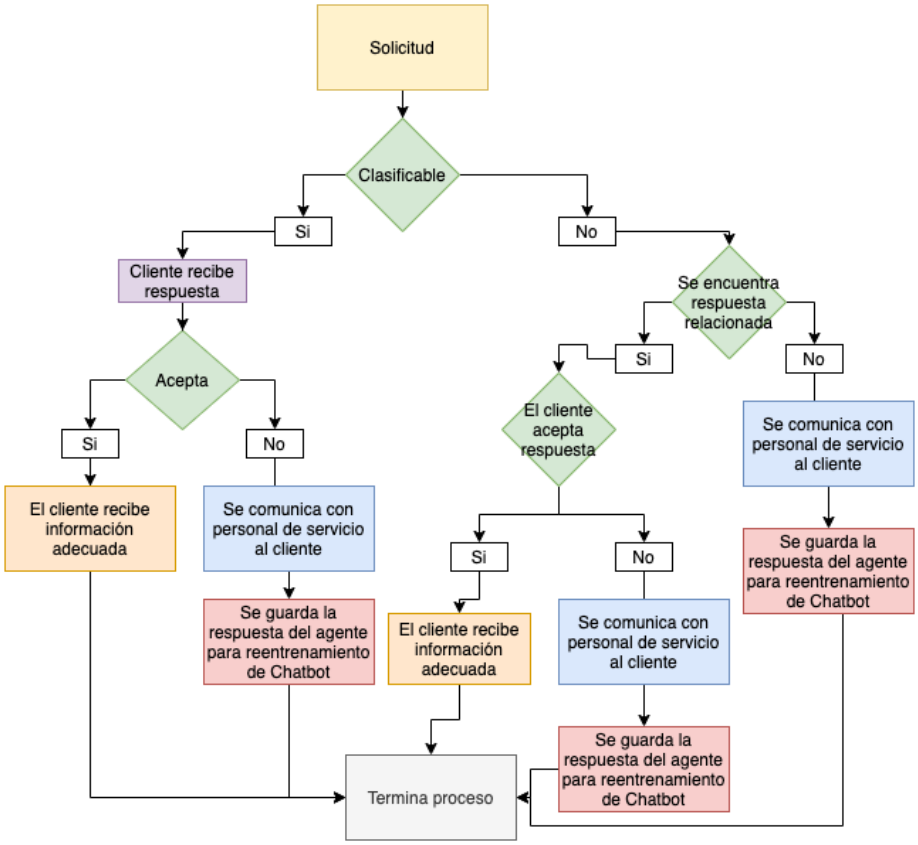

Figura 12. Flujo de trabajo de chatBot
Fuente: Elaboración propia

Como podemos observar, es posible que el entrenamiento del algoritmo de clasificación no brinde las respuestas adecuadas a las solicitudes de los clientes.

En caso de que no sea posible la caracterización, ofrecer al usuario soluciones o respuestas asociadas podría resultar útil toda vez que el cliente podría no haberse expresado de la forma correcta. Por otro lado, si las respuestas recomendadas no funcionan como salida al flujo de trabajo, el siguiente paso siempre será la comunicación con un agente de Servicio al Cliente y el resguardo de la pregunta hecha por el cliente así como la respuesta qué el agente brinda, todo ello para su uso en los procesos de reentrenamiento del algoritmo.

El funcionamiento de un chatbot se encuentra integrado por diferentes componentes. Desde el algoritmo para la clasificación e identificación de las

solicitudes del cliente o usuario, hasta la interfaz en forma de chat sobre la cual se intercambia la información entre el participante y el software.

Es importante resaltar que el proceso completo debe incluir una vía de comunicación entre los agentes y la base de conocimiento, siempre cambiante, sobre la cual toma las decisiones el algoritmo.

Capítulo 4
Resultados

Capítulo 4 Resultados

La fase de experimentación y optimización de un modelo es imprescindible toda vez que permite la mejora del modelo en miras de la puesta en marcha en producción. Entenderemos como producción al entorno final de implementación donde el cliente tendrá interacción con la herramienta.

Como hemos podido observar en el apartado de la distribución de los tópicos, algunos de ellos permiten tener una cantidad más homogénea de datos entre cada uno de ellos. Esto permitirá que la búsqueda de la pregunta más parecida a la hecha por el cliente sea hecha en una menor cantidad de tiempo. Así mismo, brindan una idea del comportamiento de las diferentes versiones de modelo respecto de su capacidad para discernir entre los tópicos asociados a las preguntas.

4.1 Generación de Tópicos

La bondad de la utilización de un modelo de clasificación para la categorización de la base de conocimiento sobre la cual se hará la búsqueda es que se obtienen las palabras asociadas a cada uno de los tópicos de salida.

Para la generación de los tópicos, como lo hemos mencionado en la metodología, se ha generado un modelo LDA que permite agrupar los diferentes textos por su posible contenido semántico.

La selección de la cantidad de palabras para construcción de los tópicos fue llevado a través de un proceso de prueba y erros sobre la capacidad computacional de la cual se dispone para la elaboración del presente proyecto. En un ejercicio subsecuente se deberá analizar la posibilidad de aumentar la cantidad de palabras y su posible impacto, tanto en el número de topicos relevantes como en la cantidad de preguntas pertenecientes a un tópico. Consideración no menor toda vez que, hay que recordar, el formato de selección de la pregunta hace un barrido sobre todo el conjunto de preguntas calculando la medida de similitud para cada una de las entradas.

A continuación enlistamos los tópicos obtenidos en los entrenamientos para cada uno de los ejemplos dentro de un rango desde 3 a 7 tópicos[16]:

1. 3 tópicos:
    a. Tópico 1: "*Apple phone iphone help got*": No tiene un problema relacionado de manera explícita. Se caracteriza como la solicitud de ayuda a un problema cualquiera.
    b. Tópico 2: "*io iphone update phone battery*": Problema relacionado con el desempeño de la batería resultado de la actualización del software de un telefóno.
    c. Tópico 3: "*Fix Iphone im update shit*": Problemas generales relacionados con una actualización de software.

2. 4 tópicos:
    a. Tópico 1: "*apple phone help email watch*": solicitud de ayuda respecto del sistema de email dentro del teléfono
    b. Tópico 2: *"fix iphone im shit letter":* No se logra relacionar un problema en específico
    c. Tópico 3: "*io update phone iphone battery*": Problema relacionado con el desempeño de la batería resultado de la actualización del software de un telefóno.
    d. Tópico 4: "*io iphone message screen app*": no se logra relacionar un problema en específico.

3. 6 tópicos:
    a. Tópico 1: "*message app io update photo*": La actualización del software no permite la salida de imágenes en los mensajes de texto.
    b. Tópico 2: "*phone fix update new io*": Problemas generales relacionados con una actualización de software.

[16] Este número de tópicos han demostrado ser los mínimos suficientes y el máximo posible para englobar la mayor cantidad de posibles temas sobre la atención a clientes de Apple.

c. Tópico 3: “*io iphone battery update phone*”: Problema relacionado con el desempeño de la batería resultado de la actualización del software de un telefóno.
d. Tópico 4: “*question mark letter box type*”: No se logra relacionar un problema en específico.
e. Tópico 5: “*apple help iphone need email*”: problema con el sistema de email del teléfono.
f. Tópico 6: “*working phone io wifi work*”: Conexión con problemas a través de wifi, resultado de una actualización de software.

4. 7 tópicos:
    a. Tópico 1: “*apple soon possible email id*”: No se logra relacionar un problema en específico
    b. Tópico 2 : “*io type la iphone que*”: No se logra relacionar un problema en específico
    c. Tópico 3: “*fix iphone shit update*”: Problemas generales relacionados con una actualización de software.
    d. Tópico 4: “*io iphone update battery*”: Problema relacionado con el desempeño de la batería resultado de la actualización del software de un teléfono.
    e. Tópico 5: “*io iphone screen work wifi*”: problemas con la conección wifi derivado de la actualización del software
    f. Tópico 6: “*apple music app store download*”: No se logra relacionar un problema en específico.
    g. Tópico 7: ”*help message iphone screen update*”: problemas de pantalla relacionado con una actualización de software

Como podemos observar, dentro de cada uno de los ejercicios con diferente cantidad de tópicos, conlleva una diferente mezcla de palabras dentro de estas clasificaciones. Sin embargo, es posible encontrar ciertas regularidades entre los diferentes modelos.

Los tópicos relacionados con las baterías y con las actualización de software aparecen al menos una vez en cada uno de los ejemplos de modelo. Esto nos brinda información sobre cuáles son las preguntas que más se encuentran dentro del servicio al cliente de Apple. De lo anterior, es posible verificar que los tópicos más comunes son:

1. Problemas de batería relacionados con una actualización
2. Problemas específicos de una actualización
3. Problemas en general.

Si bien es cierto que las respuestas relacionadas con cada uno de estos tópicos es diferente para la pregunta original, es posible encontrar una forma "normal" de responder a un tipo de pregunta en específico. Basado sobre todo en la capacidad del algoritmo de LDA para encontrar similitudes entre las preguntas y atendiendo a que la tipificación de la respuesta al menos en los primeros intercambios de mensajes de los agentes y los clientes tienden a ser fácilmente esquematizables dentro de patrones similares.

4.2 Desempeño

El desempeño de un chatbot visto desde la perspectiva de una herramienta de cara al cliente puede medirse directamente a través de la satisfacción del cliente respecto del servicio que se le brinda en un portal de atención; según NuaTalker[17], es imprescindible incluir las siguientes 7 métricas:

1. Número total de usuarios alcanzados: Número total de usuarios que son atendidos por la herramienta. Necesaria para poder garantizar escalabilidad y una correcta oferta del servicio.
2. Duración media de la sesión: depende del objetivo del chatbot. En el caso de la presente implementación al ser un tipo de comunicación muy sencilla no sería necesario tener en cuenta el tiempo promedio de la sesión.
3. Sesiones por usuario: Pensada como una forma de medir de manera indirecta la calidad de las respuestas que el cliente esta recibiendo por parte del chatbot.

[17] Recuperado de: https://www.nuatalker.com/como-medir-el-rendimiento-de-tu-chatbot/

4. Interacciones por usuario: Pensada como una forma de medir de manera indirecta la calidad de las respuestas que el cliente esta recibiendo por parte del chatbot.
5. Tiempo de respuesta: Medida de desempeño de la herramienta. En nuestro caso es importante llevar un record de los tiempos de respuesta para poder mejorar la infraestructura y la calidad del modelo que estamos utilizando.
6. Tasas de retención: Pensada como una forma de medir de manera indirecta la calidad de las respuestas que el cliente está recibiendo por parte del chatbot.
7. Tasa de satisfacción: Pensada como una forma de medir de manera indirecta la calidad de las respuestas que el cliente está recibiendo por parte del chatbot.

Sin embargo, por la naturaleza de la implementación, no es posible llevar a cabo la validación en un entorno directo con el cliente por lo que se ha optado por hacer pruebas cruzadas a través de un conjunto de validación. Este conjunto de datos para validación se construye con un 10% de las entradas totales del conjunto original, adicionales al 20% del conjunto de pruebas y el 70% del conjunto de entrenamiento.

Los datos de validación no son utilizados en ninguna parte dentro del proceso de entrenamiento, así como tampoco son deconstruidos en sus vectores en todo el paso de preprocesamiento lo que quiere decir que ninguno de los instrumentos de trabajo para el entrenamiento, preprocesamiento y test utilizan o tienen acceso a estos datos, esto con el fin de evitar fuga de información. Lo anterior puede definirse como el hecho de que el conjunto de datos que se utiliza en el proceso de entrenamiento tiene acceso a los datos de validación, lo cual podría provocar un sobreentrenamiento y dar resultados mucho mejores.

4.3 Evaluación

Las pruebas fueron construidas conforme al ciclo presentado en la figura 11, donde se hicieron pasar cada una de las preguntas dentro del flujo completo de funcionamiento del chatbot para cada uno de los modelos. Adicionalmente, se calculó la similitud entre la respuesta que brinda el modelo y la respuesta original asociada a la pregunta. De esta forma, toda respuesta devuelta por el modelo, con un valor igual o mayor a 0.9 en la similitud de coseno se consideró como una respuesta bien categorizada, es decir, que tiene el valor de 1, mientras que cualquiera con valor menor a 0.9 se le asignó el valor de 0.

La determinación del umbral para la determinación de una respuesta válida o no se llevó a cabo a través de una valorización manual con una muestra de 1000 respuestas evaluadas con dos posibles salidas: 1 en caso de que fuese una respuesta aceptable y 0 en caso negativo. Posteriormente se encontró que en el caso de las 834 respuestas que tuvieron una respuesta aceptable el 91.2 % mantenían un score mayor a 0.9 medida a través de la similitud de coseno.

Una vez predichas las respuestas para el conjunto de pruebas y de su asignación como buena o mala respuesta se construyeron las siguientes métricas:

1. *Recall*: Fracción de instancias relevantes que han sido correctamente clasificadas. La pregunta asociada con esta métrica, sería ¿Qué proporción de positivos reales se identificó correctamente? Y se define como el número de verdaderos positivos entre todos los registros que debieron haberse clasificados como positivos (Verdaderos positivos y Falsos Negativos) [18][14].

$$recall = \frac{TP}{TP + FN}$$

2. *Accuracy*: Busca responder a la pregunta ¿Qué proporción de identificaciones positivas fue correcta? Y se define como el número de verdaderos positivos, entre todas las clasificaciones de positivos, sean verdaderas o no [18].

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

3. *F1 score*: se puede definir como el doble del producto de precisión y recall sobre la suma de ambas. Dicho de otra manera, es la media armónica de estas dos métricas. Lo anterior permite tener una mejor idea del comportamiento sin tener que sacrificar cual de las dos hay que maximizar. En este sentido, siempre esta métrica tendrá que ser una mejor forma de comparar el desempeño de diferentes modelos [14].

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

4. *Precisión*: Representa el número de verdaderos positivos que son realmente positivos en comparación con el número total de valores positivos predichos. Dado que el resultado para todos los modelos, y en este caso invariablemente debido a la naturaleza de la construcción de los sets de las pruebas, el valor de precisión indica que los modelos no generan Falsos positivos[14].

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

Comparando el resultado de todas las preguntas de la muestra para validación con un vector validez en la respuesta de valor 1[18] con tamaño igual al de la muestra, los resultados obtenidos refieren que el modelo que mejor desempeño ha mostrado es el que fue construido con 5 tópicos. El cuadro 4 muestra los resultados para las métricas sobre recall, Accuracy y F1

Número de tópicos	Métrica		
	Recall	**Accuracy**	**F1 Score**
3	0.6	0.6	0.74
4	0.56	0.56	0.71
5	0.64	0.64	0.78
6	0.6	0.6	0.74
7	0.59	0.59	0.74

Cuadro 4. *Métricas de desempeño*
Fuente: Elaboración propia

Cómo podemos observar, las métricas de recall y accuracy son iguales en todos y cada uno de los tópicos que hemos evaluado. Lo anterior puede deberse principalmente a que la cantidad de valores para los falsos positivos y los falsos negativos son siempre los mismos. Sin embargo esto no representa un problema en la medición de la capacidad de cada uno de los modelos para regresar al cliente una respuesta adecuada. Lo anterior debido a que la métrica de *F1 Score* permite establecer una diferencia consistente entre las diferentes aplicaciones.

Para el cálculo del promedio de las diferentes métricas de desempeño, se ha utilizado un micro-promedio que pondera el promedio por clase por la contribución total de esa clase en la población. Lo anterior permite tener un mejor acercamiento al análisis de las métricas teniendo en cuenta el desbalance entre los diferentes tópicos y clases.

[18] Supone que todas las respuestas que brindó el modelo tengan una similitud con la respuesta original mayor a 0.9

Podemos resumir entonces que dados los resultados del valor de F1 para los diferentes modelos con sus tópicos la mejor forma de representar la estructura de los diferentes temas conversacionales es a través de 5 tipos diferentes de tópicos.

4.4 Tiempos de respuesta de interfaz

Los resultados de la aplicación de modelo de 5 tópicos a las conversaciones están supeditados a la respuesta a una pregunta. Esto quiere decir que el chatbot que se ha construido no puede mantener conversaciones entre diferentes interacciones por parte de un mismo cliente, reiterando así su aplicación como un chatbot basado en el modelado de pregunta respuesta.

Lo anterior esquematiza de manera general los ámbitos de aplicación del modelo a la segmentación inicial de un cliente por parte del servicio al cliente. Es decir puede aplicarse de manera inicial previo a la asignación del cliente a un agente. O para intentar reconocer el problema y redirigirlo a una página mucho más robusta de soluciones, dicho de otra manera un *help center*.

A continuación revisaremos algunos de los ejemplos que han sido el resultado de la puesta en marcha del modelo a través de una pregunta sencilla y las respuestas que han obtenido:

Pregunta	Respuesta	Tópico categorizado	Similitud	Tiempo de respuesta
Hi, my Iphone is not working with Apple Music, can you help me?	hi we are here to help to start can you tell us more about what you are experiencing with your iphone and apple music also can you tell us the steps taken when performing that reset	Topic 1	0.92	10 s
Hola me pueden ayudar con mi teléfono? no enciende	we offer support via twitter in english get help in spanish here or join	Topic 1	0.95	5 s
My Battery is discharging himself to quick	we would like to help what kind of iphone do you have amp what ios version is it on check in settings gt general gt about	Topic 2	0.94	2 s
The last update broke my iphone	reach out to us via message dm and let us know what is going we maybe able to help resolve your issue	Topic 3	0.9	10 s
Your shit service is going to make me crazy	we would love to help please click here to join us in dm we will continue there	Topic 4	0.98	7 s

Cuadro 5. *Ejemplo de preguntas y respuestas*
Fuente: Elaboración propia

Como podemos observar, el modelo es incluso sensible al cambio de lenguaje puesto que en el momento en que se realizó una pregunta en español ha indicado que a través de twitter sólo se da servicio al cliente en inglés.

Cómo se mencionó en el capítulo de evaluación es imprescindible referir cuáles son las métricas implicadas en la puesta en producción de una aplicación como la que se propone: chatbot basado en pregunta-respuesta. Para ello se han realizado pruebas con el conjunto de validación para encontrar el tiempo promedio de respuesta por tópico con los siguientes resultados:

Tópico	Similitud Promedio	Tiempo de respuesta (por pregunta)	Conjunto de Búsqueda
Topic 1	0.92	11.5	22,000
Topic 2	0.967	5.1	3,500
Topic 3	0.902	7.3	18,000
Topic 4	0.98	12.4	23,000
Topic 5	0.954	8.34	80,000

Cuadro 6. *Desempeño por tópico*
Fuente: Elaboración propia

De lo anterior, podemos inferir que el desempeño, medido a través del tiempo de predicción del tópico con el modelo LDA, del cálculo de la similitud con cada una de las preguntas pertenecientes a dicho tópico de la implementación está directamente relacionado con el tamaño del conjunto de búsqueda. Además, que así mismo difieren respecto a la similitud promedio de las respuestas dónde es el tópico número cuatro, el que ofrece mejor desempeño y mientras que el tópico 3 es aquel con mejor capacidad para ofrecer respuestas acordes a las preguntas.

Sin embargo lo anterior, no hay que olvidar que se obligó a que las recomendaciones de preguntas se establecieran como aquellas que superaran cierto rango de similitud. Por otro lado, el desempeño de los diferentes modelos difiere por la cantidad de ítems sobre los que tiene que calcular la distancia coseno para una pregunta nueva. Esto modifica claramente la escalabilidad y aplicabilidad del presente proyecto como parte de una infraestructura que pretende ser la primera ventana de atención a los clientes. El tiempo promedio de respuesta para todos los tópicos fue de 8,9 segundos, por lo que es un tiempo

considerablemente alto cuando hablamos de que el objetivo es la automatización de un procedimiento que de otra forma tendría que ser llevado a cabo por un humano. Adicionalmente hay que tener en cuenta que éste primer acercamiento al problema de la automatización de comunicaciones con herramientas de question-answering presenta retos importantes para su mejora, siendo lo más destacable que está pensado como la introducción para segmentación del cliente y su comunicación con un agente humano.

4.5 Aplicación

En este apartado revisaremos los resultados respecto de las opiniones sobre la implementación del chat con interacciones reales con una población de 100 individuos. Derivado de la complejidad para la implementación en un ambiente local[19]; la selección de los usuarios estuvo mayormente enfocada en aquellos con computadoras dónde permitiera la ejecución del proyecto sin interrupciones. Para ello las condiciones obligaban a que los equipos contarán con al menos 8gb de memoria ram y un procesador con al menos 2 cores. Por otro lado, la población que participó en la aplicación del chatbot de dominio específico para atención al cliente tiene la siguiente distribución por sexo:

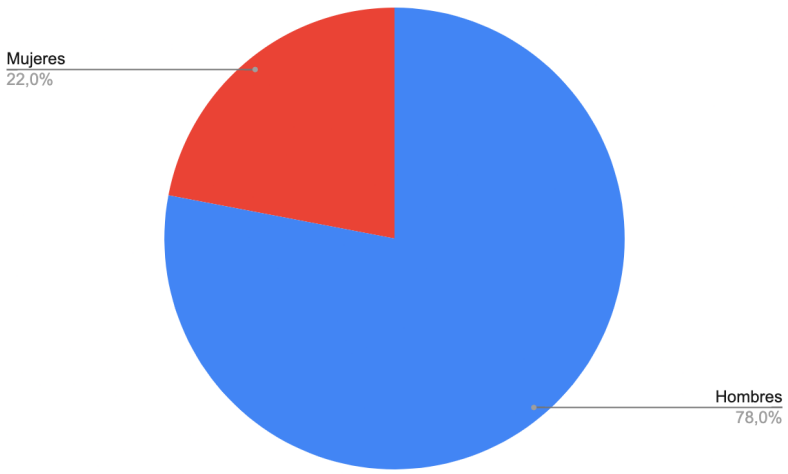

Figura 14. *Distribución por sexo*
Fuente: Elaboración propia

[19] Ambiente de desarrollo local que hace referencia al uso de tecnologías que se encuentran en cualquier máquina de escritorio.

Adicionalmente la distribución por edad confirma que en su mayoría se encuentran entre los 20 y 30 años de edad.

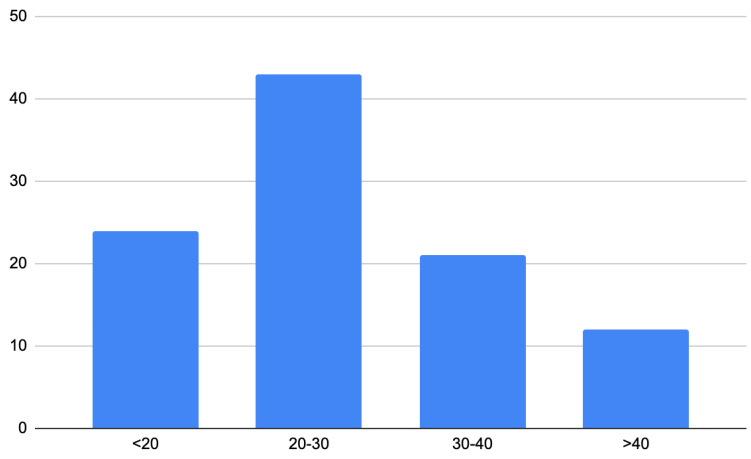

Figura 15. *Distribución por edad*
Fuente: Elaboración propia

El estudio tenía como resultado evaluar el desempeño del chatbot de dominio específico a través de la opinión de si la interacción con el mismo, les había ayudado a resolver el problema con sus dispositivos Apple. Por lo que se solicitó contestarán el siguiente cuestionario:

1. ¿En qué idioma hiciste la consulta con el chat?
2. En una escala del 1 al 10 cúal consideras que es la dificultad para comprender tu pregunta
3. En una escala del 1 al 10 qué tan satisfecho estás con la respuesta del chatbot a tu solicitud
4. ¿Consideras útil la información que te ha brindado?

Los resultados refieren que a pesar de que la mayoría de las preguntas corresponden al idioma en el que fue construido el proyecto (inglés), existe una importante proporción de usuarios que no hablan o escriben en este lenguaje.

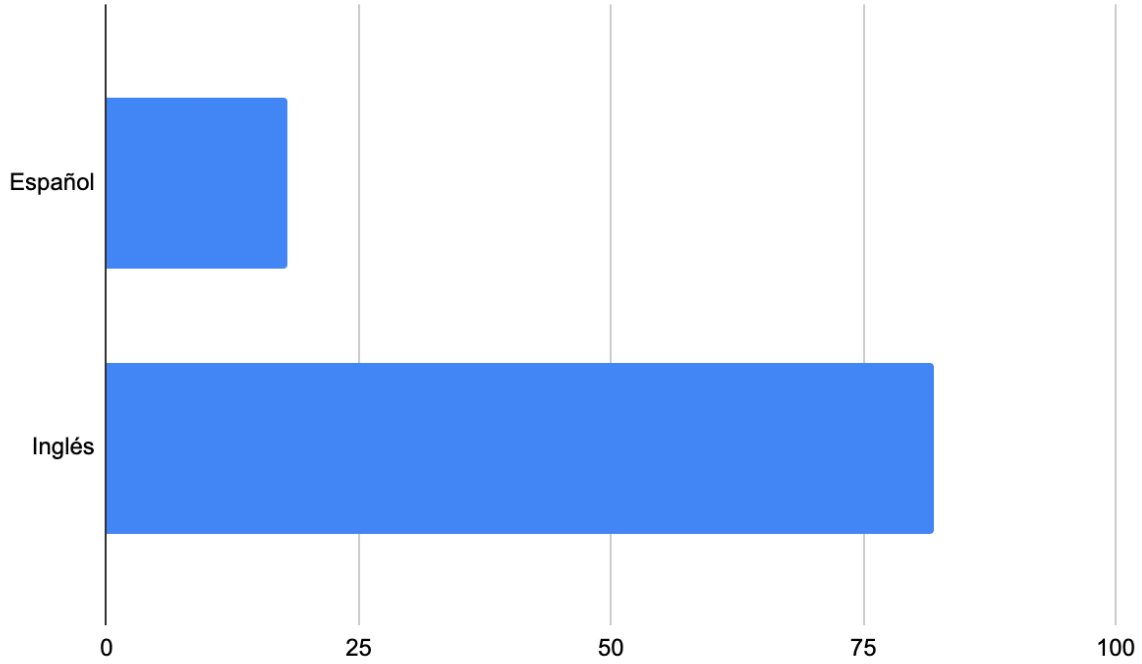

Figura 16. *Distribución por edad*
Fuente: Elaboración propia

Sin embargo lo anterior, los resultados de satisfacción fueron mucho mayores a lo esperado, principalmente resultado de la advertencia que se hace al inicio del proyecto, (this chatbot is only working on English) adicional a la respuesta principal cuando se detecta un idioma diferente: *we offer support via twitter in english, get help in spanish here or join.*

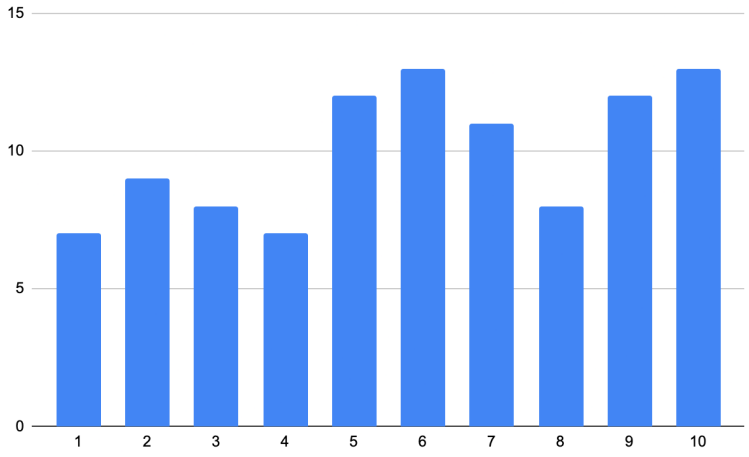

Figura 17. Satisfacción
Fuente: Elaboración propia

Por otro lado, respecto de la dificultad en la que los usuarios consideran su pregunta se encuentran los siguientes resultados:

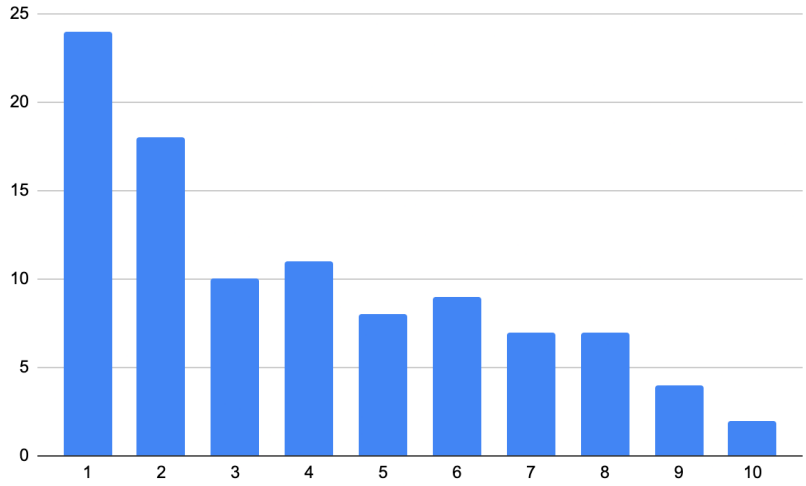

Figura 18. Dificultad de la pregunta
Fuente: Elaboración propia

Por último, la pregunta final sobre la utilidad de la herramienta que promete ser dicotómica, es decir que sólo puede ser respondida con una afirmación o una negación obtuvo los siguientes resultados:

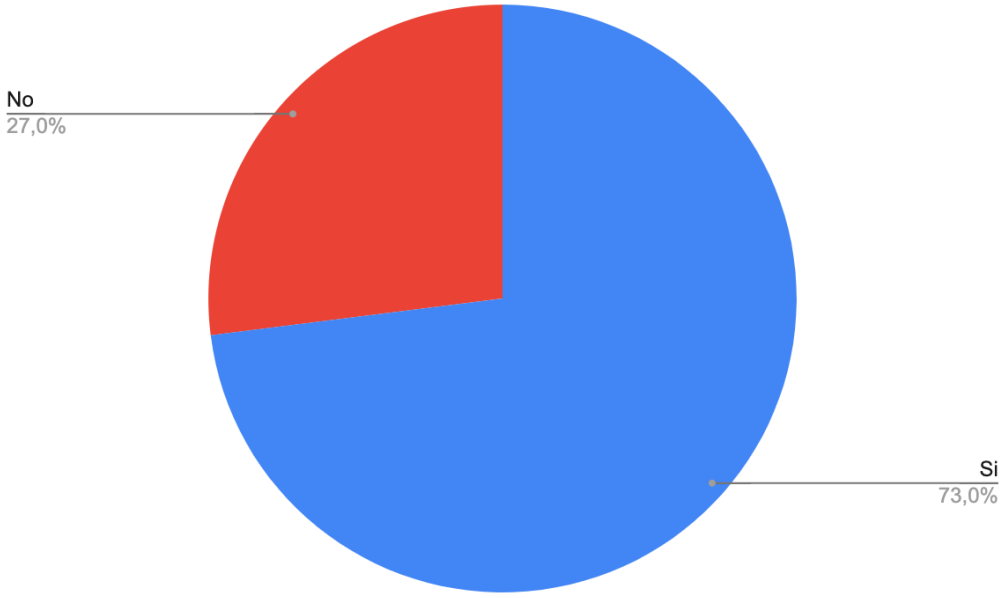

Figura 19. Utilidad
Fuente: Elaboración propia

Por lo que podemos inferir que a pesar de los problemas que se podrían encontrar a la hora de la interpretación de la respuesta brindada por el chat, o por el contrario en el caso de que la pregunta o solicitud realizada sea demasiado compleja para poder ser caracterizada de manera adecuada. Es posible encontrar utilidad en la implementación de este tipo de soluciones.

Conclusiones

Conclusiones

El desarrollo de nuevas tecnologías asociadas a la capacidad de los algoritmos de aprendizaje automático promueve de manera constante el surgimiento de herramientas que nos permiten tener desde una mejora calidad de vida hasta incluso la disminución del trabajo físico que sería necesario para el desempeño de nuestras actividades diarias. Este último caso se acopla a lo desarrollado en el presente trabajo de implementación, donde nuestro principal objetivo ha sido la construcción de una herramienta de inteligencia artificial que permitiera a los agentes de atención al cliente disminuir la cantidad de trabajo necesario para atender a la mayor cantidad de clientes posible.

En este sentido, el resultado de la implementación ha permitido vislumbrar por un lado, la enorme capacidad que tienen los algoritmos de aprendizaje computacional para poner sintetizar problemas básicos llevándolos a cabo, aunque no perfectamente, si de una manera que permite la recolocación de esfuerzos, sobre todo humanos, en las actividades de, por ejemplo, una empresa de tecnología.

De manera mucho más específica, la solución presentada a través de la categorización de preguntas de servicio al cliente en el idioma inglés para su respuesta a través de una propuesta de question-answering ha resultado, por un lado, ser prometedora respecto a su capacidad para solucionar el problema de la categorización de dudas por parte de los clientes y por otro lado gracias a las soluciones de arquitectura en la nube con la que se cuenta actualmente ofrecidas por empresas como Amazon o Google e incluso Microsoft, la posibilidad de escalar una solución para que pueda ser ejecutable sin los principales problemas de la adquisición de capacidad de cómputo, de servidores y de personal específicamente dedicado al mantenimiento de los servicios.

En este sentido, dejaremos para un posterior trabajo la construcción de la arquitectura en la nube para esta solución, dado que el objetivo del presente estriba solo en la implementación a nivel local. Sin embargo, no se deja de lado la

capacidad que tienen, por ejemplo, servicios mayormente conocidos como *serverless* para poder de manera muy sencilla llevar modelos de inteligencia artificial hasta el lado del cliente.

Por último me gustaría recordar que a pesar de que los objetivos se cumplieron, es decir, ha sido posible la construcción de un sistema de question-answering para la categorización de las conversaciones de clientes para una empresa de tecnología ofreciendo una respuesta que pudiera acercarse de manera correcta a una respuesta hecha por un agente de servicio al cliente real; es importante resaltar que aún queda mucho trabajo por delante respecto a la correcta parametrización de esta solución a una aplicación en la vida real, desde la puesta a punto de un modelo que se entrene con un conjunto de datos individual para cada empresa, hasta la contraposición de esta aproximación a otros algoritmos de aprendizaje computacional.

Bibliografía

[1] Abbate, M.L., I. Thiel and T. K. (2005). Can proactive behavior turn chatterbots into conversational agents? in IEEE/WIC/ACM International Conference on Intelligent Agent Technology.
[2] Abu Shawar, B., & Atwell, E. (2007). Chatbots: are they really useful? LDV-Forum: Zeitschrift für Computerlinguistik und Sprachtechnologie, 22(1), 29–49
[3] Accenture, Accenture. (2017) . Chatbots in Customer Service. Online. Recuperado de: https://pdfcoffee.com/qdownload/accenture-chatbots-customer-servicepdf-pdf-free.html
[4] Allison, D. (2012). Chatbots in the library: is it time? Library Hi Tech, 30(1), 95–107.
[5] Altszyler E., et al. (2017). Comparative study of LSA vs Word2vec embeddings in small corpora: a case study in dreams database. URL: https://arxiv.org/pdf/1610.01520.pdf
[6] Blei, David M.; Ng, Andrew Y.; Jordan, Michael I (2003). «Latent Dirichlet allocation» En Lafferty, John, ed. Journal of Machine Learning Research 3 (4–5): pp. 993-1022.
[7] Definición Consultada en: https://en.wikipedia.org/wiki/Pandorabots el dia 19 de Junio de 2020
[8] Definición consultada en: https://es.wikipedia.org/wiki/AIML el dia 19 de Junio de 2020
[9] Dipanjan S. (2018). Online. Recuperado de: https://www.kdnuggets.com/2018/04/implementing-deep-learning-methods-feature-engineering-text-data-cbow.html.
[10] Dorado Alfaro, S, (2017) UAM. Departamento de Ingeniería Informática. Análisis e implementación de diferentes medidas de similitud para un algoritmo global de selección de variables. Recuperado de: https://repositorio.uam.es/handle/10486/679332
[11] Estudio y uso de chatbots para el aprendizaje de inglés. Rodriguez Roa S. (SN). Escuela Técnica Superior de Ingeniería y Sistemas de Telecomunicación. Recuperado de: https://oa.upm.es/43327/3/TFG_SERGIO_RODRIGUEZ_ROA.pdf
[12] Extraido de : https://es.wikipedia.org/wiki/Word_embedding Consultado el 19 de Junio de 2020
[13] Floria, P. (2017). Master thesis : Design and implementation of a chatbot in the context of customer support.
[14] Forman G. (2003). An Extensive Empirical Study of Feature Selection Metrics for Text Classification. Journal of Machine Learning Research 3 (2003) 1289-1305. https://www.jmlr.org/papers/volume3/forman03a/forman03a_full.pdf
[15] Gao, J., Galley, M., Li, L. (2019), Neural approaches to Conversational AI. Online.
[16] Girolami, Mark; Kaban, A. (2003). On an Equivalence between PLSI and LDA (http://www.cs.bh am.ac.uk/~axk/sigir2003_mgak.pdf). Proceedings of SIGIR 2003. New York: Association for Computing Machinery.

[17] IBM, Research Staff. (2011). Knowing what it knows: selected nuances of Watson's strategy. Consultado el 19 de Junio de 2020 en: https://www.ibm.com/blogs/research/2011/02/knowing-what-it-knows-selected-nuances-of-watsons-strategy/
[18] Juba, B., & Le, H. S. (2019). Precision-Recall versus Accuracy and the Role of Large Data Sets. Proceedings of the AAAI Conference on Artificial Intelligence, 33(01), 4039-4048. https://doi.org/10.1609/aaai.v33i01.33014039
[19] Lei Y. and Huan L. (2003) Feature selection for high-dimensional data: A fast correlation-based filter solution. In ICML, volume 3, pages 856–863.
[20] Lemaitre, C., C.A. Reyes, and Gonzalez, J. (2004). Advances in Artificial Intelligence -IBERAMIA 2004. in 9th Ibero-American Conference on AI, Puebla, November 22-26. México
[21] Morgan, B. (2017). How Artificial Intelligence will impact The Insurance Industry. Forbes.
[22] Pengfei L, Xipeng Q., Xuanjing H., (2016) Recurrent Neural Network for Text Classification with Multi-Task Learning.
[23] Peters, F. (2018). Design and implementation of a chatbot in the context of customer support.
[24] Phd, Mohammad & Hussain, O. (2018). A Survey on Chatbot Implementation in Customer Service Industry through Deep Neural Networks. 54-61.
[25] Recuperado de : https://en.wikipedia.org/wiki/Latent_semantic_analysis. Consultado el Lunes 13 de Julio de 2020.
[26] Recuperado de : https://es.wikipedia.org/wiki/Red_neuronal_recurrente Consultado el lunes 13 de Julio de 2020.
[27] Recuperado de: https://es.wikipedia.org/wiki/Similitud_coseno. Consultado el Lunes 16 de noviembre de 2020
[28] T. Mikolov et al. (2013). "Efficient Estimation of Word Representations in Vector Space", URL : https://arxiv.org/pdf/1301.3781.pdf
[29] Tomas M, Greg C, Kai C, and Jeffrey D. (2013). Efficient Estimation of Word Representations in Vector Space. Proceedings of the International Conference on Learning Representations (ICLR 2013), pages 1–12,.
[30] Weizenbaum, J. (1966). "ELIZA – A Computer Program For the Study of Natural Language Communication Between Man and Machine.
[31] Wiemer-Hastings, P. (2004) Latent Semantic Analysis.. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.91.6171&rep=rep1&type=pdf

ANEXOS

ANEXOS

En el caso de los anexos por simplicidad y espacio sólo se incluirán los scripts de entrenamiento así como los *utils*.

Utils Class

```
import string
from gensim import corpora
from nltk.stem.wordnet import WordNetLemmatizer
from nltk.corpus import stopwords, wordnet
from sklearn.feature_extraction.text import CountVectorizer
import numpy as np
from sklearn.decomposition import LatentDirichletAllocation as LDA

import warnings
warnings.simplefilter("ignore", DeprecationWarning)

import numpy as np

class Cleaner(object):
    def __init__(self):
        self.punctuation = set(string.punctuation)

        self.stoplist = set(stopwords.words('english'))

        self.dictionary = corpora.Dictionary()
        self.lemma = WordNetLemmatizer()

    def remove_punctuation(self, text):
        return ''.join([char for char in text if char not in self.punctuation])

    def remove_numbers(self, text):
        return ''.join([char for char in text if not char.isdigit()])

    def remove_stopwords(self, text):
        return ' '.join([word for word in text.split() if word not in self.stoplist])

    def remove_single_chars(self, text):
        return ' '.join([word for word in text.split() if len(word) > 1])

    def lemmatize(self, text):
```

```
        return ' '.join([self.lemma.lemmatize(word) for word in text.split()])

    def clean_text(self, text):
        text = text.replace('\n', '')
        text = self.remove_punctuation(text)
        text = self.remove_numbers(text)
        text = self.remove_stopwords(text)
        text = self.remove_single_chars(text)
        text = self.lemmatize(text)
        return text
```

Train LDA Model

```
import nltk
import ssl

import warnings
warnings.simplefilter("ignore", DeprecationWarning)

try:
    _create_unverified_https_context = ssl._create_unverified_context
except AttributeError:
    pass
else:
    ssl._create_default_https_context = _create_unverified_https_context

import string
from gensim import corpora
from nltk.stem.wordnet import WordNetLemmatizer
from nltk.corpus import stopwords, wordnet
from sklearn.feature_extraction.text import CountVectorizer
import numpy as np
from sklearn.decomposition import LatentDirichletAllocation as LDA
import pandas as pd

from utils.utils import Cleaner
import pickle


class LDA_model():

    train_data = None
    test_data = None
    def __init__(self, lista):
        self.count_vectorizer = CountVectorizer(stop_words = 'english')
```

```
        self.lista = lista

    def transform(self, number_topics = 5, number_words = 5):
        self.count_data = self.count_vectorizer.fit_transform(self.lista)
        self.number_topics = number_topics
        self.number_words = number_words
        return self

    def topics(self, model,count_vectorizer, n_top_words):
        words = self.count_vectorizer.get_feature_names() # El objeto words es el corpus de
todos los textos.
        dic = {}
        for topic_idx, topic in enumerate(model.components_):
            dic.update({topic_idx:" ".join([words[i] for i in topic.argsort()[:-n_top_words -
1:-1]]) })
        return dic


    def fit(self):
        self.transform()
        lda = LDA(n_components = self.number_topics, n_jobs = 1)
        lda.fit(self.count_data)
        self.model= lda
        self.dic_topics =  self.topics(lda,self.count_vectorizer , self.number_words)
        return self

    def predict(self, text):
        X_test = self.count_vectorizer.transform(text)
        prueba = self.model.transform(X_test)
        salida  = prueba.argmax(axis = 1)
        dic = {"Predict Topic": {"index":salida[0], "text": self.dic_topics[salida[0]]}}
        return dic


if __name__ == "__main__":
    data = pd.read_csv("data/final_data.csv")
    data = data[["ID", "question", "response", "empresa"]]
    data.dropna(subset = ["question"], inplace=True)
    data.drop_duplicates(subset = ['ID'], keep = 'first', inplace = True)
    splitter = int(0.8 * (len(data)))
    train_data = data[:splitter]
    test_data = data[splitter:]
    modelo = LDA_model(lista =train_data["question"])
    modelo.fit()
    modelo.train_data = train_data
    modelo.test_data = test_data
    with open("models/lda_model.pkl", "wb") as file:
```

```
    pickle.dump(modelo, file)
```

Train LSA Model

```
import nltk
import ssl
import pandas as pd
import numpy as np

import warnings
warnings.simplefilter("ignore", DeprecationWarning)

try:
    _create_unverified_https_context = ssl._create_unverified_context
except AttributeError:
    pass
else:
    ssl._create_default_https_context = _create_unverified_https_context


from nltk.corpus import stopwords
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.decomposition import TruncatedSVD
from utils.utils import Cleaner
import matplotlib.pyplot as plt
import pickle

class LSA_model():
    def __init__(self, data = None):
        """
        Print ("Hola")
        """
        if data == None:
            data = pd.read_csv("data/final_data.csv")[["question", "response"]]
        assert len(data) > 0 , "Can't find data"
        data.dropna(subset = ["question"], inplace = True)
        data.drop_duplicates(subset = ['ID'], keep = 'first', inplace = True)

        splitter = int(0.8 * (len(data)))
        self.train_data = data[:splitter]
        self.test_data = data[splitter:]
        self.vectorizer = CountVectorizer(stop_words = 'english')

    def topics(self, model, n_top_words):
        words = self.vectorizer.get_feature_names()
```

```
        dic = {}
        for topic_idx, topic in enumerate(model.components_):
            dic.update({topic_idx:" ".join([words[i] for i in topic.argsort()[:-n_top_words -
1:-1]]) })
        return dic

    def transform(self, data = None, number_words: int = 5):
        """
            Get document_term_matrix
        """
        responses = self.train_data.response
        questions = self.train_data.question
        self.number_words = number_words
        X = self.vectorizer.fit_transform(questions)
        print("Final DataFrame Shape: ", X.shape)
        self.X = X
        return self

    def fit(self, number_of_topics: int = 5):
        lsa_model = TruncatedSVD(n_components=number_of_topics,
algorithm='randomized', n_iter=400, random_state=122)
        lsa_model.fit(self.X)
        self.model = lsa_model
        self.dic_topics =  self.topics(self.model, self.number_words)
        return self

    def predict(self, text):
        X_test = self.vectorizer.transform(text)
        prueba = self.model.transform(X_test)
        salida  = prueba.argmax(axis = 1)
        dic = {"Predict Topic": {"index":salida[0], "text": self.dic_topics[salida[0]]}}
        return dic

if __name__ == "__main__":
    model = LSA_model()
    model.transform()
    model.fit(number_of_topics= 3)
    print(model.dic_topics)
    print(model.predict(["I i have a problem with my battery"]))

    with open("models/lsa_model.pkl", "wb") as file:
        pickle.dump(model, file)
```

Train SVM Model

```
import nltk
import ssl

import warnings
warnings.simplefilter("ignore", DeprecationWarning)
try:
    _create_unverified_https_context = ssl._create_unverified_context
except AttributeError:
    pass
else:
    ssl._create_default_https_context = _create_unverified_https_context

import string
from gensim import corpora
from nltk.stem.wordnet import WordNetLemmatizer
from nltk.corpus import stopwords, wordnet
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.decomposition import LatentDirichletAllocation as LDA
from sklearn import svm

import pandas as pd
import numpy as np
from utils.utils import Cleaner
import pickle


class SVM_model():
    def __init__(self, lista):
        self.count_vectorizer = CountVectorizer(stop_words = 'english')
        self.lista = lista

    def transform(self, number_topics = 5, number_words = 5):
        self.count_data = self.count_vectorizer.fit_transform(self.lista)
        self.number_topics = number_topics
        self.number_words = number_words
        return self

    def topics(self, model,count_vectorizer, n_top_words):
        words = self.count_vectorizer.get_feature_names()
        dic = {}
        for topic_idx, topic in enumerate(model.components_):
            dic.update({topic_idx:" ".join([words[i] for i in topic.argsort()[:-n_top_words -
1:-1]]) })
        return dic
```

```
if __name__ == "__main__":
    data = pd.read_csv("data/final_data.csv")
    data = data[["ID", "question", "response", "empresa"]]
    data.dropna(subset = ["question"], inplace=True)
    data.drop_duplicates(subset = ['ID'], keep = 'first', inplace = True)

    modelo = SVM_model(lista =data["question"])
    modelo.fit()
    with open("models/svm_model.pkl", "wb") as file:
        pickle.dump(model, file)
```

To Execute chatbot

```
import pandas as pd
from train_model_LDA import LDA_model
from train_model_LSA import LSA_model
from sklearn.metrics.pairwise import cosine_similarity
import numpy as np
import ast

class chat(object):
    def __init__(self, model: str = "LDA", topics: int = 5):
        import fasttext
        import pickle
        self.model = model
        self.topics = topics
        self.model_vec = fasttext.load_model("cc.en.300.bin")
        self.table_vectors = pd.read_csv(f"datasets/table_vectors_{self.model}.csv")
        if self.model == "LDA":
            modelo = f"models/lda_model_{self.topics}.pkl"
        if self.model == "LSA":
            modelo = f"models/lsa_model_{self.topics}.pkl"

        with open(modelo, "rb") as file:
            self.model = pickle.load(file)
        print("""
        Chat Listo.
        Solo permite comunicación en Inglés
        Para terminar la conversación pulsa en cualquier momento "Adios"
        """)
        try:
            print("Características del modelo utilizado")
            print("Entrenamiento")
            print(f"Test data shape: {self.model.test_data.shape}")
```

```
            print(f"Train data shape: {self.model.train_data.shape}")
            print(self.model.train_data.shape)
        except Exception as e:
            print(f"Can not find Test and train data due to: {e}")
        print("Hi! How i can Help  you?")
        self.question = input()
        while self.question != "Adios":
            self.__get_question_related(self.question)
            self.__get_responses_related()
            self.__get_main_table()
            self.__get_answer()
            self.question = input()

    def __get_question_related(self,question):
        self.question_vec = self.model_vec.get_sentence_vector(question)
        self.question_top = self.model.predict([question])["Predict Topic"]["index"]
        return self

    def __get_responses_related(self):
        self.indexes = [i for i, x in enumerate(self.table_vectors["topics"]) if x
==self.question_top]
        # utilizar numpy para las comparaciones.
        vectores = self.table_vectors["vectores"]
        self.vectors = vectores[self.indexes]

    def __get_main_table(self):
        """
        La tabla tarda demasiado en ser construida. Porque calcula la distancia con cada una
de las posibles respuestas dentro del tópico.
        """
        lista = []
        for i in self.vectors:
            i = np.asarray(ast.literal_eval(i))
            lista.append(cosine_similarity(self.question_vec.reshape(1, -1), i.reshape(1,
-1))[0][0])
        tabla = pd.concat([pd.DataFrame({"lista":lista}), pd.DataFrame({"indices":
self.indexes})], axis = 1)
        tabla.sort_values("lista", ascending = False, inplace = True)
        self.tabla = tabla.head(10)

    def __get_answer(self):
        self.responses = self.table_vectors["responses"]
        respuesta  = self.tabla["indices"].iloc[0]
        respuesta = self.responses[respuesta]
        print(respuesta)
```

```
if __name__ == "__main__":
    chat(model = "LDA", topics = 4)
```