



**UNIVERSIDAD  
DE ANTIOQUIA**

**IMPLEMENTACIÓN DE UN SISTEMA DE DIÁLOGO  
AUTOMÁTICO COMO ASISTENTE EN EL PROCESO  
ADMINISTRATIVO DEL EXAMEN DE TRADUCTOR E  
INTÉRPRETE OFICIAL DE LA UNIVERSIDAD DE  
ANTIOQUIA**

Autor  
Maria Adriana Ruiz Osorio

Universidad de Antioquia  
Facultad de Ingeniería, Departamento de Ingeniería de Sistemas  
Medellín, Colombia  
2020



**Implementación de un sistema de diálogo automático como asistente en el proceso administrativo del examen de traductor e intérprete oficial de la Universidad de Antioquia**

**Maria Adriana Ruiz Osorio**

Tesis presentada como requisito parcial para optar al título de:  
**Ingeniera de Sistemas**

Asesor:

Antonio Jesus Tamayo Herrera  
Magíster en Ingeniería e Ingeniero de Sistemas

Línea de Investigación:

Procesamiento de Lenguaje Natural

Grupo de Investigación:

Traducción y Nuevas Tecnologías (TNT)

Universidad de Antioquia

Facultad de Ingeniería, Departamento de Ingeniería de Sistemas

Medellín, Colombia

2020

## Resumen

Los modelos, técnicas y herramientas tecnológicas con las que se cuentan en la actualidad han marcado el avance proporcionando grandes oportunidades de cambio en diferentes campos de la industria. Para el caso del presente trabajo, se propuso aprovechar estas tecnologías emergentes para el desarrollo de un sistema de diálogo que permitiera suplir la necesidad de dar respuesta de manera ágil y oportuna a las inquietudes de los interesados en realizar el examen de traductor e intérprete oficial de la Universidad de Antioquia. La problemática fué identificada dado que las preguntas concernientes a este examen son realizadas telefónicamente o enviadas por correo electrónico y la respuesta no se dá de manera inmediata, además, se debe responder constantemente a las mismas preguntas. De esta manera, en la búsqueda de una solución a esta problemática, se encontró que esta tarea podía ser realizada por un agente virtual como sistema de interacción hombre - máquina. En el desarrollo de esta solución, después de validar con diferentes modelos de representación, métodos y algoritmos de recuperación de información, se obtuvo, usando *machine learning*, más concretamente con el modelo de Regresión Logística, la representación *Term Frequency - Inverse Document Frequency* (TF-IDF) y Django como el framework para el desarrollo de una aplicación web, la mejor opción para dar solución a esta necesidad, creando un sistema de diálogo con un porcentaje de acierto del 88% en las respuestas.

## Introducción

A través de la historia, las herramientas tecnológicas han permitido evolucionar en muchos campos, convirtiéndose en una solución a muchas problemáticas de la sociedad. De esta manera, las empresas aprovechan los beneficios que proveen dichas herramientas para optimizar sus procesos y brindar una mayor satisfacción a sus clientes y usuarios. Por esta razón se han creado metodologías, sistemas y herramientas que permiten brindar una respuesta ágil y oportuna a las inquietudes de los usuarios.

La inteligencia artificial (IA) provee técnicas que son utilizadas en el tratamiento del lenguaje natural, las cuales permiten, tanto desde el enfoque computacional como del lingüístico, investigar cómo el lenguaje puede ser utilizado para cumplir diferentes tareas (Sosa, 1997). Un ejemplo de ello es la aplicación de modelos de diálogo como forma de comunicación entre dos o más personas, que incorporados en un sistema informático permiten la interacción hombre-máquina, usando el habla (o texto) para la solución de diferentes problemas cotidianos como responder de forma oportuna a las dudas presentadas por los usuarios para adquirir cualquier producto o servicio (Zapata y Mesa, 2009).

Con el pasar del tiempo, en la búsqueda de automatizar y mejorar los procesos, se han implementado sistemas de diálogo, basados en preguntas y respuestas, que combinando información de dominios específicos y técnicas de programación, dieron origen a la creación de los Chatbot, programas informáticos diseñados para mantener conversaciones prolongadas llegando a imitar diálogos no estructurados como el que se da entre humanos (Jurafsky y Martin, 2017).

La Escuela de idiomas de la Universidad de Antioquia, realiza el examen de traductor e intérprete oficial para los interesados en certificarse como traductores e intérpretes oficiales

de conformidad con la legislación colombiana (Ley 962 de 2005). En este contexto, los sistemas de diálogo se presentan como una solución automática para tratar la gran cantidad de información que surge de la interacción telefónica y vía email que se sostiene a diario con los interesados en presentar el examen. Esta información, consignada en gran parte como un conjunto de preguntas y respuestas no estructuradas, presenta patrones muy similares que dan pie a proponer una solución computacional para optimizar el proceso de asesoría a los interesados en esta certificación.

Este trabajo muestra la construcción de un sistema de diálogo que permite que los usuarios interesados en el examen de traductor e intérprete oficial de la Universidad de Antioquia puedan mantener una conversación con este sistema para encontrar respuesta a todas sus inquietudes. Su implementación es en un ambiente real y exclusivamente para la Universidad de Antioquia, específicamente la Escuela de Idiomas que es la que ofrece dicho curso, cualquier persona puede realizar las preguntas concernientes a este tema ya que no requiere registro.

Los usuarios pueden realizar sólo preguntas que tienen que ver con el examen de traductor e intérprete oficial de la Universidad de Antioquia y la interacción no puede ser grupal, es decir, no se tiene la opción de unir más usuarios a la conversación ya que el chatbot está programado para ir respondiendo a cada pregunta por usuario y en la medida en que se van realizando.

En la metodología de desarrollo del chatbot, se implementó un enfoque basado en corpus, donde se definió la respuesta ante cada pregunta, teniendo así repositorios de preguntas y respuestas predefinidas que con algoritmos de recuperación de información se logra elegir la respuesta adecuada de acuerdo al contexto.

En la actualidad el uso de agentes conversacionales es muy común, pues se han convertido en instrumento de múltiples disciplinas. Por ejemplo, en la medicina se encuentran muchas aplicaciones de chatbots que aportan en diferentes áreas, como realizar autodiagnósticos de enfermedades de acuerdo a los síntomas presentados por el paciente, como es expuesto por Divya, Indumathi, Ishwarya, Priyasankari y Devi (2018), donde se utiliza la inteligencia artificial para diagnosticar la enfermedad y proporcionar detalles de la misma antes de consultar a un médico, reduciendo de esta manera los costos en la atención médica. Del mismo modo, se encuentran aplicaciones de estos sistemas de diálogo en la Pediatría, donde se pretende guiar a los padres en cuanto a la medicación de los niños. Comendador, Francisco, Medenilla y Mae (2015) en el artículo “Pharmabot: A Pediatric Generic Medicine Consultant Chatbot” presentan un chatbot como consultor de medicina genérica, que tiene la función de prescribir, sugerir y brindar información sobre medicamentos genéricos para niños. Esta nueva revolución tecnológica que propone un cambio de paradigma donde el usuario empieza a relacionarse con una máquina de la misma forma en que lo haría con un ser humano, a través de la escritura, representa un gran aporte en el tema de innovación y automatización de los procesos en las organizaciones que buscan brindar un mejor servicio o producto, como es el caso del presente trabajo, donde se pretende ser partícipes de esta nueva era, en la que para ser competitivos hay que estar a la vanguardia de los nuevos tiempos.

## Objetivos

### Objetivo general

Construir un sistema de diálogo como solución informática para atender, en tiempo real, la mayor cantidad posible de inquietudes de los interesados en presentar el examen de traductor e intérprete oficial ofrecido por la Escuela de idiomas de la Universidad de Antioquia.

### Objetivos específicos

- Representar bajo enfoques lingüísticos y computacionales los datos no estructurados disponibles para el proyecto.
- Evaluar diferentes modelos computacionales para la construcción de sistemas de diálogo usando las diferentes representaciones de los datos.
- Construir el backend de la solución usando el modelo que presente mejores resultados.
- Construir el frontend de la aplicación usando tecnologías de vanguardia en el desarrollo web.
- Desplegar la solución construida en un servidor de aplicaciones.

## Marco Teórico

A través del tiempo, la implementación de nuevas tecnologías en las organizaciones se ha convertido en una necesidad fundamental, ya que les permite estar a la vanguardia de los nuevos tiempos, con procesos competitivos que permitan ofrecer cada vez un mejor servicio a sus clientes.

A través de las tecnologías de la información es posible formular estrategias innovadoras para la modernización de los servicios informáticos en una organización, teniendo en cuenta la calidad y oportunidad del servicio ofrecido. En relación a este tema, muchos autores proponen la implementación de estas herramientas dentro de los procesos de las organizaciones, como es el caso de Cárdenas y Bautista (2012), quienes afirman:

*“Teniendo en cuenta la necesidad de alinear el negocio con el uso de las TI, se requiere optimizar la tecnología en cualquiera de los niveles de la organización a fin de mantener procesos eficientes y de esta manera crecer a costes razonables y predecibles. Uno de los grandes problemas que se enfrentan hoy en lo que a la prestación de servicios se refiere, es el de garantizar la prestación de un servicio conforme a las necesidades del usuario sin incurrir en costes o esfuerzos adicionales”* (p.49). En este sentido, las herramientas tecnológicas actuales permiten dar un valor agregado a las organizaciones que están siempre dispuestas a priorizar la calidad y el servicio, incorporando estos avances tecnológicos en sus procesos.

En la interacción constante entre usuarios y centros de consulta para el acceso a la información, se presenta una comunicación o diálogo entre dos o más personas que puede caracterizarse de acuerdo a su estructura y temática, lo que puede permitir la identificación de modelos de diálogo. Según Zapata y Mesa (2009), los modelos de diálogo son construidos de acuerdo a técnicas basadas en la identificación y clasificación de los actos del habla, por lo



que definen los modelos de diálogo como *“esquemas que tratan de clasificar y caracterizar los diferentes actos del habla, que se pueden encontrar en un diálogo, y determinar las posibles interrelaciones que poseen”* (p.1). Es así como se llega a determinar la estructura del discurso de acuerdo a sus componentes ya que *“en un diálogo cualquiera, la forma en que los interlocutores comunican sus deseos, intenciones y creencias se caracteriza mediante el uso de los actos del habla, los cuales se clasifican en: afirmaciones, preguntas, directrices, respuestas, promesas y declaraciones, entre otras”* (p.1).

Los modelos de diálogo pueden ser aplicados en sistemas informáticos que permiten la interacción hombre-máquina, como lo afirma Zapata y Mesa (2009): *“La construcción de los modelos de diálogo tiene como fin el desarrollo de sistemas de diálogo, que son capaces de interactuar con interlocutores humanos por medio del habla.”* De esta manera, con la creciente demanda para el acceso a la información, los sistemas de diálogo proveen una mayor facilidad para esta interacción cumpliendo con varias características que permiten posicionarlos en diferentes campos, de acuerdo a la necesidad. Estos sistemas se dividen en sistemas pregunta – respuesta y sistemas de diálogos intencionales. El sistema pregunta – respuesta se caracteriza por dar respuestas a preguntas sencillas del interlocutor como por ejemplo preguntas cerradas, en cambio, los sistemas de diálogos intencionales son sistemas más completos ya que tienen un acercamiento al diálogo que se establece entre humanos, donde, *“se presenta la posibilidad de ejecutar otras acciones diferentes a la simple interacción entre una pregunta del interlocutor y una respuesta del sistema, mostrando, de esta forma, la intencionalidad que caracteriza a este tipo de sistemas”* (Zapata y Mesa, 2009, p.1).

Jurafsky y Martin (2017) en el capítulo 27, exponen dos paradigmas basados en los sistemas de diálogo pregunta-respuesta; el primero es llamado respuesta a preguntas basadas

en un sistema de recuperación de información, donde de acuerdo a la pregunta del usuario, las técnicas de recuperación de información extraen pasajes tomados de una cantidad de información disponible en la web guiados por el texto de la pregunta. Este método procesa la pregunta para determinar el probable tipo de respuesta, usando entidades como persona, localidad, tiempo para determinar consultas que son enviadas al motor de búsqueda permitiendo clasificar pasajes relacionados para finalmente extraer las cadenas de respuesta de los pasajes clasificados.

Por otro lado, está el paradigma respuesta a preguntas basadas en el conocimiento, donde se responde a una pregunta en lenguaje natural, a través de una consulta sobre una base de datos estructurada. De esta manera, se tienen dos modelos, uno que trabaja sobre bases de datos estructuradas y otro que realiza las consultas sobre datos de la web, que no tienen estructura alguna.

Según Jurafsky y Martin (2017) en el capítulo 28, los agentes de diálogo se dividen en dos clases: agentes de diálogo orientados a tareas que como su nombre lo dice son diseñados para una tarea en particular ya que se configuran para tener conversaciones cortas y los chatbots que son sistemas diseñados para sostener conversaciones extensas, caracterizados por tener una interacción similar a la de humano – humano. De esta manera, de acuerdo a la necesidad que se tenga se hace el entrenamiento requerido, como lo afirma Tolosa y Bordignon (1999), *“Los agentes cumplen con los requerimientos para los cuales fueron entrenados. El usuario “delega” en el agente una o varias tareas que debe llevar a cabo quedando a la espera de los resultados. Dichas tareas son a menudo fáciles de especificar, pero - en algunos casos – complejas de realizar”* (p.1). Es así como en la actualidad se tienen implementaciones reales de agentes de diálogo o chatbots que cumplen tareas específicas llegando a convertirse en una herramienta vital para el manejo de la información, como es el

caso de las redes sociales, donde cada vez se tiene mayor demanda de usuarios, por lo que se tiene la necesidad de introducir aplicaciones que permitan nuevas formas de interacción. Griol, Callejas, Arroyo, Patricio, Molina y López (2010), muestran un trabajo donde se integran estos sistemas de diálogo en las redes sociales, que consiste en la utilización de un sistema de diálogo que proporciona información académica para crear un avatar conversacional dentro del mundo virtual de Second Life<sup>1</sup>, ofreciendo de esta manera la posibilidad de socializar, explorar, conocer otros usuarios además del acceso a recursos educativos y culturales. Por otro lado, existen muchas tesis de grado que muestran cantidad de implementaciones de los chatbot; una de ellas propone la integración de tres sistemas de diálogo en uno solo, como lo hace Cobos (2013), donde se busca que el chatbot resultado pueda manejar diferentes temas de conversación, caracterizado como la habilidad de un robot social que dialoga en castellano y que es desarrollado en base a sistemas de diálogo reconocidos. En el campo educativo, estos sistemas de diálogo pueden aportar de manera positiva, como lo afirman García, Fuertes y Molas (s.f): *“Saber en qué aula se va a dar la clase siguiente, entender la diferencia entre dos conceptos, practicar una lengua chateando por el teléfono móvil. Estas son solo algunas de las funcionalidades de los chatbots cuando son utilizados en el campo educativo para hacer acompañamiento a los estudiantes”* (p.5).

Esto es sólo un acercamiento a lo que puede lograrse, pues la educación se basa en la comunicación e interacción con herramientas y los chatbots tienen un potencial educativo importante precisamente por su capacidad comunicativa mediante lenguaje natural. De la misma manera, es posible encontrar aplicaciones de sistemas de diálogo que cubren necesidades en la parte administrativa, como es el caso de una empresa de seguros, donde se implementa un chatbot para la gestión de requerimientos e incidentes presentados, con la

---

<sup>1</sup> “Second Life (SL) es un mundo virtual tridimensional desarrollado por Linden Lab en 2003 y accesible a través de Internet” (Griol et al., 2010, p.108).

finalidad de mejorar en la calidad del servicio y automatizar los tiempos de respuesta (Estrada, 2018). Por otro lado, las instituciones educativas también están a la vanguardia de estas nuevas tecnologías, como lo exponen Cambizaca y Francisco (2018), donde se implementa un sistema de diálogo para dar respuesta a las inquietudes presentadas por los usuarios sobre las becas que ofrece la universidad.

Continuando con la exploración de las implementaciones de los sistemas de diálogo, se encuentran también muchas aplicaciones en el campo de la salud. En este sentido, Divya et al. (2018) muestran un maravilloso trabajo, que teniendo en cuenta la crisis de la salud en la actualidad, brinda un gran aporte ya que proponen un chatbot para diagnosticar las enfermedades de los usuarios de acuerdo a sus síntomas, realizando una clasificación de acuerdo a si es una enfermedad mayor o menor para así referir el médico adecuado. También en cuanto a los niños, en el tema de la salud, se presenta un chatbot que orienta en la medicación de los mismos, en cuanto a la medicina genérica, aporte de Comendador et al. (2015): *“One of the objectives of the system is to develop a software application that will assist the user in taking the right generic medicine for a certain ailment”* (pp.2-3). La implementación de estos sistemas de diálogo en las compañías del sector salud trae grandes beneficios tanto para la empresa como para los usuarios, ya que los procedimientos se hacen más rápidos y asertivos, especialmente en un tema tan importante y delicado como lo es la salud.

En la industria financiera, una de las más interesadas en incorporar en los últimos tiempos la mejora en el servicio al cliente mediante los chatbots, se encuentra también muchas implementaciones, una de ellas es tratada por Dole, Sansare, Harekar y Athalye (2015), donde a través de un chatbot inteligente se puede dar respuesta a todas las inquietudes de los usuarios referente al banco, usando la inteligencia artificial. Este chatbot es basado en la web,

utiliza un modelo generativo, es decir, genera las respuestas de forma personalizada y aquellas preguntas que no son comprendidas por el chatbot, son reprocesadas por un sistema experto para luego archivar las respuestas obtenidas y ser utilizadas en la generación de respuestas. Del mismo modo, Kulkarni, Bhavsar, Pingale y Kumbhar (2017), en el artículo “BANK CHAT BOT – An Intelligent Assistant System Using NLP and Machine Learning”, proponen un chatbot donde se utiliza Procesamiento del lenguaje natural y modelos de aprendizaje automático. A diferencia del chatbot presentado por Dole et al. (2015), este utiliza un modelo basado en recuperación, donde se tienen repositorios predefinidos de respuestas, e implementan modelos de aprendizaje automático para el entrenamiento y selección de las respuestas.

## Metodología

De acuerdo al problema planteado en el presente trabajo, se propuso la siguiente metodología como solución: como primer paso, explorar diferentes técnicas de desarrollo de sistemas de diálogo para identificar la que mejor se ajustara a la necesidad ya mencionada. En este proceso se identificaron diferentes enfoques, los cuales se describen a continuación:

### Sistema de diálogo basado en reglas

Según Jurafsky y Martin (2017) en el capítulo 28, son sistemas caracterizados por atender una tarea en particular teniendo conversaciones cortas. En este sentido, guiados por un tutorial proporcionado por Nagesh Singh (2019)<sup>2</sup> donde se explica el paso a paso para la construcción de chatbots con estas características, se creó el back-end de un sistema de diálogo (chatbot) basado en reglas. En este proceso, se utilizó la librería *Nltk* de Python, caracterizada como líder en el Procesamiento del Lenguaje Natural (Bird, Klein y Loper, 2009), la cual tiene un módulo llamado *nltk.chat*, que simplifica la creación de chatbots proporcionando un marco genérico. De esta manera, es posible crear chatbots sin usar algoritmos de aprendizaje automático (Nagesh Singh, 2019). El sistema consistió en tomar los datos proporcionados por el grupo de investigación de Traducción y Nuevas Tecnologías (TNT) de la Universidad de Antioquia, encargado de administrar el examen de traductor e intérprete oficial (preguntas y respuestas) e identificar las palabras claves de cada pregunta asignando algunas reglas y asociando su posible respuesta en cada etiqueta. De esta manera, si la pregunta del usuario cumplía con alguna de las reglas creadas, se mostraba la respuesta asociada, de lo contrario su retorno era *None*.

---

<sup>2</sup> Build Your First Chatbot Using Python & NLTK:  
<https://www.kdnuggets.com/2019/05/build-chatbot-python-nltk.html>

## Sistema de diálogo basado en corpus

Jurafsky y Martin (2017) en la sección 28.1.2, exponen dos tipos de sistemas basados en corpus: uno basado en recuperación de información y el otro basado en modelos secuencia a secuencia. En el presente trabajo, se implementó el de recuperación de información, para el cual se diseñaron dos conjuntos fijos de datos (datos proporcionados por el grupo de investigación TNT), uno contiene las preguntas relacionadas con el examen de traductor e intérprete oficial y el otro conjunto de datos contiene las respuestas a dichas preguntas. De esta manera, se usó un conjunto de respuestas predefinidas que son accedidas por diferentes modelos y que permiten hallar la respuesta más adecuada basada en la entrada del usuario.

Jurafsky y Martin (2017) en la sección antes mencionada afirman que se puede usar cualquier algoritmo de recuperación para elegir la respuesta adecuada, y proponen dos métodos para ello. El primero consiste en que dada la consulta del usuario se encuentra la más similar en un conjunto de datos (en el archivo de preguntas) y retorna la respuesta asociada a esta (desde el archivo de respuestas). El segundo método es que al ingresar la consulta del usuario, se encuentra la que más se asemeja (en el archivo respuestas) y la retorna como respuesta.

En el presente trabajo se aplicaron ambas estrategias. Para aplicar estos métodos de recuperación de información, se hizo necesario representar los datos. Dicha representación consistió básicamente en dos grandes fases:

- a. Limpiar los datos no estructurados a partir de expresiones regulares, eliminando toda la información irrelevante para el sistema de diálogo, dejándolos listos para ser usados como texto en lenguaje natural. Este proceso, consistió inicialmente en pasar todas las palabras a minúsculas, las correspondientes a las preguntas del examen de traductor e intérprete oficial, esto se realizó con la función *lower()* de Python, que devuelve una copia de

la cadena que se pasa al método con todas las letras en minúscula (Van Rossum y Drake, 1995). Luego, era necesario seleccionar sólo las palabras, es decir, sacar los caracteres especiales además de la puntuación, para esto, se implementó el módulo *re*, que es un módulo de Python que contiene las funciones y clases relativas a búsqueda de patrones mediante expresiones regulares con las funciones *compile()* y *escape()* (Van Rossum y Drake, 1995).

La aplicación de estas funciones fué de esta manera:

```
re.compile('[¿%s]' % re.escape(string.punctuation))
```

Con esta línea de código, se hizo el filtrado de los caracteres especiales que se muestran en la expresión regular `[¿%s]`, más los que retorna la función *string.punctuation* que contiene también los signos de puntuación (Van Rossum y Drake, 1995).

b. Llevar los datos desde texto en lenguaje natural a un modelo de espacio vectorial.

Los datos fueron representados usando las siguientes estrategias:

**Matriz de términos en documentos o Bag of Words:** Como lo explican Jurafsky y Martin (2017) en la sección 15.1.1, una colección de textos puede ser representado en una matriz, en la que cada columna representa una palabra en el vocabulario (conjunto de palabras presentes en todos los textos de la colección) y cada fila representa un documento de la colección de textos o viceversa. Para este caso cada fila representa una pregunta del dominio del examen de traductor e intérprete oficial y cada columna una palabra del vocabulario.

Básicamente, la representación de cada pregunta, se realiza en función de las palabras que esta contiene, construyendo por cada pregunta un vector que contabiliza el número de veces que una palabra está presente en dicha pregunta, produciendo de esta manera una matriz sparse<sup>3</sup>, en la que la mayoría de las entradas son ceros ya que cada pregunta sólo contiene un pequeño subconjunto de palabras del vocabulario.

---

<sup>3</sup> Matriz Sparse es una matriz en la que la mayor parte de sus elementos son cero (Brownlee, 2018).



**Term Frequency - Document Inverse Frequency (TF-IDF):** Este término, está dividido en dos factores: Se define TF como la frecuencia del término (Luhn, 1957), es decir, el número de veces que la palabra se encuentra en el documento. El factor IDF, se define como la frecuencia inversa del documento (Sparck Jones, 1972) se utiliza para dar mayor peso a las palabras que aparecen solo en unos pocos documentos. Los términos que se limitan a unos pocos documentos son útiles para discriminar esos documentos del resto de la colección; los términos que aparecen con frecuencia en toda la colección no son tan útiles. En la sección 15.2.1 del libro de Jurafsky y Martin (2017), IDF es definido usando la fracción  $N/d f_i$ , donde  $N$  es el total de documentos en la colección y  $d f_i$  es número de documentos en los cuales ocurre el término  $i$ . A los términos que aparecen en todos los documentos se les asigna un peso menor a 1. Debido a la gran cantidad de documentos que se pueden tener, esta medida fue definida con una función *log*:

$$\text{idf}_i = \log\left(\frac{N}{d f_i}\right) \quad (1)$$

Según Jurafsky y Martin (2017) en la sección 15.2.1: “*La combinación de la frecuencia del término (TF) con IDF, da como resultado un esquema conocido como la ponderación tf-idf del valor de la palabra  $i$  en el documento  $j$ ,  $w_{ij}$* ” (p.278):

$$w_{ij} = \text{tf}_{ij} \text{idf}_i \quad (2)$$

En resumen, tf-idf prefiere palabras que son recurrentes en el documento actual  $j$  pero que no son frecuentes en la colección en general.

**Word embeddings:** Son representaciones vectoriales de palabras donde las que son semánticamente similares se encuentran cerca unas de otras en un espacio vectorial. Firth (1957) en la década de 1950, definió el término colocación como la coincidencia común de

palabras particulares, este famoso lingüista Británico, afirmó que “*You shall know a word by the company it keeps*”, indicando con esto que el significado de una palabra deriva de las palabras con las que se coexiste.

Según García (2018), “*Los word embedding son un enfoque de la semántica de distribución que representa palabras como vectores de número reales. Dicha representación tiene propiedades de agrupamiento útiles, ya que agrupa palabras que son semánticamente y sintácticamente similares. Por ejemplo esperamos que las palabras “delfín y foca” se encuentren cerca, pero “París” y “delfín” no se encuentren cerca ya que no existe una fuerte relación entre ellas*” (p.9).

Existen métodos de generación de embeddings y también embeddings pre entrenados. Para el presente trabajo, por temas de costo computacional y por su amplia difusión y éxito en muchos problemas de procesamiento de lenguaje natural, se usaron los siguientes dos modelos de embeddings pre entrenados:

***Fasttext***: Contiene distribuciones de vectores de palabras para 157 idiomas, cada palabra es tratada como la suma de sus composiciones de caracteres llamados *ngrams*. El entrenamiento de los vectores de palabras usa conjuntos de datos compuestos de una mezcla de Wikipedia y Common Crawl. Cuenta con representaciones de 300 dimensiones por palabra (Grave, Bojanowski, Gupta, Joulin y Mikolov, 2018).

***Wikipedia2Vec***: Contiene distribuciones de vectores de palabras para 12 idiomas en formato binario y de texto. Tiene representaciones de 300 dimensiones por palabra y ha sido entrenado desde páginas de wikipedia (Yamada, Asai, Shindo, Takeda, y Takefuji, 2018).

Una vez identificados los tipos de representaciones a utilizar para el conjunto de datos, se pasó a implementar los diferentes clasificadores para hallar la respuesta ante las posibles preguntas de los usuarios (interesados en el examen de traductor e intérprete oficial). Para esto se utilizaron los siguientes algoritmos de recuperación de información:

**Distancia coseno:** La distancia coseno entre  $x$  y  $y$  es definida como:

$$D(x, y) = 1 - \left( \frac{x \cdot y}{\|x\| \|y\|} \right) \quad (3)$$

Este modelo mide la similitud entre dos palabras hallando el producto punto entre la representación vectorial de ambas palabras, dividido por el tamaño de cada uno de los vectores, donde entre más alto sea el resultado de dicha operación, es mayor la similitud entre las palabras. Para este caso se implementó la librería de *Scipy* de Python (Virtanen et al., 2019) con el módulo *scipy.spatial*, donde la función *distance.cosine* halla la distancia coseno restando la similitud mencionada anteriormente al número 1, indicando así la mayor similitud en la distancia cero y la menor en la distancia 1.

**Distancia Jaccard:** Definida por (Jaccard 1908, Jaccard 1912) como:

$$\text{Jaccard}(\vec{x}, \vec{y}) = \frac{\sum_{i=1}^N \min(x_i, y_i)}{\sum_{i=1}^N \max(x_i, y_i)} \quad (4)$$

Se explica la similitud de Jaccard, como otra medida para hallar la similitud entre palabras, donde se divide el valor mínimo sobre el máximo de un par de vectores que en este caso son representaciones de preguntas y respuestas. Esta división halla la similitud entre ambos vectores y para hallar su distancia se resta este valor al número 1, indicando así en la distancia cero la mayor similitud y en la 1 la mayor diferencia.

**Machine learning:** Se implementaron 4 clasificadores:

***Support Vector Machines (SVM):*** Las Máquinas de Vectores Soporte fueron desarrolladas por Vapnik (1999) y sus colaboradores en el marco de la Teoría de Aprendizaje Estadístico (SLT). Constituyen un método basado en aprendizaje para la resolución de problemas de clasificación y regresión.

Esta es una nueva técnica de clasificación que ha tenido mucho uso en los últimos años. *“Tiene la capacidad de formar una frontera de decisión alrededor del dominio de los datos de aprendizaje con muy poco o ningún conocimiento de los datos fuera de esta frontera. Los datos son mapeados por medio de un kernel Gaussiano u otro tipo de kernel a un espacio de características en un espacio dimensional más alto, donde se busca la máxima separación entre clases”* (Betancourt, 2005, p.67).

Para este trabajo se implementó *SVC* con el módulo *svm* de la librería de *sklearn* de python (Pedregosa et al., 2011).

***Random Forest:*** El algoritmo para inducir un Random Forest fué desarrollado por Breiman (2001) y Cutler (2013) como una combinación de árboles predictores, de tal manera que cada uno depende de los valores de un vector aleatorio probado independientemente y con la misma distribución para todos los árboles.

Para las pruebas realizadas en el presente trabajo, se implementó *RandomForestClassifier* del módulo *ensemble* de la librería *sklearn* de python. *“Es un estimador que se ajusta a varios clasificadores de árboles de decisión en varias submuestras del conjunto de datos y utiliza el promedio para mejorar la precisión predictiva y controlar el sobre ajuste”* (Pedregosa et al., 2011).

***K-Nearest-Neighbor (KNN):*** Fue introducido por Fix y Hodges (1951) como un método no paramétrico para la clasificación de patrones, el cual se conoce como k-vecinos

donde su esencia se centra en buscar usuarios con tendencias y características semejantes, que permitan obtener información relevante.

Para este trabajo, se utilizó *KNeighborsClassifier* del módulo *neighbors* de la librería de *sklearn* de python. Este modelo clasifica valores buscando los puntos de datos más similares por cercanía, aprendidos en la etapa de entrenamiento (Pedregosa et al., 2011).

**Regresión Logística:** Surge en la década de los 60 con la aparición de un trabajo sobre el riesgo de padecer una enfermedad coronaria. Su generalización dependía de la solución que se diera al problema de la estimación de los coeficientes. El algoritmo de Walker y Duncan (1967) para la obtención de los estimadores de máxima verosimilitud vino a solucionar en parte este problema, pero era de naturaleza tal que el uso de computadoras resultaba imprescindible (Cramer, 2002).

Para el desarrollo del trabajo se implementó *LogisticRegression* del módulo *linear\_model* de *sklearn*. Este modelo es conocido en la literatura como clasificador de máxima entropía o modelo logit, donde las probabilidades que describen los posibles resultados de una sola prueba son modelados usando una función logística (Pedregosa et al., 2011).

**Redes Neuronales de Propagación hacia adelante (Feed-Forward Neural Networks):** Entre las décadas de 1950 y 1960, el científico Rosenblatt (1958) diseñó el Perceptrón, un tipo de clasificador lineal, la unidad desde donde nacería y se potenciarían las redes neuronales artificiales.

Para este caso se utilizó el modelo *secuencial* con la librería de *Keras* (Chollet, 2015) para un problema de clasificación multi clase donde se trabajó con diferentes configuraciones de la red neuronal.

**Implementaciones:** Para iniciar con las pruebas de los diferentes modelos, se partió de la base de datos original, es decir, sólo con las preguntas y respuestas proporcionadas por el grupo de investigación TNT, donde se tenían 33 preguntas y respuestas:

***Bag of Words (MTD) con distancia Coseno:*** Se realizó la representación de los datos con *Bag of Words*, y luego para seleccionar la respuesta correcta se utilizó la distancia coseno. En primer lugar se hizo la verificación hallando la similitud de la entrada del usuario con el archivo de preguntas donde de acuerdo a la representación vectorial de la entrada, se buscaba con cuál de las preguntas tenía mayor similitud para retornar la respuesta asociada a esta. De igual manera se realizó la prueba hallando la similitud de la entrada del usuario con el archivo de respuestas, donde se retornaba esa respuesta en caso de tener la mayor similitud allí.

***Bag of Words (MTD) con distancia Jaccard:*** En la búsqueda de mejores resultados, representando los datos con *Bag of Words* se implementó la similitud de Jaccard donde se realizó la misma validación que en la implementación anterior, la diferencia es que aquí se utilizó otro tipo de medida de la similitud.

***Term Frequency - Document Inverse Frequency (TF-IDF) con distancia Coseno:*** La representación TF-IDF como se explicó anteriormente, tiene un valor agregado a la matriz de términos en documentos o *Bag of Words*, ya que se hace la representación de cada pregunta de forma vectorial pero teniendo en cuenta la frecuencia de los términos en cada pregunta y en el vocabulario en general. Una vez realizada esta representación se aplicó la distancia coseno y se realizaron las validaciones tanto con el archivo de preguntas como con el de respuestas.

Para el análisis de la eficiencia de estos modelos (MTD\_Coseno, MTD\_Jaccard, TF-IDF\_Coseno), se realizaron pruebas con 4 preguntas (se detallarán en breve), teniendo en

cuenta el parámetro *ngram\_range* de la función *CountVectorizer* de *Sklearn* para MTD y *TfidfVectorizer* para TF-IDF, la cual permite la creación de la matriz tomando rangos desde (1,1) hasta (4,4) en este caso. Donde (1,1) toma cada palabra, (2,2) pares de palabras consecutivas, (1,2) tiene en cuenta las dos opciones anteriores, y así sucesivamente (Pedregosa et al., 2011).

Las preguntas propuestas fueron las siguientes:

***Quiero inscribirme al examen, ¿qué hago?***

***Si tengo inconvenientes para realizar la prueba después de estar inscrito, ¿me devuelven el dinero?***

***¿Puedo presentar el examen en español inglés, español francés?***

***¿Puedo tomar nota durante la prueba?***

En la búsqueda de mejorar los resultados, se aumentó la base de datos aplicando la técnica de parafraseo, tomando cada pregunta y escribiéndola de diferentes maneras, llegando a 263 preguntas para 46 respuestas.

***Bag of Words (MTD) y Term Frequency - Document Inverse Frequency (TF-IDF) con modelos Machine Learning:*** Se realizó el planteamiento como un problema de aprendizaje supervisado y se resolvió con diferentes modelos de machine learning. Para esta implementación se hizo necesario la configuración de la base de datos, asignando valores numéricos a las clases que son en este caso las respuestas a las preguntas planteadas. De esta manera, se identificaron con la base de datos modificada 46 clases (respuestas) para 263 preguntas con 367 características. Se realizó el entrenamiento con los 4 modelos de machine learning mencionados anteriormente con las representaciones MTD y TF-IDF, inicialmente con el parámetro *ngram\_range* (1,1). Debido a los pocos datos que se tenían, se utilizó la metodología de validación cruzada *KFold* de *sklearn* con *leave-one-out*, donde *K* es igual al

número de muestras, lo que significa una iteración a través de cada muestra (Pedregosa et al., 2011).

Para realizar el testing de estos modelos, con la intención de mejorar los resultados ya que en la validación de los modelos anteriores se hizo una evaluación superficial donde los resultados fueron muy deficientes, para este caso se seleccionaron 20 preguntas relacionadas con el examen de traductor e intérprete oficial. Esto, teniendo en cuenta que se trata de modelos machine learning donde se debe evitar crear falsas expectativas sobre los resultados y tener una mejor validación. Estas fueron las preguntas definidas:

*¿Cómo puedo pagar el examen?*

*No puedo presentar el examen, ¿pueden devolverme el dinero?*

*Necesito información del examen, ¿a dónde me comunico?*

*Quiero saber si puedo pagar el examen financiado*

*Perdí mi documento de identidad, ¿qué otro documento puedo presentar?*

*¿Tomar nota?*

*Diccionarios*

*Pagar*

*Charla informativa*

*¿Cuánto cuesta?*

*Me gustaría saber cuándo hay inscripciones al examen para traductor e intérprete oficial*

*Estaba seguro de que pasaría el examen, ¿podemos revisarlo?*

*¿Qué documento me entregan una vez aprobado el examen?*

*Quiero presentar el examen en las vacaciones de mitad de año, ¿es posible?*

*Mi cédula se me perdió, tengo el denuncia, ¿hay algún problema?*

*Omisión de párrafos*



***Tengo dudas con mi calificación, ¿puedo solicitar revisión o retroalimentación del examen?***

***¿Cómo se hace el pago del examen?***

***Durante la prueba, ¿es posible acceder a algún recurso?***

***Necesito cancelar el examen por motivos personales, ¿puedo hacerlo?***

***Embeddings con Coseno y machine learning:*** Para esta implementación se representaron inicialmente en embeddings las palabras del archivo preguntas de la base de datos modificada usando los embeddings pre entrenados explicados anteriormente, donde se generó un archivo en formato txt con dicha representación tanto para *Fasttext* como para *Wikipedia2Vec*. Luego se realizó la representación vectorial de cada pregunta realizando una estandarización de acuerdo al mayor vector encontrado, quedando cada pregunta representada por un vector de 8700 posiciones. Esto dado que la pregunta más larga de la base de datos tenía 29 palabras y los embeddings otorgan 300 representaciones por palabra. De esta manera, las entradas del usuario son representadas también en embeddings y estandarizadas para luego pasarlas al modelo ya sea coseno o machine learning.

#### ***Term Frequency - Document Inverse Frequency (TF-IDF) con Feed-Forward***

***Neural Networks:*** En vista de los buenos resultados de la representación TF-IDF, se planteó la idea de verificar cómo sería su comportamiento con un modelo de red neuronal artificial. De esta manera, con la misma base de datos modificada, se realizó la representación con TF-IDF y luego se realizó el entrenamiento del modelo con diferentes configuraciones de la red neuronal para validar su comportamiento.

Con todas las validaciones realizadas con los modelos presentados, se identificaron patrones de falla como por ejemplo, el no responder adecuadamente ante preguntas muy cortas o simplemente palabras claves. Para tratar de solucionar esto, se parafraseó la base de

datos nuevamente, incluyendo el mayor número posible de formas en que podría referirse el usuario, además de agregar preguntas importantes que no estaban incluidas en la base de datos inicial y que el usuario podría realizar. De esta manera, la base de datos quedó con un total de 594 preguntas para 45 clases o respuestas con 537 características o palabras en el vocabulario. Con esta nueva base de datos, se realizaron nuevamente las pruebas con los modelos de machine learning y sus diferentes representaciones de datos.

Después de esta validación, se hicieron diferentes interacciones con el chatbot para verificar las posibles fallas y así mirar si podrían ser tratadas mediante reglas. Una de las fallas identificadas tenía que ver con la salida del chat, se debía definir con qué entradas el chatbot debía cerrar la conversación. Para esto, se creó un archivo en formato txt con diferentes palabras de despedida, el cual es leído y comparado con cada entrada del usuario y en caso de ser igual, se cierra la conversación, en caso de no coincidir se pasa al modelo para encontrar la respuesta a la inquietud. Otra de las fallas que se debía tratar era filtrar las intervenciones del usuario para identificar si está hablando de asuntos ajenos al examen de traductor e intérprete oficial. Para este caso, se diseñó una *stop list* para eliminar las palabras que no tienen contenido útil; esto se hizo implementando una representación TF-IDF, sólo para hacer esta validación, de esta manera, en caso de ingresar por ejemplo la palabra 'casa' o 'maleta' el chatbot responderá que no puede responder a este tipo de inquietudes. Esta validación se hace verificando el vector que retorna la representación TF-IDF, con el método *numpy.count\_nonzero()* de la librería *Numpy* (Oliphant, 2006) que retorna la cantidad de posiciones diferentes de cero, donde si el resultado es cero quiere decir que la entrada no tiene que ver con el examen de traductor e intérprete oficial. En caso de ser diferente de cero el valor retornado, es decir, si la entrada tiene que ver con el examen de traductor e intérprete oficial, hay que verificar que la pregunta del usuario tenga algo de sentido, es decir, validar

que al menos se ingresen dos palabras, aunque en algunos casos sí se acepta una sola palabra, como: si la entrada es 'si', se valida que esté afirmando tener otra inquietud, si la entrada es 'no', se valida que no tengan más dudas para cerrar la conversación o si la entrada corresponde a afirmaciones de una sola palabra que se tienen definidas en la base de datos como 'genial', 'ok', se pasan al modelo para obtener la respuesta. De esta manera, se tienen algunos casos donde se aceptan entradas de una sola palabra, para los demás casos se tiene el mensaje 'No comprendo su pregunta, por favor sea más claro'. En la búsqueda de estrategias para identificar las consultas relacionadas con el examen de traductor e intérprete oficial, se utilizó la similitud con coseno para determinar los casos donde podría pasarse dicha consulta al modelo de acuerdo a un umbral definido, pero los resultados no fueron buenos, pues hay muchas limitaciones ya que es difícil hallar un umbral que funcione bien para todas las diferentes entradas que puede hacer el usuario.

Después de la etapa de evaluación de los modelos, se seleccionó el mejor de ellos y se llevó al desarrollo del sistema de diálogo como solución web. Se realizó la creación del *back-end* usando las tecnologías que mejor se adaptaron a la necesidad del proyecto. Luego, para el desarrollo del *front-end*, se trabajó con Django que es un framework de desarrollo web de código abierto, escrito en Python, que respeta el patrón de diseño conocido como modelo, vista, template (Django, 2015). En las consultas realizadas se encontraron diferentes tutoriales de desarrollo de chatbots con este framework, tal es el caso de Planeta Chatbot, donde hay un tutorial de cómo implementar un chatbot en Django (Muiruri, 2018). En este tutorial se utiliza ChatterBot como *back-end*, que es una librería de Python que utiliza algoritmos de aprendizaje automático para la generación de diferentes tipos de respuestas (ChatterBot, 2019). Después de analizar este tutorial, se encontró que se podía adaptar esta interfaz gráfica

al presente proyecto ya que realizando modificaciones sencillas y viables se podía cubrir la necesidad, pues dichas modificaciones eran básicamente agregar el *back-end* desarrollado.

Finalmente, se realizó el despliegue de la solución en un servidor de aplicaciones, como la integración de todos los componentes del sistema, validando la correcta operatividad del mismo. Para este caso se implementó Heroku, que es una plataforma como servicio de computación en la nube que soporta distintos lenguajes de programación, además es gratuita y es la más utilizada en el mercado (Lindenbaum, Wiggins y Henry, 2007).

En la búsqueda de información con explicación detallada para llevar a cabo este objetivo se encontraron diferentes tutoriales con los cuales se logró comprender el procedimiento para esta tarea y básicamente teniendo como guía el tutorial de *djangogirls* (2014) donde se explica el paso a paso para realizar despliegue de una aplicación Django en Heroku se pudo dar cumplimiento al último objetivo específico del presente trabajo.

## **Resultados y análisis**

De acuerdo a la metodología planteada, se propuso trabajar básicamente bajo dos modalidades; sistema de diálogo basado en reglas y sistema de diálogo basado en corpus. Para el primer caso, se desarrolló un sistema de diálogo muy limitado donde la mayoría de respuestas correctas se daban cuando se ingresaba la pregunta de la forma en que fue programada, obteniendo de esta manera muchos errores. En el proceso de validación del modelo, se simuló una conversación donde como se muestra en la figura 1, para los casos donde no fué programada una respuesta, el retorno es *None* (desconocido):

>¿puedo tomar nota?  
 Sí.  
 >ok  
 Muy bien, ¿Alguna otra inquietud?  
 >¿cuánto cuesta el examen?  
 \$1'060.000 (tarifa plena) \$954.000 (estudiantes y egresados del programa de traducción de la Universidad de Antioquia)  
 >perfecto  
 Muy bien, ¿Alguna otra inquietud?  
 >¿dónde se realiza el examen?  
 En la Universidad de Antioquia ubicada en Medellín.  
 >bien  
 None  
 >¿qué documento debo presentar el día del examen?  
 None  
 >omisión de párrafos  
 None  
 >¿qué es omisión de párrafos?  
 Se trata simplemente de eso mismo, cuando se omite un párrafo completo, es decir, aunque no lo creas es un error muy frecuente en los estudiantes de Traducción que olvidan traducir un párrafo, puede ser por la longitud de algunos textos o por simple descuido.  
 >¿qué recursos puedo usar durante el examen?  
 Sí, tenemos un listado de las fuentes de consulta que se sugieren sean utilizadas (Archivo “fuentes de consulta.”)  
 >perfecto  
 Muy bien, ¿Alguna otra inquietud?  
 >¿Cómo me inscribo al examen?  
 None  
 >¿cómo hago la inscripción?  
 Debe ingresar al portal y verificar el cronograma.  
 >¿cómo realizo la inscripción al examen?  
 La inscripción la puede realizar en el siguiente enlace: <https://goo.gl/eFJ4mS> . En 'Palabra clave' escribe la palabra 'traductor'.

*Figura 1:* Interacción con chatbot basado en reglas

Fuente: elaboración propia

Como puede verse en la interacción con el chatbot de la figura 1, es claro que en los casos donde el usuario ingresa la misma consulta con alguna palabra diferente, no hay respuesta como en el caso “¿qué es omisión de párrafos?” y “omisión de párrafos”, para las que debía obtenerse la misma respuesta por parte del chatbot. Realmente es un método complejo y sería muy inviable en caso de quererlo mejorar, pues se tendría que definir una

cantidad de reglas para cada entrada o pregunta, teniendo en cuenta las diferentes formas en que el usuario puede referirse.

La modalidad de sistemas de diálogo basados en corpus utilizan repositorios de datos predefinidos que en este caso son los conjuntos de preguntas y respuestas, donde con alguna estrategia o algoritmo de recuperación de información es posible seleccionar la respuesta adecuada basada en la entrada y el contexto. En este sentido, se realizó la implementación con diferentes modelos, de acuerdo a diferentes representaciones de los datos y algoritmos de recuperación de información.

La primera validación fue realizada con las representaciones de *Bag of Words* (MTD) y *Term Frequency - Document Inverse Frequency* (TF-IDF) con la distancia coseno y Jaccard. Para estos 3 modelos se realizaron pruebas con las 4 preguntas predefinidas indicadas en la metodología y los diferentes *ngram\_range*, tanto para el archivo preguntas como para el de respuestas. Aunque la literatura dice que se encuentra un mayor acierto al realizar la comparación directa con las respuestas, como lo afirman Jurafsky y Martin (2017) en la sección 28.1.2; en este caso solo se obtuvieron mejores resultados en las validaciones de acuerdo al archivo de preguntas, usando diferentes n-gramas (*ngram\_range*: (1,1), (1,2), (1,3) y (1,4)) como se muestra en la tabla 1:

Tabla 1: Resultados del testing con 4 preguntas para los modelos MTD y TF-IDF con la distancia coseno y Jaccard de acuerdo a diferentes n-gramas

MODELO	ACIERTOS (ngram_range (1,1))	ACIERTOS (ngram_range (1,2))	ACIERTOS (ngram_range(1,3))	ACIERTOS (ngram_range(1,4))
MTD_Coseno	1	4	4	4
MTD_Jaccard	2	4	4	3
TF-IDF_Coseno	3	4	4	4

Fuente: elaboración propia

Analizando los resultados de la tabla 1, se puede deducir que el modelo que tiene más aciertos es TF-IDF\_Coseno, es decir, muestra una mayor eficiencia; además es claro que la mayoría de los aciertos se dan con los *ngram\_range* (1,2) y (1,3). De aquí se podría afirmar que la representación TF-IDF funciona mejor que MTD, pues muestra 2 aciertos más de diferencia.

Con las mismas representaciones de datos se hizo la validación con los modelos machine learning para: *Support Vector Machines* (SVM), *K-Nearest-Neighbor* (KNN), Regresión Logística (R.Logística) y *Random Forest* (Random\_Forest) donde se encontraron los siguientes resultados:

Tabla 2: Resultados de las pruebas de acuerdo al error de validación con las representaciones MTD y TF-IDF con los modelos de machine learning

<b>MODELO</b>	<b>ERROR DE VALIDACIÓN</b>	<b>MODELO</b>	<b>ERROR DE VALIDACIÓN</b>
MTD_SVM	0.34	TF-IDF_SVM	0.24
MTD_KNN	0.44	TF-IDF_KNN	0.323
MTD_R.Logística	0.24	TF-IDF_R.Logística	0.22
MTD_Random_Forest	0.31	TF-IDF_Random_Forest	0.34

Fuente: elaboración propia

La tabla 2 muestra un error de validación para cada modelo, donde el menor se encuentra en el modelo TF-IDF\_R.Logística con 0.22, lo que indica que la representación TF-IDF sigue siendo la más eficiente como se vió también en los modelos anteriores.

En las pruebas realizadas con las preguntas definidas para el testing (20 preguntas) se confirma como lo muestra la tabla 2 que el modelo TF-IDF\_R.Logística proporciona mejores resultados. Los aciertos están determinados por 4 sets que son conjuntos de a 5 preguntas.

Este es el resumen de lo que se obtuvo:



Tabla 3: Resultados del testing con 20 preguntas para la representación MTD y los modelos de machine learning

MODELO	ACIERTOS (SET 1)	ACIERTOS (SET 2)	ACIERTOS (SET 3)	ACIERTOS (SET 4)	TOTAL ACIERTOS
MTD_KNN	3	3	1	5	12
MTD_R.Logística	5	5	3	4	17
MTD_SVM	3	0	2	4	9
MTD_Random_Forest	4	1	4	4	13

Fuente: elaboración propia

Tabla 4: Resultados del testing con 20 preguntas para la representación TF-IDF y los modelos de machine learning

MODELO	ACIERTOS (SET 1)	ACIERTOS (SET 2)	ACIERTOS (SET 3)	ACIERTOS (SET 4)	TOTAL ACIERTOS
TF-IDF_R.Logística	5	5	4	4	18
TF-IDF_KNN	4	4	2	4	14
TF-IDF_SVM	4	5	4	3	16
TF-IDF_Random_Forest	4	1	3	4	12

Fuente: elaboración propia

Analizando las tablas 3 y 4, se observó que los mejores resultados se obtienen con la representación TF-IDF, donde se muestra un margen de error muy mínimo, pues de 20 preguntas, se obtienen sólo dos incorrectas, lo que indica una eficiencia del 90%.

Con la representación usando *word embeddings*, se realizaron validaciones tanto con la distancia coseno como con los modelos de machine learning.

Para verificar la eficiencia de los modelos *word embeddings* de *Wikipedia2Vec* y *word embeddings* de *Fasttext* con distancia coseno, se realizó el testing utilizando las mismas 20 preguntas mencionadas anteriormente y se obtuvieron estos resultados:

Tabla 5: Resultados del testing con 20 preguntas para los modelos de *word embeddings* con distancia coseno

MODELO	ACIERTOS (SET 1)	ACIERTOS (SET 2)	ACIERTOS (SET 3)	ACIERTOS (SET 4)	TOTAL ACIERTOS
Wikipedia2Vec_Coseno	3	1	1	1	6
Fasttext_Coseno	2	1	2	2	7

Fuente: elaboración propia

En la tabla 5 se aprecia que la representación de *word embeddings* con *Fasttext* muestra un acierto más respecto a los resultados obtenidos usando *Wikipedia2Vec*, aunque sólo se tienen 7 aciertos de un total de 20 preguntas, lo que indica más respuestas erradas que acertadas.

En la validación con la representación usando *word embeddings* y los modelos de *machine learning* se obtuvieron los siguientes resultados en cuanto al error de validación:

Tabla 6: Resultados de las pruebas de acuerdo al error de validación con las representaciones en *word embeddings* y los modelos de *machine learning*

MODELO	ERROR DE VALIDACIÓN	MODELO	ERROR DE VALIDACIÓN
Fasttext_SVM	0.69	Wikipedia2Vec_SVM	0.71
Fasttext_KNN	0.71	Wikipedia2Vec_KNN	0.72
Fasttext_R.Logística	0.67	Wikipedia2Vec_R.Logística	0.67
Fasttext_Random_Forest	0.70	Wikipedia2Vec_Random_Forest	0.70

Fuente: elaboración propia

De acuerdo a los resultados de la tabla 6 y con respecto a los demás modelos analizados, se encontró que estas representaciones en *word embeddings* proporcionan un error de validación muy alto, el menor es 0,67, lo que indica una eficiencia de un 33%, que es bastante baja para los propósitos del presente trabajo.

En el testing con las 20 preguntas definidas estos fueron los resultados:

Tabla 7: Resultados del testing con 20 preguntas para la representación en *word embeddings* con *Fasttext* y los modelos de *machine learning*

MODELO	ACIERTOS (SET 1)	ACIERTOS (SET 2)	ACIERTOS (SET 3)	ACIERTOS (SET 4)	TOTAL ACIERTOS
Fasttext_KNN	1	0	0	0	1
Fasttext_R.Logística	2	2	2	3	9
Fasttext_SVM	2	1	2	4	9
Fasttext_Random_Forest	2	0	0	4	6

Fuente: elaboración propia

Tabla 8: Resultados del testing con 20 preguntas para la representación en *word embeddings* con *Wikipedia2Vec* y los modelos de *machine learning*

MODELO	ACIERTOS (SET 1)	ACIERTOS (SET 2)	ACIERTOS (SET 3)	ACIERTOS (SET 4)	TOTAL ACIERTOS
Wikipedia2Vec_KNN	1	0	0	0	1
Wikipedia2Vec_R.Logística	2	2	0	3	7
Wikipedia2Vec_SVM	2	1	0	2	5
Wikipedia2Vec_Random_Forest	2	0	1	4	7

Fuente: elaboración propia

En el análisis realizado a los resultados obtenidos en las tablas 7 y 8, se obtiene que el total de aciertos es bastante bajo para estos modelos, pues el que proporciona mejores resultados solo tiene 9 aciertos de 20 preguntas.

En cuanto a los resultados obtenidos con las representaciones en *word embeddings*, MTD y TF-IDF, se encontró que los modelos TF-IDF con regresión logística y SVM proporcionaban los mejores resultados. Estos resultados fueron obtenidos con el parámetro *ngram\_range* (1,1), por tanto, en la búsqueda de una mejor eficiencia, se realizó la validación para estos dos modelos con otros n-gramas (*ngram\_range*: (1,2), (1,3), (2,2), (2,3), (3,3)); como se muestra en la tabla 9:

Tabla 9: Resultados de acuerdo a diferentes *n*-gramas para la representación TF-IDF y los modelos de R.Logística y SVM

MODELO	ERROR DE VALIDACIÓN				
	<i>ngram_range</i>				
	(1,2)	(1,3)	(2,2)	(2,3)	(3,3)
TF-IDF_R.Logística	0,25	0,27	0,44	0,54	0,67
TF-IDF_SVM	0,33	0,38	0,52	0,57	0,66

Fuente: elaboración propia

Haciendo la comparación de los resultados obtenidos de los modelos en la tabla 9 con el error de validación encontrado en la tabla 2, donde se utilizó el parámetro *ngram\_range* (1,1), es claro que no hay mejoría en la predicción, por el contrario, el error aumenta.

En vista de los buenos resultados de la representación TF-IDF con *ngram\_range* (1,1), se verificó con el modelo *Feed-Forward Neural Networks* si se encontraba mejor eficiencia y se obtuvo lo siguiente:

Tabla 10: *Resultados de acuerdo al error de validación para la representación TF-IDF y el modelo Feed-Forward Neural Networks*

<b>CONFIGURACIÓN DEL MODELO</b>	<b>ERROR DE VALIDACIÓN</b>
Número de capas: 4 Unidades de entrada: 367 Unidades ocultas capa 1: 15 neuronas Unidades ocultas capa 2: 5 neuronas Unidades de salida: 46 Funciones de activación: relu y softmax	0,60
Número de capas: 7 Unidades de entrada: 367 Unidades ocultas capa 1: 15 neuronas Unidades ocultas capa 2: 50 neuronas Unidades ocultas capa 3: 40 neuronas Unidades ocultas capa 4: 20 neuronas Unidades ocultas capa 5: 5 neuronas Unidades de salida: 46 Funciones de activación: relu y softmax	0,70
Número de capas: 4 Unidades de entrada: 367 Unidades ocultas capa 1: 1 neurona Unidades ocultas capa 2: 184 neuronas Unidades de salida: 46 Funciones de activación: relu y softmax	0,95
Número de capas: 4 Unidades de entrada: 367 Unidades ocultas capa 1: 20 neuronas Unidades ocultas capa 2: 10 neuronas Unidades de salida: 46 Funciones de activación: relu y softmax	0,38
Número de capas: 4 Unidades de entrada: 367 Unidades ocultas capa 1: 40 neuronas Unidades ocultas capa 2: 20 neuronas Unidades de salida: 46 Funciones de activación: relu y softmax	0,30
Número de capas: 4 Unidades de entrada: 367 Unidades ocultas capa 1: 100 neuronas Unidades ocultas capa 2: 50 neuronas Unidades de salida: 46 Funciones de activación: relu y softmax	0,24
Número de capas: 3 Unidades de entrada: 367 Unidades ocultas capa 1: 40 neuronas Unidades de salida: 46 Funciones de activación: relu y softmax	0,23

Fuente: elaboración propia

Aunque los resultados evaluados en la tabla 10 son buenos, no alcanzan a ser mejores que el que proporciona la representación TF-IDF con Regresión Logística, en la que el error es de 0.22.

Con el propósito de encontrar mejoría en la predicción de los modelos, se implementó la técnica de reducción de dimensión *Principal Component Analysis* (PCA) con el modelo TF-IDF y la Regresión Logística que hasta ese momento brindaba los mejores resultados y se encontró que el error aumentó, pasando de 0.22 a 0.23, por tanto no es viable esta aplicación.

Con la finalidad de corregir las fallas identificadas, se realizaron nuevamente las pruebas de todos los modelos con la nueva base de datos explicada en la metodología, donde se logró un total de 594 preguntas para 45 clases o respuestas con 537 características o palabras en el vocabulario. Los resultados fueron los siguientes:

#### **MTD y TF-IDF con Modelos machine learning:**

Tabla 11: *Resultados de las pruebas con la base de datos actualizada de acuerdo al error de validación con las representaciones MTD y TF-IDF con los modelos de machine learning*

<b>MODELO</b>	<b>ERROR DE VALIDACIÓN</b>	<b>MODELO</b>	<b>ERROR DE VALIDACIÓN</b>
MTD_SVM	0.28	TF-IDF_SVM	0.17
MTD_KNN	0.37	TF-IDF_KNN	0.24
MTD_R.Logística	0.18	TF-IDF_R.Logística	0.12
MTD_Random_Forest	0.30	TF-IDF_Random_Forest	0.31

Fuente: elaboración propia

## Word Embeddings con Machine Learning:

Tabla 12: Resultados de las pruebas con la base de datos actualizada de acuerdo al error de validación con las representaciones en word embeddings y los modelos de machine learning

MODELO	ERROR DE VALIDACIÓN	MODELO	ERROR DE VALIDACIÓN
Fasttext_SVM	0.32	Wikipedia2Vec_SVM	0.34
Fasttex_KNN	0.48	Wikipedia2Vec_KNN	0.50
Fasttex_R.Logística	0.28	Wikipedia2Vec_R.Logística	0.28
Fasttext_Random_Forest	0.30	Wikipedia2Vec_Random_Forest	0.30

Fuente: elaboración propia

También se realizó la prueba con el modelo *Feed-Forward Neural Networks* con la representación TF-IDF para verificar si había mejoría con la nueva base de datos. En la tabla 13 se presentan los resultados obtenidos:

Tabla 13: Resultados de la prueba con la base de datos actualizada de acuerdo al error de validación para la representación TF-IDF y el modelo *Feed-Forward Neural Networks*

CONFIGURACIÓN DEL MODELO	ERROR DE VALIDACIÓN
Número de capas: 3 Unidades de entrada: 537 Unidades ocultas capa 1: 40 neuronas Unidades de salida: 45 Funciones de activación: relu y softmax	0.18

Fuente: elaboración propia

Después de todas las pruebas realizadas con los diferentes modelos y estrategias de representación de los datos, y analizando los resultados obtenidos en las tablas 11, 12 y 13, se identificó que el modelo que proporciona los mejores resultados era la regresión logística con la representación TF-IDF, con una eficiencia del 88%. Finalmente, para verificar si se podían



mejorar los resultados, se implementó nuevamente la técnica de reducción de dimensión con PCA, sin embargo los resultados empeoraron, pasando el error de validación del 12% al 20%.

A continuación se presenta la matriz de confusión para el modelo de regresión logística con la representación TF-IDF, el cual presentó los mejores resultados:

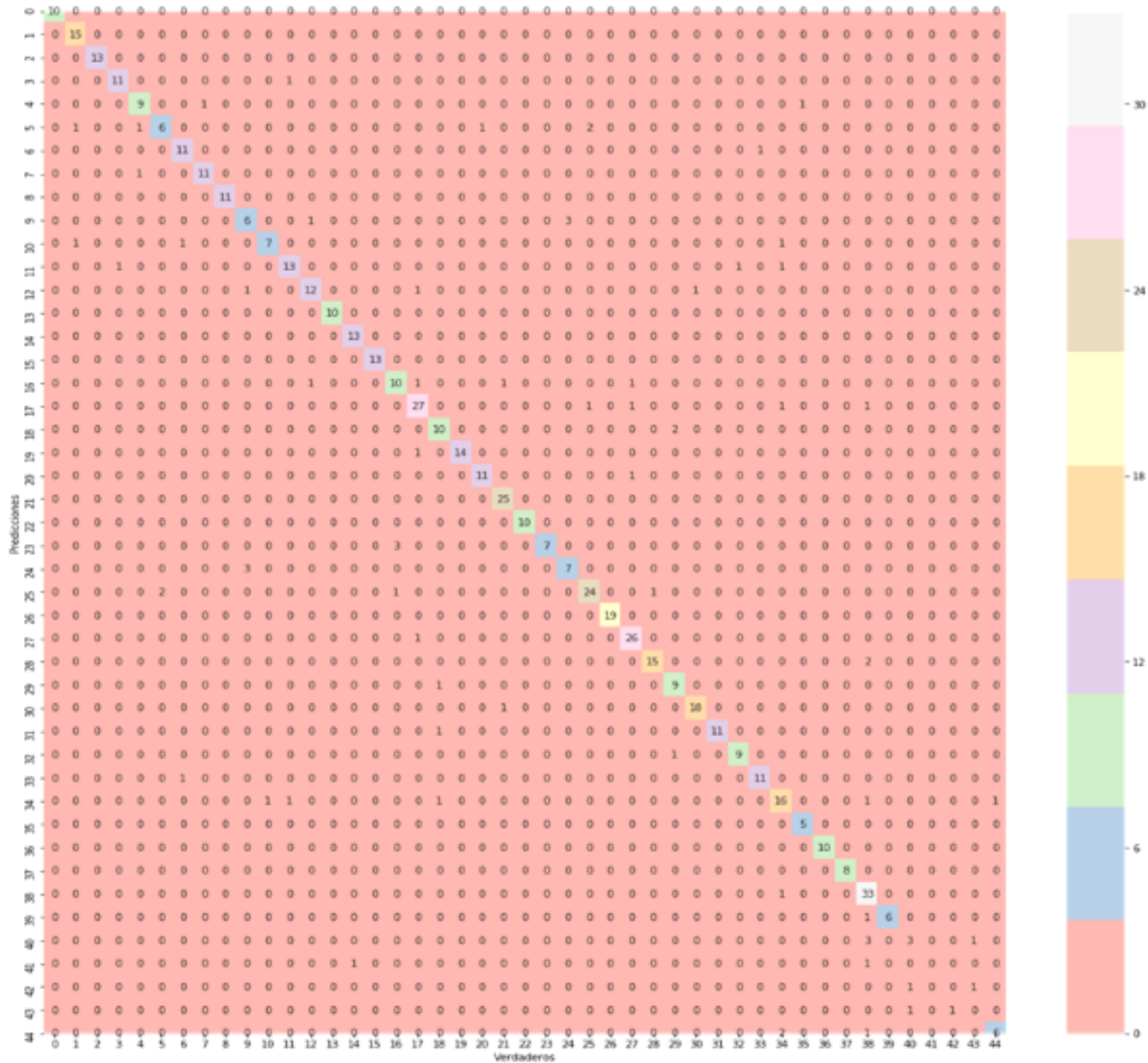


Figura 2: Matriz de confusión modelo de regresión logística con representación TF-IDF  
Fuente: elaboración propia

Como puede observarse en la figura 2, los valores que muestra la diagonal, que son los aciertos, son los más altos, lo que indica que por cada predicción, siempre se tienen mayor número de aciertos que de errores, por ejemplo, para la clase 17 que hace referencia a la respuesta **“A los candidatos e interesados en el examen se les comparte exámenes realizados en semestres anteriores para que conozcan la dinámica del examen y les sirva como material de estudio (En drive estan “material de estudio francés” y “material de estudio inglés”). No requiere solicitud.”**, se tienen 27 predicciones acertadas de 31, indicando así solo 4 fallas.

A continuación se presente una interacción con el chatbot en la que se aprecia la calidad de las respuestas:

Hola, soy ChatExTIO, el chatbot del Examen de Traductor e Intérprete Oficial de la Universidad de Antioquia. ¡Bienvenido! ¿En qué le puedo ayudar?

>Quiero saber cuánto cuesta el examen  
 El examen tiene un costo de \$1'060.000 pesos colombianos (tarifa plena) y \$954.000 pesos colombianos para estudiantes y egresados del programa de traducción de la Universidad de Antioquia.

>perfecto  
 Muy bien, cuénteme, ¿En qué más puedo ayudarle?

>Puedo pagarlo financiado o debe ser de contado  
 El pago del examen es de contado. Pague en línea utilizando su cuenta de ahorros, corriente o tarjeta de crédito a través de "Pagos electrónicos en línea", o imprima el formato de consignación y diríjase a una de las entidades bancarias señaladas en el formato.

>umm, ok  
 Muy bien, cuénteme, ¿En qué más puedo ayudarle?

>En caso de no poder presentar el examen se devuelve el dinero  
 En caso de tener algún inconveniente para la presentación del examen comunicarse a la Escuela de Idiomas. Prof. Gabriel Quiroz, teléfono (57)(4) 219 8799 o escribir al correo electrónico: [examentraductorinterpreteoficial@udea.edu.co](mailto:examentraductorinterpreteoficial@udea.edu.co).

>Está bien  
 Muy bien, cuénteme, ¿En qué más puedo ayudarle?

>¿En qué fechas hay inscripciones?  
 El examen se realiza dos veces al año en las fechas estipuladas por la comisión del examen. Debe ingresar al portal y verificar el cronograma:  
<http://www.udea.edu.co/wps/portal/udea/web/inicio/unidades-academicas/idiomas/extension/examen-traductor-interprete-oficial>.

>ok  
 Muy bien, cuénteme, ¿En qué más puedo ayudarle?

>¿De qué trata la charla informativa?  
 La charla informativa no es obligatoria, es de carácter informativo donde se comparte información general sobre el examen que también se envía por correo electrónico a los candidatos del examen. Puede asistir a ella sin haber realizado la inscripción en el sistema y no tiene costo.

>De acuerdo  
 Muy bien, cuénteme, ¿En qué más puedo ayudarle?

>Es todo, muchas gracias  
 Hasta pronto. Vuelva y cualquier inquietud con gusto será atendida. Para peticiones, quejas, reclamos y sugerencias por favor comunicarse a la Escuela de Idiomas. Prof. Gabriel Quiroz, teléfono (57)(4) 219 8799 o escribir al correo electrónico: [examentraductorinterpreteoficial@udea.edu.co](mailto:examentraductorinterpreteoficial@udea.edu.co)

*Figura 3: Interacción con chatbot implementando modelo de regresión logística con representación TF-IDF*  
 Fuente: elaboración propia

Con la interacción presentada en la figura 3, y los resultados de la tabla 11, el modelo de regresión logística con representación TF-IDF, presenta una eficiencia que se puede considerar aceptable para ser llevado a producción, con un 88% de efectividad en las respuestas, por tanto, el desarrollo del back-end de la solución queda representado por este modelo.

Con Django como tecnología de desarrollo web, se encontró una forma de desarrollar el *front-end* de manera ágil y sencilla. Este framework implementa una arquitectura *Model Template View* (MTV), que se describe a continuación:

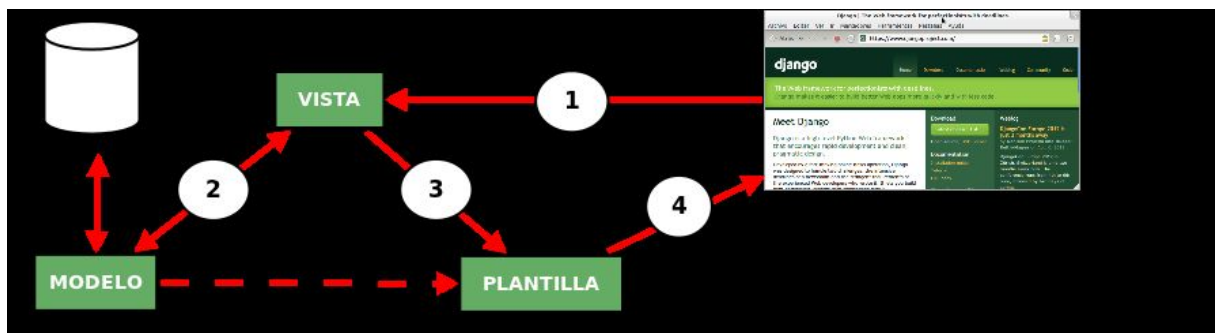


Figura 4: Funcionamiento de la arquitectura *Model Template View* (MTV) de Django

Fuente: Recuperado de “Curso Django: Entendiendo como trabaja Django”, de Infante Montero, S. 30 de Abril de 2012 obtenido de <http://www.maestrosdelweb.com/curso-django-entendiendo-como-trabaja-django/>

En la figura 4 se aprecian los 3 módulos de la arquitectura Django, donde:

**Modelo:** Es el que define los datos almacenados, se encuentra en forma de clases de Python, lo que permite indicar y controlar el comportamiento de los datos.

**Vista:** Es representada en forma de funciones en Python, básicamente determina qué datos serán visualizados ya que contiene toda la lógica necesaria para devolver una respuesta.

**Plantilla:** Es básicamente una página HTML, permiten acomodar el resultado que devuelve la vista. No solamente permiten crear contenido HTML, también XML, CSS, Javascript, AJAX.

El funcionamiento del sistema, como aparece en la figura 4, se describe de la siguiente manera:

1. El navegador envía una solicitud
2. La vista interactúa con el modelo para obtener los datos
3. La vista llama a la plantilla correspondiente
4. La plantilla renderiza la respuesta a la solicitud del navegador

De este modo, con el framework Django se hizo el desarrollo del *front-end*, donde se implementó el tutorial de Muiruri (2018) en el que se hicieron los ajustes necesarios para la adaptación del *back-end* propuesto en el presente trabajo.

A continuación se presenta en la figura 5, una de las interacciones con ChatExtIO (Chatbot del Examen de Traductor e Intérprete Oficial):



Hola

Hola, soy ChatExTIO, el chatbot del Examen de Traductor e Intérprete Oficial de la Universidad de Antioquia. ¡Bienvenido!. ¿En qué le puedo ayudar?

Quiero inscribirme al examen, qué debo hacer?

La inscripción la puede realizar en el siguiente enlace: <https://goo.gl/eFJ4mS> . En "Palabra clave" escribe la palabra "traductor". A continuación le aparecerán dos resultados, "CHARLA INFORMATIVA" Y "EXAMEN DE ACREDITACIÓN", el primero es para la charla informativa y el segundo para el examen en sí. El proceso de inscripción para ambos es el mismo: Al final de la línea verá dos botones azules; en el primero puede ver más información sobre la actividad, el segundo es para realizar su inscripción. Dé click sobre el segundo. El sitio la redirigirá a un formulario que debe diligenciar con sus datos personales. Al finalizar, presione "Guardar". DEBE CERCORARSE DE ACEPTAR " Autorizo tratamiento de datos personales " Y " Acepto los términos y condiciones del servicio ". Para confirmar su pre-inscripción, diríjase a este enlace: <https://goo.gl/jzh1lD> y entre con su tipo y número de documento. En este sitio, que es el historial de cursos a los que se ha inscrito, deberían aparecer la charla informativa y el examen en caso de haberse inscrito para ambos. En este mismo sitio tendrá la opción de descargar el formulario de pago o cancelar el valor del examen en línea. El proceso de registro terminará cuando haya pagado y enviado su colilla de pago (bien sea escaneada, si canceló en una entidad bancaria o la captura de pantalla del certificado de pago que da la plataforma de su banco cuando paga en línea).

¿Cuánto cuesta el examen?

El examen tiene un costo de \$1'060.000 pesos colombianos (tarifa plena) y \$954.000 pesos colombianos para estudiantes y egresados del programa de traducción de la Universidad de Antioquia.

genial!

Muy bien, cuénteme, ¿En qué más puedo ayudarle?

¿Cómo puedo pagar?

El pago del examen es de contado. Pague en línea utilizando su cuenta de ahorros, corriente o tarjeta de crédito a través de "Pagos electrónicos en línea", o imprima el formato de consignación y diríjase a una de las entidades bancarias señaladas en el formato.

¿Dónde consigo el formato?	El recibo de consignación o factura se descarga de la página web del examen después de realizar la inscripción.
Muy bien	Muy bien, cuénteme, ¿En qué más puedo ayudarle?
¿Dónde se presenta el examen?	El examen se realiza en la Universidad de Antioquia ubicada en Medellín. Dirección: calle 67 No. 53 - 108. Para indicaciones de ubicación puede comunicarse al teléfono (57)(4) 219 8799.
¿Tienen algún examen de preparación?	A los candidatos e interesados en el examen se les comparte exámenes realizados en semestres anteriores para que conozcan la dinámica del examen y les sirva como material de estudio (En drive estan <input type="checkbox"/> material de estudio francés <input type="checkbox"/> y <input type="checkbox"/> material de estudio inglés <input type="checkbox"/> ). No requiere solicitud.
¿Cuántas preguntas son?	El examen no es por preguntas sino por textos y son 3 textos además de un cuarto en formato de audio.
¿De qué tratan?	El examen consta de 3 textos y un audio. El audio siempre se trata de un discurso de un político o diplomático. Los 3 textos adicionales son textos científico técnicos, generalmente de medicina, biología o farmacéutica y textos jurídicos como registros civiles, formularios, contratos, etc. También habrá un texto de lenguaje general para hacer interpretación a vista.

Excelente!

Muy bien, cuénteme, ¿En qué más puedo ayudarle?

¿Cuánto se tardan los resultados?

Los resultados del examen se dan a conocer 60 días hábiles después de la presentación del examen. Son publicados en la página <http://www.udea.edu.co/wps/portal/udea/web/inicio/unidades-academicas/idiomas/extension/examen-traductor-interprete-oficial>.

¿Hay devolución de dinero por algún motivo?

En caso de tener algún inconveniente para la presentación del examen comunicarse a la Escuela de Idiomas. Prof. Gabriel Quiroz, teléfono (57)(4) 219 8799 o escribir al correo electrónico: [examentraductorinterpreteoficial@udea.edu.co](mailto:examentraductorinterpreteoficial@udea.edu.co).

¿Cuánto dura el examen?

El tiempo para el desarrollo del examen se distribuye de la siguiente manera: Competencia ESCRITA: 2 horas. Competencia ORAL: 10-15 minutos por turnos de máximo 4 personas.

¿Cuándo es el examen?

El examen se realiza dos veces al año en las fechas estipuladas por la comisión del examen. Debe ingresar al portal y verificar el cronograma: <http://www.udea.edu.co/wps/portal/udea/web/inicio/unidades-academicas/idiomas/extension/examen-traductor-interprete-oficial>.

¿Cómo se aprueba el examen?



Para la calificación del examen se procede de esta manera: Dos jurados que se desconocen entre sí revisan las pruebas y emiten un concepto basados en la plantilla de evaluación. Cada uno de los cuatro textos del examen (2 de interpretación y 2 de traducción) se calificarán sobre una base de 100 puntos. Los errores de traducción o interpretación se valorarán como puntos negativos, cuyo total será restado en cada sección de los 100 puntos asignados a cada una de ellas. El resultado final del examen se reportará como APROBADO o REPROBADO, según los resultados obtenidos por el candidato en las dos (2) pruebas evaluadas (Interpretación y Traducción). Para aprobar el examen es indispensable haber aprobado ambas pruebas del examen que totalizan 4 secciones (2 modalidades de interpretación y 2 de traducción escrita). Una prueba podrá aprobarse con un puntaje de 90 puntos al PROMEDIAR las 2 secciones de dicha prueba. Puede encontrar mayor información en el enlace <http://www.udea.edu.co/wps/portal/udea/web/inicio/idades-academicas/idiomas/extension/examen-traductor-interprete-oficial>.

¿Quién califica el examen?

Para la calificación del examen se procede de esta manera: Dos jurados que se desconocen entre sí revisan las pruebas y emiten un concepto basados en la plantilla de evaluación. Cada uno de los cuatro textos del examen (2 de interpretación y 2 de traducción) se calificarán sobre una base de 100 puntos. Los errores de traducción o interpretación se valorarán como puntos negativos, cuyo total será restado en cada sección de los 100 puntos asignados a cada una de ellas. El resultado final del examen se reportará como APROBADO o REPROBADO, según los resultados obtenidos por el candidato en las dos (2) pruebas evaluadas (Interpretación y Traducción). Para aprobar el examen es indispensable haber aprobado ambas pruebas del examen que totalizan 4 secciones (2 modalidades de interpretación y 2 de traducción escrita). Una prueba podrá aprobarse con un puntaje de 90 puntos al PROMEDIAR las 2 secciones de dicha prueba. Puede encontrar mayor información en el enlace <http://www.udea.edu.co/wps/portal/udea/web/inicio/idades-academicas/idiomas/extension/examen-traductor-interprete-oficial>.



*Figura 5:* Interacción con ChatExTIO

Fuente: elaboración propia

Con la finalidad de acceder al chatbot vía web, y para que la solución fuera funcional, se realizó el despliegue de la aplicación en Heroku. Para esto, básicamente se agregaron los archivos necesarios y se realizaron las configuraciones pertinentes en los elementos del proyecto ubicado en el repositorio <https://github.com/adriruizo/ChatExTIO>; luego, después de la ejecución de los comandos necesarios se realizó la publicación en la nube de la solución desarrollada en el presente trabajo, obteniendo como acceso a ChatExTIO la siguiente url:

<https://chatbotextio.herokuapp.com/>

## Conclusiones

En el presente trabajo se presentó el desarrollo de un chatbot en busca de automatizar el proceso de respuesta ante las inquietudes de los interesados en el examen de traductor e intérprete oficial que proporciona la Escuela de Idiomas de la Universidad de Antioquia.

En primera instancia, se consiguió representar bajo 3 estrategias los datos que fueron proporcionados en relación al examen de traductor e intérprete oficial. Estas representaciones consistieron en llevar el texto en lenguaje natural a estructuras computacionales que para el presente trabajo fueron espacios vectoriales.

Posteriormente se evaluaron diferentes modelos computacionales donde se plantearon 7 algoritmos de recuperación de información que fueron analizados con los tipos de representaciones de datos, para así determinar cuál proporcionaba los mejores resultados, encontrando la mejor eficiencia en la representación TF-IDF y el modelo de Regresión Logística.

De esta manera se construyó el *back-end* para el sistema de diálogo con el modelo de Regresión Logística y la representación TF-IDF, que de acuerdo a todas las pruebas realizadas fué el modelo con los mejores resultados generando una eficiencia del 88% en las respuestas.

Para el desarrollo del *front-end* se implementó la tecnología Django, un framework muy competitivo en el mercado de uso libre con muy buenas características que dieron lugar a un desarrollo ágil y acorde a la necesidad del presente trabajo.

Finalmente se consiguió desplegar la solución en una plataforma de servicio en la nube, flexible, fácil de usar y gratuita como Heroku. La solución desarrollada es de acceso libre y se puede encontrar aquí: <https://chatbotextio.herokuapp.com/>

Vale la pena resaltar cómo en la actualidad, los agentes virtuales son implementados en muchas organizaciones, siendo especialmente aplicados al servicio de atención al cliente respondiendo a las preguntas de los usuarios de manera personalizada e inmediata, las 24 horas del día. De la misma manera, la solución desarrollada en este trabajo condujo a un chatbot como asistente virtual para lo concerniente al examen de traductor e intérprete oficial, donde las respuestas a las consultas de los usuarios son encontradas en un corpus predefinido, obteniendo así un sistema de diálogo basado en corpus, con métodos de recuperación de información, que de acuerdo a la representación de los datos usando el esquema TF-IDF y al algoritmo de recuperación de la información dotado de inteligencia artificial con el modelo de Regresión logística, permite entregar una respuesta coherente a las dudas en torno al Examen de Traductor e Intérprete Oficial de la Universidad de Antioquia.

## Referencias Bibliográficas

- Betancourt, G. A. (2005).** Las máquinas de soporte vectorial (SVMs). *Scientia et Technica*, 1(27).
- Bird, S., Klein, E., & Loper, E. (2009).** *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc."
- Breiman, L. (2001).** Random forests. *Machine learning*, 45(1), 5-32.
- Brownlee, J. (14 de Marzo de 2018).** *A Gentle Introduction to Sparse Matrices for Machine Learning*. Obtenido de <https://machinelearningmastery.com/sparse-matrices-for-machine-learning/>
- Cambizaca, T., & Francisco, A. (2018).** *Desarrollo de un chatbot que ayude a responder a preguntas frecuentes referentes a becas en la Universidad Técnica Particular de Loja* (Bachelor's thesis).
- Cárdenas, Y. C. M., & Bautista, D. W. R. (2012).** Mejores prácticas de gestión para la calidad de los servicios en tecnologías de información. *Gerencia Tecnológica Informática*, 11(29), 47-58.
- ChatterBot. (2019).** Recuperado el 3 de Enero de 2020, de <https://chatterbot.readthedocs.io/en/stable/>
- Chollet, F. (2015).** Keras documentation. *keras.io*.
- Cobos Torres, J. C. (2013).** *Integración de un Chatbot como habilidad de un robot social con gestor de diálogos* (Master's thesis, ESPAÑA/Facultad de Ingeniería de Sistemas y Automática-Universidad Carlos III de Madrid/2013).

**Comendador, B. E. V., Francisco, B. M. B., Medenilla, J. S., & Mae, S. (2015).**

Pharmabot: a pediatric generic medicine consultant chatbot. *Journal of Automation and Control Engineering*, 3(2).

**Cramer, J. S. (2002).** The origins of logistic regression.

**Cutler, A. (2013).** Trees and random forests. *NIH IR15AG037392-01*, 92.

**Divya, S., Indumathi, V., Ishwarya, S., Priyasankari, M., & Devi, S. K. (2018).** A

self-diagnosis medical chatbot using artificial intelligence. *Journal of Web Development and Web Designing*, 3(1), 1-7.

**Django. (2015).** Recuperado el 10 de Diciembre de 2019, de

<https://docs.djangoproject.com/es/3.0/>

**djangoirls. (2014).** Recuperado el 17 de Enero de 2020 de:

<https://tutorial-extensions.djangoirls.org/es/heroku/>

**Dole, A., Sansare, H., Harekar, R., & Athalye, S. (2015).** Intelligent Chat Bot for

Banking System. *International Journal of Emerging Trends & Technology in Computer Science*, 4(5), 49-51.

**Estrada Cutimbo, L. (2018).** Implementar chatbot basado en inteligencia artificial

para la gestión de requerimientos e incidentes en una empresa de seguros.

**Firth, J. (1957).** A synopsis of linguistic theory 1930-1955. *Studies in Linguistic*

*Analysis*.

**Fix, E. and Hodges, J.L. (1951).** Discriminatory Analysis, Nonparametric

Discrimination: Consistency Properties. Technical Report 4, USAF School of Aviation Medicine, Randolph Field.

**Garcia Brustenga, G., Fuertes Alpiste, M., & Molas Castells. (s.f).** Briefing paper:

los chatbots en educación.

**García Ferrero, I. (2018).** Estudio de Word Embeddings y métodos de generación de Meta Embeddings.

**Gensim. (9 de Julio de 2019).** Recuperado el 9 de Septiembre de 2019, de  
models.word2vec-Word2vec embeddings:  
<https://radimrehurek.com/gensim/models/word2vec.html>

**Grave, E., Bojanowski, P., Gupta, P., Joulin, A., & Mikolov, T. (2018).**  
Aprendizaje de vectores de palabras para 157 idiomas

**Griol, D., Callejas, Z., Arroyo, Á., Patricio, M. Á., Molina, J. M., & López-Cózar, R. (2010).** Integración de los Sistemas de Diálogo para la Interacción en Redes Sociales. *Procesamiento del lenguaje natural*, 44.

**Infante Montero, S. (30 de Abril de 2012).** *Curso Django: Entendiendo como trabaja Django*. Obtenido de  
<http://www.maestrosdelweb.com/curso-django-entendiendo-como-trabaja-django/>

**Jaccard, P. (1908).** Nouvelles recherches sur la distribution florale. Bulletin de la Société Vaudoise des Sciences Naturelles, 44, 223–227.

**Jaccard, P. (1912).** The distribution of the flora of the alpine zone. *New Phytologist*, 11, 37–50.

**Jurafsky, D., & Martin, J. (2017).** *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. New Jersey. Prentice Hall.

**Kulkarni, C. S., Bhavsar, A. U., Pingale, S. R., & Kumbhar, S. S. (2017).** Bank chat bot-an intelligent assistant system using nlp and machine learning. *International Research Journal of Engineering and Technology*, 4, 2374-2376.

- Lindenbaum, J., Wiggins, A., & Henry, O. (2007).** *Heroku*. Obtenido de <https://www.heroku.com/>
- Luhn, H. P. (1957).** A statistical approach to the mechanized encoding and searching of literary information. *IBM Journal of Research and Development*, 1(4), 309–317.
- Muiruri, S. (13 de Agosto de 2018).** *Planeta ChatBot*. Obtenido de Tutorial: Implementando un chatbot en Django: <https://planetachatbot.com/tutorial-implementando-chatbot-en-django-fl185653d60a2>
- Nagesh Singh, C. (Mayo de 2019).** *KDnuggets*. Obtenido de Build Your First Chatbot Using Python & NLTK: <https://www.kdnuggets.com/2019/05/build-chatbot-python-nltk.html>
- Oliphant, T. E. (2006).** *A guide to NumPy* (Vol. 1, p. 85). USA: Trelgol Publishing.
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Vanderplas, J. (2011).** Scikit-learn: Machine learning in Python. *Journal of machine learning research*, 12(Oct), 2825-2830.
- Rosenblatt, F. (1958).** The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6), 386.
- Sosa, Eduardo. (1997).** Procesamiento del lenguaje natural: revisión del estado actual, bases teóricas y aplicaciones (Parte I). [En línea] Enero de 1997. [Citado el: 1 de Junio de 2019.] [http://www.elprofesionaldelainformacion.com/contenidos/1997/enero/procesamiento\\_del\\_lenguaje\\_natural\\_revisin\\_del\\_estado\\_actual\\_bases\\_tericas\\_y\\_aplicaciones\\_parte\\_i.html](http://www.elprofesionaldelainformacion.com/contenidos/1997/enero/procesamiento_del_lenguaje_natural_revisin_del_estado_actual_bases_tericas_y_aplicaciones_parte_i.html).



- Sparck Jones, K. (1972).** A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28(1), 11–21.
- Tolosa, G. H., & Bordignon, F. R. A. (1999).** Revisión: tecnología de agentes de software. *Ciencia de la Información*, 28(3).
- Van Rossum, G., & Drake Jr, F. L. (1995).** *Python reference manual*. Amsterdam: Centrum voor Wiskunde en Informatica.
- Vapnik, V.N.** An overview of statistical learning theory. *Neural Netw. IEEE Trans.* Vol. 10, nº 5, pp. 988–999, 1999.
- Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., ... & van der Walt, S. J. (2019).** SciPy 1.0--fundamental algorithms for scientific computing in Python. *arXiv preprint arXiv:1907.10121*.
- Walker, S. H., & Duncan, D. B. (1967).** Estimation of the probability of an event as a function of several independent variables. *Biometrika*, 54(1-2), 167-179.
- Yamada, I., Asai, A., Shindo, H., Takeda, H., & Takefuji, Y. (2018).** Wikipedia2Vec: An Optimized Tool for Learning Embeddings of Words and Entities from Wikipedia.
- Zapata, C. , & Mesa, J. (2009).** Los modelos de diálogo y sus aplicaciones en sistemas de diálogo hombre-máquina: Revisión de la literatura. *DYNA*, 76(160), 293-303.