Procesamiento de lenguaje aplicado a herramientas de búsqueda de información

Proyecto de Grado
Ingeniería en computación

Nikolai Jolodkow
Tiziana Romani

Ingeniería en Computación
Facultad de Ingeniería
Universidad de la República
Montevideo – Uruguay
Diciembre de 2021

Procesamiento de lenguaje aplicado a herramientas de búsqueda de información

Nikolai Jolodkow
Tiziana Romani

Proyecto de Grado de Ingeniería en Computación presentada al Programa de Grado en Ingeniería en Computación, Facultad de Ingeniería de la Universidad de la República, como parte de los requisitos necesarios para la obtención del título de Ingeniero en Computación

Directores:
Aiala Rosá
Juan José Prada

Montevideo – Uruguay
Diciembre de 2021

INTEGRANTES DEL TRIBUNAL DE DEFENSA DE PROYECTO DE GRADO

Mathias Etcheverry

Diego Garat

Adriana Marotta

Montevideo – Uruguay
Diciembre de 2021

Resumen

El año 2020 fue un año distinto, un año con muchos cambios provocados por una pandemia a nivel mundial. Esta pandemia provocaría que a mediados de marzo de 2020 comience un confinamiento a nivel nacional que posteriormente dejaría a muchos residentes del Uruguay sin empleo. Debido a esta grave situación, el Instituto Nacional de Empleo y Formación Profesional (INEFOP) [1], encargado en gran parte de capacitar personas en seguro de desempleo, se vio fuertemente afectado en cuanto a su servicio de atención al público dada la gran demanda que se generó. Motivados por la problemática planteada, se decide impulsar el proyecto de grado descrito en este informe.

Este proyecto propone la aplicación de técnicas de Procesamiento de Lenguaje Natural (PLN) para la construcción de herramientas que den soporte a sistemas de búsqueda de información. El objetivo principal es minimizar tiempos de respuesta y facilitar el acceso de la información. En este proyecto se construye un sistema capaz de contestar preguntas con un tiempo de respuesta acotado, cuya respuesta se encuentra en la página web de INEFOP. Mediante técnicas de detección de similitud textual y recuperación de información, el sistema desarrollado tiene la capacidad de emparejar la pregunta ingresada con preguntas almacenadas si son similares, o en caso contrario, retornar un enlace dentro de la página web de INEFOP donde podría encontrarse la respuesta.

Entre los aspectos destacables del proyecto se presenta la construcción en su completitud de un sistema con las características mencionadas. El prototipo desarrollado cuenta con una interfaz web simple para comunicarse con el sistema, esta permite retroalimentación con la finalidad de mejorar la eficiencia del sistema. El servidor disponibiliza distintos servicios y consume y consulta información de una base de datos. Entre las técnicas y métodos explorados a lo largo del proyecto se destacan las redes neuronales siamesas, *web scraping*, la utilización del motor de búsqueda Elasticsearch y gran variedad de herramientas y métricas de PLN.

Palabras clave: Procesamiento de lenguaje natural, aprendizaje automático, detección de preguntas similares, recuperación de información.

Tabla de Contenidos

1. Introducción

INEFOP[1] fue creado en octubre del 2008 como una persona pública no estatal (de la misma manera que el LATU). Su principal cometido es ejecutar políticas de formación profesional y fortalecimiento del empleo de trabajadores y trabajadoras del Uruguay, administrando el fondo de reconversión laboral.

Su misión incluye fomentar la capacitación laboral, el desarrollo de políticas públicas para la formación de profesionales, brindar asistencia técnica para la creación y el desarrollo de empresas privadas, y la investigación y exposición de estudios prospectivos sobre el mercado de trabajo. Su visión gira en torno a la formación profesional del sector privado. Actualmente existen 3 gerencias operativas principales en la empresa:

- Formación profesional: forma trabajadores en seguro de desempleo.
- Empleo: se encarga por ejemplo de ofrecer cursos para trabajadores activos.
- Empresas: ofrece capacitaciones para empresas, emprendedores, asistencia técnica, etc.

1.1 Problema y motivación

Agustin Guerra y Pablo Lazo, usuarios responsables por parte de INEFOP, ambos del área de Formación Profesional de INEFOP, definen la cantidad de atención requerida por los usuarios como su motivación principal en la creación de una solución. Explican cómo cientos de personas diariamente se contactan a través de los distintos medios (teléfono, mail) para realizar consultas cuya respuesta se encuentra en la página web. Admiten que el formato de la página web no es amigable al usuario, y que a veces incluso resulta confuso y difícil para sus mismos compañeros de trabajo encontrar la información que están buscando. Afirman, sin embargo, que casi la totalidad de la información debería estar disponible en la web, por lo que es posible etiquetar la situación como un problema de accesibilidad, más que de disponibilidad. Los principales afectados son aquellos operarios de INEFOP que deben diariamente atender a un público que falló en hacer un correcto uso de un recurso web.

Como estudiantes que enfocaron gran parte de su carrera en la inteligencia artificial (IA), nos maravilló la idea de poder aplicar muchos de los conocimientos adquiridos en un problema real y tan cercano. El proceso de tener que entender las necesidades del usuario responsable fue una las mayores motivaciones que tuvimos al iniciar este proyecto, lo vimos como una oportunidad que enriquecería nuestra carrera profesional.

[1] http://www.inefop.org.uy/

1.2 Objetivos

El objetivo principal de este trabajo radica en aportar asistencia a INEFOP en el problema mencionado sin dejar de lado nuestro objetivo académico de profundizar en el área del procesamiento de lenguaje natural aplicado en un problema real. Una solución que se desprende instantáneamente, es una reestructuración de la página web. Aunque ésta sea factible, no cumple con el objetivo académico por lo que se descarta.

Se aclara desde un comienzo que el instrumento a desarrollar no necesariamente tiene que ser enfocado al público general, sino que puede apuntar a los trabajadores de INEFOP, brindando facilidad y velocidad a la hora de buscar una respuesta para contestar al usuario. Idealmente, la solución final cubriría las 3 gerencias operativas principales.

En este trabajo se propone una herramienta que asista aquellos usuarios que buscan información en la web de INEFOP. Consta de dos instancias. La primera es de identificación de preguntas frecuentes donde se presenten cuestiones semánticamente equivalentes a lo ingresado junto a su respuesta. Y la siguiente implementa una búsqueda de respuestas en la totalidad de la web, retornando links a aquellas páginas que podrían contener una contestación o información relevante a lo que se busca.

1.3 Insumos

En la etapa de exploración, se nos facilitaron diversos materiales por parte del usuario responsable. Entre estos materiales se encuentran documentos con información esencial para los operarios de INEFOP, varios de estos se encuentran en la web y contienen la respuesta a muchas de las preguntas que realizan potenciales usuarios. Se los considera documentos indispensables y suponen una base de la cual obtener información. La realidad es que son documentos largos con mucho formalismo que pueden abrumar a cualquier interesado. Un ejemplo son las *Bases Generales*, de más de 50 páginas, donde es posible leer reglas para la presentación y evaluación de ofertas, ejecuciones de contratos, sanciones, pagos de viáticos y subsidios, e instrumentos varios para empresas. Esta información asimismo es posible encontrarla distribuida por la web. También se proporcionan documentos de FAQ y plantillas de preguntas usuales. Este material se detalla en profundidad en la segunda sección del capítulo 3.

1.4 Resultados esperados

Como objetivo nos proponemos generar un prototipo que explore el área de procesamiento de lenguaje natural, obtenga resultados realistas y en lo posible útiles. Esto se puede traducir a:

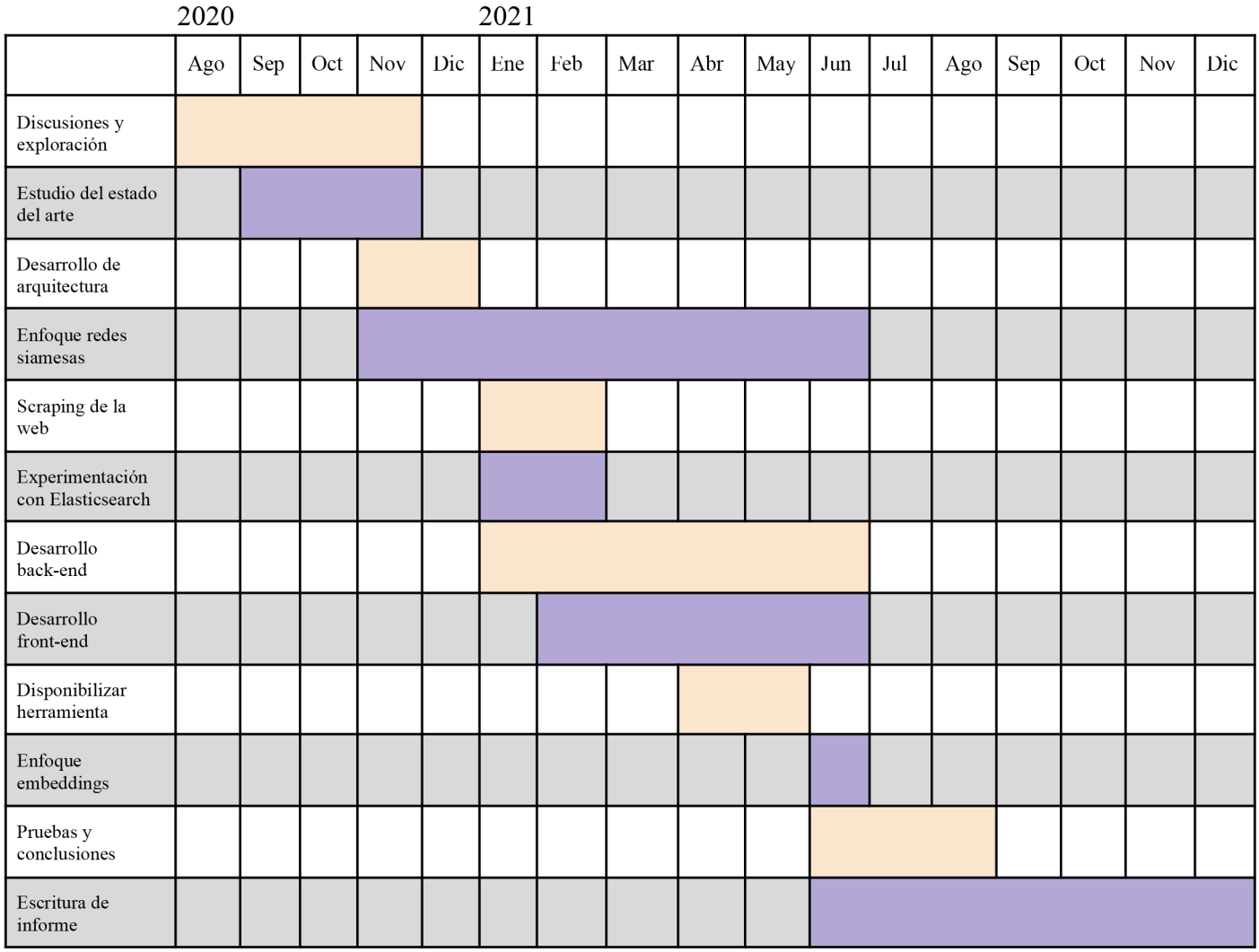
- ❏ Relevamiento del estado del arte para las áreas a tratar del PLN junto con el estudio de metodologías y arquitecturas actualmente en uso. Se debe detallar la información relevada, resultados intermedios y pruebas realizadas.
- ❏ Diseño de una arquitectura que haga uso de lo relevado y sea una solución para el problema planteado. Incluyendo una descripción de los componentes y las tecnologías empleadas.
- ❏ Construir un prototipo que tenga una usabilidad adecuada, el cual nos proponemos habilitar de forma temprana para uso del usuario responsable, de esta forma pudiendo verificar que cumpla con sus expectativas.

1.5 Cronograma

En el Cuadro 1.1 se presentan las principales etapas atravesadas y un aproximado del tiempo empleado en cada una.

1.6 Estructura del Documento

Este documento se encuentra organizado en 7 capítulos, seguido de una sección bibliográfica, un glosario y el anexo. Dentro de los 7 capítulos, el segundo incluye una presentación sobre conceptos de procesamiento de lenguaje natural necesarios para comprender el trabajo en su totalidad, también realiza una descripción de los problemas dentro del área que se intentarán solucionar en el desarrollo del proyecto. El tercer capítulo incluye un análisis del problema con una descripción sobre los materiales presentados por el usuario responsable los cuales se utilizan a la hora de plantear una solución a alto nivel, presentada al final de ese mismo capítulo. El capítulo cuatro presenta dos enfoques desarrollados que intentan solucionar el problema de similitud de textos presentados en el capítulo dos, seguido de una comparación de resultados. El capítulo cinco presenta la resolución del problema de recuperación de información con resultados obtenidos. El capítulo seis describe la implementación llevada a cabo, presentando los distintos módulos que componen el sistema creado, también menciona las dificultades técnicas que se presentaron a lo largo del proyecto. El capítulo siete presenta las conclusiones y trabajo a futuro.



Cuadro 1.1: Diagrama de Gantt de las tareas más importantes desarrolladas en el transcurso del proyecto de grado.

2. Marco teórico

Este capítulo presenta conceptos teóricos fundamentales empleados a lo largo del proyecto, incluye nociones de aprendizaje automático, procesamiento de lenguaje natural y redes neuronales. Aparte, se incluyen trabajos y estudios realizados relevantes al problema presentado.

2.1 Inteligencia Artificial

La inteligencia demostrada por una máquina difiere de la inteligencia natural presentada por seres vivos, la cual incluye una conciencia propia y emociones. C. Bartneck et al. [2] presentan varias definiciones. Se puede definir como el campo que estudia y desarrolla "agentes inteligentes" o dispositivos que perciban un determinado entorno y tomen acciones para lograr su objetivo. Informalmente, otra descripción podría incluir que investiga y crea máquinas que mimetizan una función cognitiva asociada con la mente humana, tal como aprender o resolver un problema.

En 1950 Alan Turing plantea que es posible determinar si una máquina es inteligente si logra exhibir comportamientos indistinguibles de los de un humano. Para esto propone el conocido Test de Turing, prueba que consiste en que una máquina y un humano respondan una serie de preguntas. Si un tercer humano no logra determinar cuál es la máquina en base a las respuestas dadas, entonces la máquina pasa la prueba. Fue un hito que hasta hoy en día genera controversia. El intento de definir la inteligencia artificial con respecto a una actividad testeable, como lo es una conversación, no satisface a todo el público. El rango de opiniones va desde exacerbado hasta no suficientemente demandante. [3]

Los problemas tradicionales encontrados en esta área incluyen representación de conocimiento, razonamiento, planificación, aprendizaje, procesamiento de lenguaje natural, y la percepción. [4] Que el campo se haya fundado bajo la suposición de que la inteligencia humana puede ser precisamente descrita como para que una máquina pueda simularla genera controversia hoy en día. Se plantean argumentos filosóficos sobre la ética de crear un ser con el potencial de una máquina que pueda realizar procesos cognitivos. Personas renombradas como Elon Musk, temen que un progreso desmesurado del campo genere un alto índice de desempleo, resulte en mejor armamento, o llevado a un extremo, que ponga en peligro la raza. [5]

2.2 Procesamiento de Lenguaje Natural

El procesamiento de lenguaje natural (PLN) es una rama de la inteligencia artificial, cuyo objetivo concierne la forma en la que las computadoras procesan y analizan el lenguaje. Se apunta a que sean capaces de "entender" el contenido de un documento. Para una computadora,

la habilidad de procesar el lenguaje como lo hace un humano significa algún tipo de inteligencia. A grandes rasgos hay 6 niveles importantes a comprender dentro del entendimiento del lenguaje. Estos son la fonética, la morfología, la sintaxis, la semántica, la pragmática y el discurso. Resolver al menos algunos de ellos, podría permitir automatizar tareas de extracción, categorización, y organización de la información. [6]

Aparte, creemos conveniente la mención de la ambigüedad, resolver un problema de procesamiento de lenguajes usualmente se reduce a resolver una ambigüedad dentro de los distintos niveles. Una entrada se considera ambigua, si es representable mediante varias estructuras. Por ejemplo, para la frase -*el banco de la plaza*- la palabra banco, puede referirse a un asiento o a una empresa comercial dedicada a operaciones financieras.

Se podría decir que en los últimos 50 años de estudio del PLN, el conocimiento sobre los niveles ya mencionados puede capturarse mediante un pequeño número de modelos formales y teorías. Estos modelos y teorías se han estudiado en ciencias de la computación, matemática y lingüística. Se apoyan en herramientas como máquinas de estado, sistemas de reglas, lógica y teoría de la probabilidad.

Destacamos la teoría de la probabilidad. Esta resulta de mucha utilidad a la hora de resolver problemas de ambigüedad como reconocimiento de entidades o desambiguación de términos. Existen modelos probabilísticos que forman parte de un conjunto de modelos del aprendizaje automático, se entrenan con grandes cantidades de corpus y se enfocan en aprender representaciones automáticamente. En un principio, se hacía uso de métodos simbólicos, como gramáticas, conjuntos de reglas y diccionarios, pero la velocidad a la cual estos conjuntos crecen es exponencial, además de que no arrojan resultados precisos. Haciendo uso del aprendizaje automático es posible eliminar este problema pero aparece un nuevo dilema: la obtención del corpus.

2.2.1 Aprendizaje Automático

El aprendizaje automático[7] es un campo que estudia algoritmos computacionales que mejoran la automaticidad de una tarea mediante la experiencia y el uso de datos. Aparte de tomar conceptos y resultados de la inteligencia artificial, también lo hace de la estadística, teoría de la información, ciencia cognitiva y complejidad computacional, entre otros. En esta disciplina, se construye un modelo y se lo entrena con datos. Su objetivo es realizar predicciones o tomar decisiones sin haber sido específicamente programado para eso.

Estos algoritmos son empleados en diversas áreas, por ejemplo medicina, filtrado y detección de anomalías, reconocimiento de rasgos y voz, visión computacional y procesamiento de lenguaje.

El campo del aprendizaje automático está fuertemente relacionado con el campo de la estadística en términos de métodos pero difieren en su objetivo principal. En la estadística se extraen inferencias de una muestra dada, mientras que en el aprendizaje automático se extraen patrones repetitivos o se generaliza de la experiencia. Esto le permite desempeñar una tarea sobre un ejemplo no incluido en el conjunto de datos experimentales. Se asume que estos datos vienen de una distribución probabilística desconocida y la tarea es construir un modelo sobre el espacio de ocurrencias que sea capaz de producir predicciones precisas.

Existen dos tipos principales de entrenamiento, el supervisado y el no supervisado. En el entrenamiento supervisado, un modelo recibe las entradas junto con las etiquetas o la "respuesta correcta". Se quiere en lo posible que el modelo prediga la mayor cantidad de respuestas correctas, esto se puede cuantificar mediante una función que mide el error cometido. La parte automática ocurre cuando el modelo modifica sus parámetros internos con el objetivo de minimizar esta función. A estos parámetros se les llama pesos.

El entrenamiento no supervisado no requiere que las entradas vengan etiquetadas. El modelo trabaja detectando patrones o información relevante por sí mismo. Hoy en día el éxito de las técnicas supervisadas sobrepasa a las no-supervisadas, pero la realidad es que los humanos y animales aprenden, en general, de forma no supervisada. Descubrimos la estructura del mundo observando, esto nos permite suponer que existe una chance de que eventualmente las técnicas no-supervisadas se desarrollen aún más.

Deep Learning

Deep learning[8] se define como una familia de algoritmos dentro del aprendizaje automático. Originalmente, los métodos convencionales se veían limitados por su incapacidad de procesar entradas *raw*, o datos directamente colectados del dominio. Previo a construir un modelo que reconociera un patrón, se requería de un extenso proceso de ingeniería y conocimientos profundos en el área para diseñar cómo extraer las características importantes de los datos. En cambio, el aprendizaje de representaciones o *feature learning* es un método en el cual se alimenta una máquina con un conjunto de entradas y automáticamente aprende a diferenciar entre las distintas representaciones posibles.

La familia de algoritmos de Deep Learning emplea aprendizaje de representaciones a varios niveles. Se componen módulos simples no lineales que transforman la entrada en cada nivel a una representación un poco más abstracta, se componen en varias capas secuenciales y cada una toma como entrada la salida de la capa anterior. La composición de varias de estas transformaciones simples resultan en una función compleja.

Estos algoritmos, han hecho grandes avances en el campo. Han resultado útiles a la hora de encontrar estructuras complejas dentro de datos con muchas características. Esto ha resultado conveniente para aplicar en dominios científicos, económicos, y políticos. Han mostrado excelencia resolviendo problemas de reconocimiento de imágenes, prediciendo interacciones entre drogas, analizando datos sobre aceleración de partículas, entre otros. Aparte, han demostrado potencial para resolver varias tareas dentro del campo de procesamiento de lenguaje.

Existe una extensa cantidad de arquitecturas dentro del área, *Recurrent Neural Networks, Convolutional Neural Networks, Generative Adversarial Networks* son algunos nombres que se emplean hoy en día. A continuación se describe muy brevemente la arquitectura que se utiliza en el trabajo: *Siamese Networks.*

Una *Siamese Network o* red siamesa es una arquitectura dentro del conjunto de algoritmos de aprendizaje conocidos como redes neuronales que se ven inspirados en el cerebro humano. Mimetizan la forma en que se comunican las neuronas al procesar información. Las redes siamesas se emplean en tareas de distinción y comparación. Consta de dos redes neuronales paralelas que comparten los parámetros internos o pesos. Cada una recibe un input, los procesan conjuntamente y se realiza una comparación entre las salidas. Se emplean en tareas de verificación de firmas, rostros y reducción de dimensionalidad de características. En el capítulo 4 se desarrolla una solución para detectar preguntas similares basado en la funcionalidad de comparación de estas redes.

Word Embeddings

Previo a continuar con la descripción del área, se considera adecuada la mención de la herramienta de representación de texto *word embeddings*. En el siguiente trabajo de Tomas Mikolov et al. [9], ingenieros de Google presentan modelos capaces de generar representaciones vectoriales continuas sobre conjuntos de datos grandes en tiempos razonables y con buena calidad. Hoy en día las representaciones son utilizadas ampliamente en el área del procesamiento de lenguaje. Hay varios modelos accesibles, muchos de esos previstos por empresas como Google o Baidu. También, ha sido utilizada previamente por los estudiantes en diversas instancias de facultad, por lo que se decide emplear esta forma de representación en el desarrollo del trabajo.

La fabricación de estas representaciones vectoriales es posible debido a que palabras que ocurren en contextos similares tienden a tener significados similares. El área de investigación que se ocupa de categorizar palabras basado en las similitudes semánticas y la distribución de estas se le llama semántica distribucional. La idea general de los *word embeddings* recae sobre definir el significado de una palabra basado en su distribución o contexto. Se representa mediante un vector de largo fijo. Esto permite a las palabras con similar significado semántico tener una representación vectorial cercana. La obtención de estos vectores se realiza mediante aprendizaje

por representaciones. El carácter automático lo transforma en un área importante de investigación dentro del PLN, donde grandes empresas dedican recursos a su estudio.[6]

Se han descubierto propiedades interesantes, como que al realizar cálculos con los vectores es posible derivar en vectores de palabras que tengan sentido. Un ejemplo es que si al vector de -*rey*- se le resta el vector de -*hombre*- y se le suma el vector de -*mujer*-, la representación queda cercana al vector de -*reina*-. Este mecanismo ha permitido el desarrollo de modelos de *deep learning* enfocados al lenguaje.

2.2.2 Modelos Actuales

Como se dijo, el entrenamiento de un modelo se basa en la experiencia. Esto evidencia la necesidad de grandes cantidades de ejemplos dado que el lenguaje es un campo extenso. Hoy en día con internet y la nube, el acceso a información se ha simplificado permitiendo avances notorios en el área y la generación de modelos de lenguaje.

En la actualidad, BERT[10] es uno de los proyectos más conocidos. Es un modelo del tipo Transformer entrenado tomando en cuenta una entrada completa, es decir, de un token se considera el contexto izquierdo y derecho, por ello se le llama bidireccional. La fortaleza de BERT es que es posible utilizarlo como un modelo pre-entrenado. Esto significa que se pueden resolver varias tareas agregando únicamente una última capa entrenada específicamente A esta estrategia se le llama *fine tuning*. Ha mejorado resultados en el estado del arte para 11 instancias de problemas del PLN descritas en el trabajo referenciado y se destaca su uso en tareas como contestar una pregunta del texto o predicción de la próxima frase. Tiene una versión exclusiva para el español llamada BETO[11].

No solo se ha facilitado el acceso a la información, sino que aparte, el avance en el mundo del hardware ha permitido mayor y mejor cantidad de recursos trayendo consigo el desarrollo de modelos más grandes. GPT-3[12] es otro ejemplo, un modelo también de tipo Transformer que consta de 175 billones de parámetros. Es la arquitectura de auto regresión lo que lo diferencia de BERT, en cambio emplea observaciones de pasos previos como entrada para la regresión y predecir el valor del próximo paso. GPT-3 no se había liberado al público hasta recientemente.[13]

Actualmente, son muchos y variados los problemas que se tratan de resolver en el PLN, entre ellos homónimos, sinónimos, ambigüedades, sarcasmo, etiquetar POS-tags, generar grafos de dependencia, extraer referencias a una entidad, resúmenes automáticos, traducción, extracción de relaciones, análisis de sentimiento, reconocimiento del habla, segmentación de tópico y más. Las dos áreas de estudio dentro del PLN detalladas a continuación resultaron relevantes para el problema estudiado.

2.2.3 Similitud textual semántica

En el problema de similitud textual, dados dos textos s_1 y s_2, un sistema debe predecir qué tan semánticamente cercanos son. Este tema es de interés y se ha visto abordado en eventos y competencias, como por ejemplo las SemEval en 2012 a 2017[14], donde se desafía a los participantes a desarrollar un modelo que reciba como entrada dos textos y retorne un score de similitud. Los lenguajes empleados incluyen inglés, español, árabe y turco. En particular, se investiga la competencia del 2015 al ser la más reciente cuya tarea contiene la resolución de similitud entre textos en español. El corpus provisto por la organización incluye textos extraídos de la *Wikipedia* y de la *Newswire*. La escala de similitud va de 0 (no relacionados) a 4 (semánticamente equivalentes).

Los organizadores de la competencia, escriben un trabajo[15] donde se detalla la forma en que obtienen el corpus y la línea base del trabajo. Ellos emplean *bag-of-words* para representar los textos y computan distancia coseno entre los dos vectores. *Bag of words* es una forma de representación de documentos, es posible encontrar su descripción en el glosario. Este método se podría considerar puramente léxico, y sus resultados no son relevantes para la actualidad.

De manera similar, en el trabajo de Wanpeng Song et al. [16], aparte de la similitud léxica, consideran una similitud semántica y computan la semejanza textual como una combinación lineal de ambas. La similitud semántica se calcula haciendo uso de WordNet[17], una red jerárquica donde las palabras se organizan en conjuntos de sinónimos con punteros a otros conjuntos. A pesar de que existen algunas versiones de WordNet para el español que permitirían la reproducción de este trabajo, consideramos que no está muy completo y optamos por emplear un método incluido dentro del campo de aprendizaje automático. Nuevos y más actualizados trabajos han surgido en los últimos años que se consideran más relevantes para la actualidad. A continuación se describirán brevemente el conjunto de trabajos estudiados que inspiran la solución desarrollada. Un análisis más detallado puede ser encontrado en el anexo.

P. Neculoiu et al. [18], trabajan sobre la tarea de clasificar nombres para puestos de trabajo. Dentro de un conjunto predefinido, dado un nuevo título deben seleccionar el más parecido. Esto lo hacen mediante representaciones vectoriales o *embeddings*. Plantean que su solución de generar un espacio vectorial es flexible, pudiendo buscar puestos similares, relaciones o clusters basados en la cercanía dentro del espacio. También mencionan que se podría usar los vectores como entrada para un modelo que clasifique. Para generar los vectores se entrena un par de redes siamesas con un conjunto de pares de títulos generado por ellos, algunos semánticamente equivalentes. Luego, se separan las siamesas y se toma una de las redes. Esta se utiliza para generar los vectores y así emparejar dentro del espacio vectorial. Los resultados son muy buenos, superando el 80% de precisión.

J. V. Andrioli de Souza et al. [19], proponen un trabajo sobre detección de similitud entre textos empleando una única arquitectura pero variando el idioma y el campo del texto, en este caso inglés y portugués para periodismo y medicina. Utilizan redes siamesas para computar la similitud. Las representaciones vectoriales las obtienen de modelos ya publicados, así como de un modelo entrenado por ellos. Como novedoso, agregan características léxicas para complementar la red neuronal. Los resultados varían comparados al estado del arte, algunos mejoran y otros empeoran. En las conclusiones se discute que las mejores representaciones vectoriales son aquellas pre-entrenadas para cada idioma específico, esto resulta una limitación ya que se requieren distintos modelos entrenados específicamente sobre cada lenguaje que retornen estas representaciones. Una forma de acatar esta necesidad podría ser utilizando por ejemplo un modelo multi-lenguaje como BERT. También se incluye que el agregado de las características léxicas mejoró mínimamente los resultados, en un 5%.

W. Zhu et al. [22], buscan generar una representación vectorial de una oración procesando secuencias de tokens. Para esto, similar a los trabajos anteriores, emplean redes siamesas pero proponen el agregado de un complemento: un módulo para las dependencias. En este módulo se emplea un parser que identifica aquellos elementos importantes en una oración, por ejemplo sujeto, objeto o predicado, agregándoles importancia dentro del modelo. Experimentan variando el peso que tiene este módulo en la totalidad de la red sobre el conjunto de datos SICK[23]. Con esto concluyen que el agregado del componente multiplicado por un factor de peso de 0.5, es aquel que da los resultados óptimos.

2.2.4 Respuesta a preguntas

Los sistemas de respuesta a preguntas o *question answering* tienen la tarea de contestar las preguntas de los usuarios, como menciona Jurafsky [6], la mayoría de estos sistemas se centran en preguntas factuales, preguntas que pueden ser brevemente contestadas. Ejemplos de estas preguntas son:

- ¿Quién fue el primer presidente de la República Oriental del Uruguay?
- ¿Cuántos lados iguales tiene un triángulo equilátero?

Jurasfky describe distintos paradigmas sobre este tipo de preguntas, solo uno será explicado ya que es el utilizado en el trabajo que se está presentando. Este paradigma es conocido como "basado en recuperación de información(RI)" o *"Information-retrieval-based" (IR)* y la tarea específica en la que trabajaremos es recuperación *ad hoc*. Consiste en retornar un conjunto ordenado de posibles resultados para la pregunta del usuario, estos posibles resultados pertenecen a una colección específica de información. Este concepto está más profundizado en la sección 2.2.5.

Sameera A. Abdul-Kader et al. [25] presentan un trabajo dedicado a diseñar un chatbot alimentado con información de la web. Lo interesante de la investigación es la variedad de métodos que se utilizan para la extracción de características. Presentan la combinación de éstos como un nuevo método para cuantificar las respuestas del chatbot. Entre los métodos utilizados se destacan: selección de características, N-Match, POS tagging y similitud semántica. Para evaluar el sistema se utiliza el Mean Reciprocal Rank que consiste en ponderar una cantidad de preguntas determinada según el ranking de la respuesta correcta.

Girish Kumar et al. [26] plantean el problema de la cantidad exagerada de preguntas que se le hacen a un vendedor sobre sus productos publicados en un mercado en línea. Comentan que gran parte de las preguntas pueden ser respondidas utilizando la descripción del producto. Para atacar este problema buscan extender un modelo presentado por Matthew Henderson et al. [27] en el cual se utiliza el producto vectorial de dos redes neuronales recurrentes, una para codificar pregunta y otra para generar la respuesta. Este trabajo propone agregar redes LSTM y mecanismos de atención al modelo mencionado.

Carlos Segura et al. [29] presentan un trabajo basado en la construcción de un chatbot que conteste preguntas en español sobre "La Liga", la liga española de fútbol. Utilizan una arquitectura simple basada en el modelo Rasa NLU, útil para procesar y buscar la información necesaria. Rasa NLU utiliza distintos mecanismos y técnicas de procesamiento explicadas anteriormente sobre la pregunta, para luego buscar una posible respuesta en Wikipedia. Para la búsqueda se utiliza SparQL, un lenguaje de consulta. Si no se encuentra una respuesta válida, busca en un corpus de conversaciones de OpenSubtitles para darle una respuesta amigable al usuario.

2.2.5 Recuperación de información

En el procesamiento de lenguaje, la recuperación de información implica encontrar información usualmente en documentos no estructurados dentro de un conjunto dado que satisfagan la necesidad de información inicial. El sistema que realiza el proceso de recuperación no contesta una pregunta específica, sino que ayuda a encontrar documentos que pueden contener la respuesta. Se le llama documentos relevantes a aquellos que contienen información que satisface la necesidad inicial del usuario y un sistema de recuperación es considerado perfecto si solo facilita documentos relevantes.[30][31]

Un modelo de RI, contiene:

- D - una forma de representar los documentos
- C - una forma de representar las consultas
- $R(c,d_i)$ - una función de similitud o ranking que ordena los documentos con respecto a la consulta.

De entre las características fundamentales a tener en cuenta para diseñar un sistema de recuperación de información, quizá la más importante es la forma en la que se recuperan los documentos. El índice invertido es una de las estructuras para recuperación de datos más conocidas. En esta estructura se tiene una colección que contiene cada palabra del sistema junto con los documentos que la contienen y su frecuencia. Un método muy común utiliza las palabras ingresadas por el usuario en la consulta y busca documentos conteniendo esas mismas palabras. Esto es conocido como el paradigma de búsqueda de *keywords* que explota la estructura de índice invertido mencionada.

Otra característica importante a tener en cuenta es la forma en la que se manejan las *stopwords*. Estas son aquellas palabras que aparecen frecuentemente en los textos pero su valor semántico es muy bajo. Por ejemplo preposiciones **en**, **por**, o también artículos **un**, **las**. Usualmente, la forma de manejar estos términos es eliminándolos al momento de búsqueda e indexado. Hay que prestar atención para no eliminar términos que podrían resultar relevantes aunque fuesen considerados *stopwords*.

También se considera importante mencionar el método conocido como *stemming*. Este es el proceso de reducir una palabra a su raíz (*stem*), eliminando sufijos o prefijos, como por ejemplo estudios o estudiando, se podrían reducir a estudi. Este método resulta de utilidad para generalizar un poco más y quitar peso de lo específico que puede ser un género o un número en una palabra.

Aparte de la búsqueda de *keywords*, otro método muy conocido en el campo de RI y que se usa hoy en día por su éxito al evaluar relevancia de términos es la ponderación estadística de *term frequency/inverse document frequency,* o TF-IDF. *Term frequency* es la suma de todas las ocurrencias de un término en un documento. *Inverse document frequency* es inversamente proporcional al número de documentos en los que aparece un término, entonces, a menor cantidad de documentos en los que encontrar el término, mayor es este factor. El peso de un término en un documento se mide como el producto entre su TF y su IDF. Esto permite asignar importancia a las palabras de una consulta. Y ha probado ser útil.

Pero si se utilizara únicamente estadística de frecuencias en la RI, nos encontraríamos con varios problemas, por ejemplo la polisemia o sinonimia. La polisemia es un fenómeno del lenguaje en el cual una palabra puede tener varios significados dependiendo de su contexto. Por ejemplo para

la pregunta **¿Dónde está mi gato?**, la palabra **gato** puede hacer referencia a un animal felino o a una herramienta de elevación de cargas. Por otro lado, la sinonimia es cuando varias palabras tienen un mismo significado. Por ejemplo **casa**, **vivienda**, u **hogar**, entre otras.

Es visualizable el carácter semántico de estos problemas que para un humano resultan fácilmente resolubles, pero trasladados al entorno computacional se convierten en un problema. Este es un campo actualmente estudiado y trabajado por la comunidad de PLN.

- Lucene

Existen varias herramientas empleadas para la RI, una de ellas es Lucene [32], una biblioteca de búsqueda *open source* que se enfoca en obtener resultados relevantes así como en performance. Provee una interfaz para resolver tareas de indexado, consulta, resaltado y análisis de lenguaje.

Una de las responsabilidades de la herramienta es la de recibir contenido en forma de documento o consulta y convertirlo en una representación interna apropiada para su uso. Los tokens obtenidos de un documento al momento de indexado o de una consulta al momento de búsqueda, se procesan en tres fases:

- Filtrado de caracteres y normalización (eliminación de tildes por ejemplo)
- Tokenización
- Filtrado de tokens (*stemming*, *lemmatization*, remoción de *stopwords*, etc.)

A grandes rasgos, los documentos se modelan como una lista ordenada de campos con contenido, de los cuales algunos se indexan. El texto plano es procesado en tokens mediante las tres fases mencionadas y se recaban atributos como valor, posición, *offsets*, etc. Luego todo esto es procesado por la herramienta de indexado, agregando los tokens a las listas correspondientes para luego aplicar compresión delta. Lucene provee varios tipos de búsqueda, entre ellos *queries* de términos, *queries* booleanas AND, OR y NOT, *queries* basadas en posición, términos difusos, expresiones regulares y más.

Durante la evaluación de las consultas, la función de similitud toma en cuenta estadísticas por término así como globales. Ejemplos son distancia entre términos de una frase, cantidad de términos emparejados, distancia de Levenshtein[1] para términos difusos, frecuencia de términos total, conteo de términos únicos y más.

Esta biblioteca ha sido elogiada y adoptada por la comunidad y se han construido herramientas sobre sobre ella, de entre las que se destaca Elasticsearch.

- Elasticsearch

Empleado hoy en día por empresas como Netflix[2], Uber[3], Slack[4], o Microsoft[5]. Elasticsearch[6] es un motor de análisis distribuido, escalable y con búsquedas en tiempo real. Provee un sistema distribuido donde se facilita una REST API JSON mediante la cual es posible acceder a las funcionalidades que ofrece Lucene.[35]

Actualmente es una herramienta reconocida por la comunidad para la tarea de recuperación de información. Existen trabajos como el de Brian D. Sloan[33], quien intenta mejorar la información recuperada por Elasticsearch modificando la consulta mediante algoritmos de expansión y fortalecimiento, orientado a complementar mediante componentes semánticos.

- *Boosting* o fortalecimiento, implica obtener etiquetas semánticas con las que se etiqueta a la consulta y los documentos relevantes. Si uno o más tags coinciden, se debe aumentar la relevancia del documento en cuestión.
- *Query expansión* o expansión, implica agregar etiquetas semánticas al texto de la consulta. Se debe seleccionar cuidadosamente qué peso asignar a cada concepto, ya que muy fácilmente se podría agregar ruido.

Aunque sus resultados no son del todo fuertes, sugiere que una combinación de las técnicas propiamente calibradas tomando en cuenta el corpus a usar, podría mejorar *precision* y *recall* de la recuperación. Menciona que se deben expandir los estudios y experimentación para declarar un método de búsqueda mejor que otro o avanzar propiamente en el estado del arte de la búsqueda semántica.

[1]Distancia de Levenshtein es la cantidad de operaciones necesarias para transformar una cadena de caracteres en otra. Las operaciones pueden ser inserción, eliminación o sustitución.
[2]https://www.netflix.com
[3]https://www.uber.com/
[4]https://slack.com/
[5]https://www.microsoft.com/
[6]https://www.elastic.co/

3. Análisis del Problema

Este capítulo incluye una descripción y análisis más profundo del problema presentado incorporando ejemplos de materiales aportados por el usuario responsable, un análisis de la estructura de su página web[1] y una propuesta de solución a alto nivel.

3.1 Descripción del Problema

El problema a describir se basa en que el acceso a información encontrada en la página de INEFOP no es simple. En una de las reuniones que fueron llevadas a cabo se nos informó que con la cuarentena por COVID-19 hubo un aumento del 500% de personas accediendo a los servicios. Esto hizo que los operarios se vieran abrumados con la cantidad de consultas que les llegaban diariamente. El tiempo requerido para informar a todas estas personas se incrementó, desbordando a los trabajadores y reduciendo la calidad del servicio ofrecido.

Inicialmente se discute una solución potencialmente accesible tanto para usuarios que ingresan de la web en busca de información como operarios de INEFOP encargados de contestar consultas. Pero el hecho de que esto es un proyecto de grado universitario con un tiempo acotado de implementación, llevó a restringir la solución para que sea únicamente accesible por operarios en una primera versión del prototipo.

3.2 Material aportado por INEFOP

A continuación, se detallan los documentos que nos fueron provistos en las primeras etapas del proceso. Estos nos permitieron comprender un poco más el problema, tanto del lado del usuario que consulta, como de la información que se encontraba disponible en la web y su estructura.

3.2.1 Bases Generales

Uno de los documentos facilitados son las *Bases Generales*, orientadas a informar a Entidades de capacitación y/o consultores que estén interesados en ofrecer cursos de capacitación o asistencia técnica. De 56 hojas de largo y en 14 capítulos, se detalla el proceso inicial de registro necesario para entidades de capacitación, documentos a presentar, plazos e ítems importantes a la hora de presentar una oferta, la forma en la que se valoran y seleccionan las ofertas, desde contenido y materiales hasta metodología, docentes, o la forma en la que se evalúa. También incluye temas contractuales y obligaciones legales.

[1] http://www.inefop.org.uy/

El usuario responsable realiza un gran hincapié en la importancia de este documento al comienzo del proceso. Dada la formalidad con la que está escrito y su gran tamaño, muchas personas pasan de leerlo para realizar consultas cuya respuesta se encuentra en este documento a través de los distintos medios. Inicialmente se discute como posible objetivo que el sistema a desarrollar pueda contestar preguntas de este documento, pero esto no se termina llevando a cabo debido a falta de tiempo.

3.2.2 Consultas

También se nos facilitan tres plantillas con 3118 ejemplos de consultas realizadas por usuarios reales. Estas resultan útiles para obtener la percepción interna de cómo luce y qué pregunta el usuario real.

Consultas sistema gestión INEFOP es la primera plantilla y contiene más de 1600 líneas que constan de una consulta (C), la respuesta retornada por el operario (R), y una clasificación del tipo de consulta (T).

Ejemplo 3.1

> **C)** Necesito saber donde ver y encontrar las postulaciones.
> **R)** La ECA puede ver sus propias postulaciones en su usuario de Gestión INEFOP.
> **T)** Evaluación y Monitoreo - Postulaciones a Llamados

Ejemplo 3.2

> **C)** Buenos días. Queríamos saber cuando quedaría acreditada la OC XXXX-2019 y XXXX-2019 del curso Nº 22215-2019. Desde ya muchas gracias. Saludos.
> **R)** El 20 de mayo.
> **T)** Administración y Finanzas – Facturación y pagos

Ejemplo 3.3

> **C)** Hola, quisiera informarme de qué hacer y cómo hacer para inscribirme en inefop como una ECA y brindar cursos de Marketing Digital.
> **R)** Comentario:
> Estimado la inscripción como ECA en inefop es autoadministrado en la web https://gestion.inefop.org.uy/eca/registro_eca/
> Cualquier consulta registro@inefop.org.uy Por la temática aconsejo contactarse y postular a través del PROGRETEC que es un programa conjunto de INEFOP y la CNCS para esta temática http://www.progretec.cncs.com.uy/ Saludos
> **T)** EMPRESAS-Asistencias Técnicas Consultorías

Aquí se ven varios casos de consultas, algunos resultan más impersonales como 3.1 o 3.3. Estas se basan en conocimiento general sobre procedimientos o información relevante al público. A primera vista, las preguntas impersonales parecerían ser más accesibles para que una herramienta automática responda. Las clasificaciones también podrían ser de utilidad para clusterizar preguntas, pero esto requeriría de un estudio más profundo del área y las posibilidades que ofrece.

Consultas Tarea Gestión INEFOP contiene alrededor de 1500 consultas realizadas, sin su respuesta pero con un asunto (A) que asumimos está escrito por el usuario. También se encuentran etiquetadas con un tag, de la misma forma que las preguntas anteriores. Ejemplos de esta planilla se presentan a continuación.

Ejemplo 3.4

C) Hola soy monotributo social mides me gustaría alguna capacitación para llevar a delante mi pequeña empresa
T) EMPRESAS-Capacitación
A) Cursos para pequeñas empresas

Ejemplo 3.5

C) HOLA BUENOS DIAS. solo quiero hacer una consulta soy de XXX y tengo mi padre aca sin trabajo, trabajo por muchos años hoteleria en mi pais el tiene 54 años y ¿QUIERO SABER SI EL PUEDE TENER OPORTUNIDADES DE HACER UN CURSO ACÁ POR INEFOP ? yo hize un curso de cocina por inefop me encanto. GRACIAS
T) Evaluación y Monitoreo - Postulaciones a Llamados
A) CONSULTAS

Ejemplo 3.6

C) Buen día, realice el registro completando el formulario de solicitud de ser una Entidad Capacitadora. Les consulto como sigue el tramite?, Gracias
T) Evaluación y Monitoreo - Registro de ECAs
A) Inscripción ECA

Este tipo de consultas representan desafíos en varios niveles. Primero, al estar realizadas por un usuario no técnico, éste emplea formalidades indispensables en el trato social que para un modelo que procesa lenguaje natural no es más que ruido. También hay faltas de ortografía o información que no aporta contenido real al objetivo. Estos parámetros deben tomarse en cuenta si se desea diseñar un sistema automático que conteste preguntas de este tipo.

Idealmente, la pregunta ingresada debería plantear una información precisa, no debería incluir formalidades como por ejemplo “Hola, que tal?” ni debería contener faltas de ortografía. Estas son algunas de las nociones ideales para trabajar con el procesamiento de consultas. Cuando las preguntas son planteadas bajo estos fundamentos, es más simple identificar la intención o necesidad del texto correctamente. En nuestro caso no era posible asegurar que se cumplieran estos fundamentos, ya que las preguntas serán hechas por usuarios sin conocimiento técnico.

Otra manera de ayudar a identificar la necesidad de un texto podría incluir un posible etiquetado del problema. Quien ingresa su consulta debe seleccionar un tag de entre un conjunto predeterminado, similar a lo que se presentó en los ejemplos anteriores. De esta forma se limita el espacio al que pertenece el cuestionamiento y por ende el espacio de respuestas se ve reducido. Esta idea se discute pero no se desarrolla debido a que requiere tener vasto conocimiento sobre todas las áreas de consulta posibles.

El último documento de consultas, llamado *Consultorio,* contiene 15 preguntas contestadas por operarios, con sus tags correspondientes. Estas preguntas son un poco más concisas. Se presentan ejemplos a continuación.

Ejemplo 3.7

C) Existen apoyos económicos para contratar asistencias técnicas, consultorías y acompañamiento de técnicos para las empresas?
T) Empresa - Capacitaciones
R) Puede consultar las herramientas disponibles en el siguiente link http://www.inefop.org.uy/iconos/Instrumentos-para-fortalecer-a-su-Empresa-con-Capacitacion-o-asistencia-tecnica-uc3258

Ejemplo 3.8

C) Apoyo para el desarrollo de productos y nueva propuesta de negocio
T) Empresa - Impositiva
R) Se recomienda los subsidios de Asistencia técnica http://www.inefop.org.uy/iconos/Instrumentos-para-fortalecer-a-su-Empresa-con-Capacitacion-o-asistencia-tecnica-uc3258
Dependiendo de la localidad acercarse a los Centros de Competitiviidad centros.uy
Y si es un modelo de negocio o producto innovador se le comenta de la web uruguayemprendedor con algunos links específicios (CANVAS, ANII, ANDE semilla, etc)

Ejemplo 3.9

C) Quisiera saber si disponen o saben donde puedo conseguir un documento que detalle los salarios que se pagan en el uruguay por tipo de empresa (chica, mediana, grande) y por cargo/trabajo.
T) Empresa - Otros
R) Puede encontrar en el mtss en los consejos de salarios https://www.gub.uy/ministerio-trabajo-seguridad-social/tematica/consejos-salarios-negociacion-colectiva
Otra opción es Uruguay XXI que tiene algunas referencias de sueldos de empresas enfocadas en asesorarar a inversores. https://www.uruguayxxi.gub.uy/es/centro-informacion/

Tomar en cuenta algunas de estas preguntas a la hora de diseñar el sistema parecería ayudar a comprender qué enfoque le da un usuario a la herramienta de consultas y si el mismo debiera ser restringido de alguna forma o guiado en caso de estar interactuando con el chatbot en vez de un operario. Luego de analizar las preguntas y tomando en cuenta los fundamentos explicados a lo largo de esta sección, se toma la decisión de exponer el sistema únicamente a operarios de INEFOP, dejando a los usuarios que realizan consultas por fuera. A los operarios de INEFOP se les puede dar instrucciones o proveer asesoramiento sobre cómo hacer uso de la herramienta.

3.2.3 Preguntas Frecuentes

También se nos facilita un documento de FAQ donde se detallan más de 30 preguntas con su respuesta. Este documento contiene las preguntas más frecuentemente realizadas por los usuarios, y son empleados por los operarios como plantillas. En los siguientes recuadros se presentan ejemplos.

Ejemplo 3.10

C) Mi empresa me mandó al seguro de desempleo, ¿Cuáles son los requisitos para acceder a un curso por INEFOP?
R) Participar de una entrevista con un Técnico de INEFOP, la cual debe ser agendada durante el período vigente de seguro.

Ejemplo 3.11

C) Si tengo claro el curso que quiero realizar, ¿igualmente debo participar de una entrevista?
R) Si, dado que durante la entrevista -en conjunto con el Técnico Orientador se analizará cuál de los cursos disponibles es el más adecuado a su perfil, experiencia y expectativas.

Ejemplo 3.12

> **C)** Quedé en lista de espera, pero nunca me llamaron, ¿qué debo hacer?
> **R)** Puede presentar un reclamo por medio de la página web: (Sugerencias y Reclamos) o en Recepción de sede central.

Estas preguntas consideradas frecuentes, resultan valiosas a la hora de diseñar una herramienta que cumpla con la premisa. Visualizamos como posible solución desarrollar un sistema capaz de contestar estas preguntas, convirtiéndo en uno de nuestros objetivos.

3.2.4 Estructura de la Web

Se investiga la actual estructura de la web (www.inefop.org.uy) para comprender un poco con qué panorama se encuentra un usuario que ingresa. Se diseña un diagrama[34] donde se despliegan los accesos a los distintos menús, emparejando en color aquellos accesos a los mismos lugares. También se incluye información relevante a cada ítem, como si la página contiene documentos, formularios o simples menciones relacionadas al contenido.

Nos encontramos con una página no demasiado amigable, con una estructura no clara, accesos repetidos e información considerada irrelevante para alguien interesado en acceder a un curso por ejemplo. Este sentimiento resultó en general compartido con aquellas personas o allegados nuestros que conocían la página y habían hecho uso de esta en algún momento.

3.3 Solución a Alto Nivel

Para la solución se propone inicialmente que el sistema generado se asemeje a un *chatbot,* una aplicación de software que simula una conversación de chat con una persona real. Este planteo genera dudas como: el chatbot mismo debería contestar preguntas en sí o redireccionar al usuario a una página dentro del dominio que contenga información relacionada. Para esto ayudaría tener las páginas categorizadas por ejemplo. Otros planteos incluían la decisión de si el chatbot podría llegar a facilitar un documento al usuario o permitirle llenar un formulario.

Como segunda opción, se discute la posibilidad de un motor de búsqueda, el cual podría ubicarse en la página o en un lugar independiente. Una ventaja de esta opción es que evita la necesidad de diseñar un diagrama de flujo complejo como el de un chatbot. Cualquiera de ellas parecía ser una solución viable para facilitar el trabajo de los operarios que contestan dudas, asistiendo en la búsqueda de información requerida por el usuario que pregunta. E incluso, si su funcionamiento

fuera muy bueno, con más trabajo podría extenderse para el público general, ayudando a cualquiera a encontrar la información buscada.

De entre los documentos presentados previamente se destaca el que contiene preguntas frecuentes. El hecho de que la web contiene el 80% de la información, contribuye a tomar la decisión de realizar una división parcial del problema en dos partes; una primera que pueda contestar preguntas frecuentes y la otra que busque respuestas en la web. Estas ideas nos resultan acordes al objetivo académico ya que tocan áreas que actualmente se encuentran en proceso de desarrollo e investigación y se alinean con aprendizajes académicos obtenidos anteriormente. A continuación se presentan los problemas ya descritos en el marco teórico relacionándolos con nuestros objetivos.

- ❏ Question Similarity

Este problema consiste en definir si es posible considerar un par de preguntas con diferente sintaxis como semánticamente equivalentes. Se utilizan como base las preguntas encontradas en el documento de FAQ. A pesar de ser solamente 30 preguntas, el usuario responsable asegura que puede continuar facilitando material similar.

- ❏ Question-Answering

En este problema se tiene una pregunta y un párrafo de texto, el objetivo es devolver un extracto del párrafo que contenga la respuesta a la pregunta. Para ello se puede tomar toda la página de INEFOP como base, descartando aquellos documentos demasiado extensos. Luego de analizar el alcance del proyecto, decidimos no llevar a cabo el enfoque de forma automática. Entrenar un modelo neuronal requiere una gran cantidad de ejemplos que debíamos fabricar sin conocimiento.

El sistema final consta de un motor de búsqueda para consultas de usuario que incluye una recuperación de preguntas similares mediante métodos automáticos y representaciones vectoriales. Para la parte de respuestas a preguntas, se decide aproximar una respuesta como un link dentro de la totalidad de la web de INEFOP apoyados por una herramienta muy potente de recuperación de información.

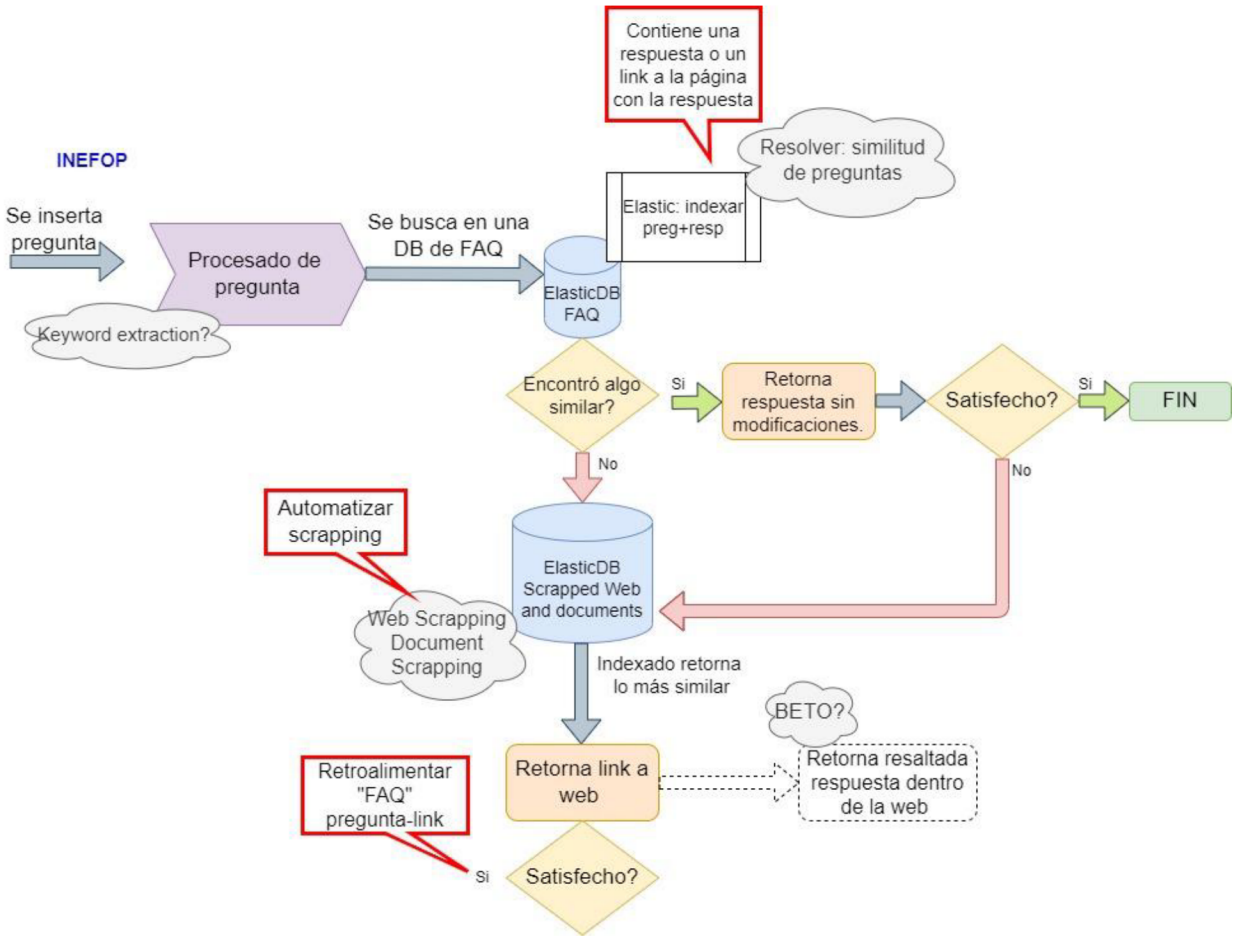
3.3.1 Diagrama de Flujo

Habiendo analizado el problema, se presenta una primera versión de la idea en un diagrama de flujo en el Cuadro 3.1 que describe al sistema. Algunos de los comentarios incluidos son ideas a tener en cuenta para un trabajo futuro.

- ❏ El proceso completo comienza con la inserción de una pregunta a procesar. Se comienza aplicando remoción de *stopwords*. Para la extracción de palabras existen diferentes estrategias.

Existen modelos automáticos pero consideramos que no valía la pena emplear uno. El sistema desarrollado filtra basándose en el etiquetado POS, se permiten únicamente sustantivos, adjetivos y verbos.

- Luego se busca esta pregunta en una colección de preguntas almacenadas dentro de Elasticsearch. Aquí se debe resolver el problema de similitud entre preguntas, ya que si Elasticsearch recupera más de una opción, se debe entender cuál es la más similar y por ende la que potencialmente contenga la respuesta a la pregunta ingresada. En el diagrama se menciona que las respuestas no tienen por qué tener formato oración, sino que podrían ser un link que contenga la información.



Cuadro 3.1: Diagrama de flujo del motor de búsqueda que contempla ambos problemas, similitud de consultas y respuestas a preguntas. No todas las ideas presentadas dentro de los globos grises o recuadros de diálogo rojos se llevan a cabo. La extracción de keywords no es automática, el scraping en el servidor no se automatiza y no se hace uso del modelo multilenguaje BERT. En general debido al alcance acotado.

- Si se encontrase algo similar, se retorna la respuesta al usuario, y este pasa a decidir si tal respuesta satisface su duda. En caso de que esta sea satisfactoria se finaliza. En caso de que no sea satisfactoria o de que no se haya encontrado una pregunta similar a la pregunta ingresada, se pasa a realizar una búsqueda dentro de la colección de webs y documentos relevados mediante un *scraping* previo. Hay varios detalles a considerar sobre cómo se realiza el scraping de la web, por ejemplo verificar que se hayan almacenado las páginas indicadas en formato correcto, sin ruido y no repetidas. Otro punto a tener en cuenta es la automatización del proceso de *scraping*, pues la página se actualiza regularmente. En una situación ideal, cada vez que se actualice la página de INEFOP se debería aplicar el proceso de *scraping* y guardar lo relevado para estar al día en Elasticsearch.

- Una vez recuperadas las páginas cuya información se relaciona con la consulta, se retornan los links. Una opción ya mencionada descartada debido al alcance del proyecto pero que se puede tener en cuenta para trabajo futuro es aplicar Q&A dentro de estos links.

- Por último, un paso de retroalimentación o feedback es necesario al menos para las primeras etapas del proyecto. Obteniendo así ejemplos sobre los cuales iterar en el proceso de mejora del sistema.

4. Detección de preguntas similares

Este capítulo incluye una descripción del proceso seguido al detectar preguntas similares. Se presentan los dos enfoques que fueron abordados. Dentro de cada uno se describen los modelos empleados, recursos utilizados y resultados intermedios y finales.

El segundo enfoque, surge de los resultados no satisfactorios del primero. Se cambia radicalmente la idea y se obtienen resultados casi perfectos en un tiempo de desarrollo mínimo.

4.1 Detección mediante redes siamesas

Se comienza tratando de predecir similitud sobre dos strings haciendo uso de redes siamesas tal y como se lee en los trabajos descritos en la sección 2.2.3. Para ello se decide trabajar con *embeddings*, transformando las listas de tokens a listas de vectores y utilizándolos como entrada para las redes. Se espera tener como salida un valor de similitud.

Embeddings

La obtención de los *embedding* se logra mediante un modelo[41] que tokeniza y convierte la tira de palabras a una tira de vectores de **768** dimensiones. Este modelo es entrenado por los ingenieros en Google empleando el corpus de la wikipedia en español con un diccionario total de 32.000 palabras, tiene una arquitectura de tipo Transformer XL[42], una de las más actuales. También retorna otros campos como activaciones en cada capa, *negative log likelihood*[1] del texto y lista de tokens pero no se hace uso de ellos.

Previo a la inserción de las listas de vectores en las redes, se debe considerar que estas requieren que sus entradas siempre tengan un mismo tamaño. Una oración puede tener una cantidad variable de palabras, por lo que se debe comenzar definiendo un número máximo de términos. A partir de eso, si la oración es más larga, se remueven algunos y si la oración es más corta, se agregan términos vacíos. A este último procedimiento se le conoce como *padding*. En nuestro caso, se coloca un máximo de **100** términos, y se agregan vectores nulos de 768 dimensiones hasta completar el máximo.

[1]Negative log likelihood: es un método de pérdida utilizado durante el entrenamiento de modelos neuronales. Representa la seguridad con la que el modelo retorna el output, a mayor certeza, los valores de esta función de costo son menores.

Relacionado a la descripción del modelo, a pesar de que se prueban más de 10 modelos distintos variando parámetros como capas, arquitectura, algoritmos de aprendizaje y cantidad de iteraciones de entrenamiento, se realiza únicamente la descripción del modelo elegido en el Cuadro 4.1. Como el proceso de elección de hiperparámetros es un proceso que usualmente es de prueba y error, se decide minimizar su descripción incluyendo solamente algunos resultados intermedios.

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 100, 768)]	0	
input_2 (InputLayer)	[(None, 100, 768)]	0	
lstm (LSTM)	(None, 100, 100)	347600	input_1[0][0] input_2[0][0]
lstm_1 (LSTM)	(None, 100, 128)	117248	lstm[0][0] lstm[1][0]
lstm_2 (LSTM)	(None, 64)	49408	lstm_1[0][0] lstm_1[1][0]
lambda (Lambda)	(None, 1)	0	lstm_2[0][0] lstm_2[1][0]

Total params: 514,256
Trainable params: 514,256
Non-trainable params: 0

Cuadro 4.1: Arquitectura de la red empleada. Se trata de dos redes idénticas cuyo input inicial consta de una lista de 100 elementos, siendo cada uno un vector de 768 dimensiones. Sigue con 3 capas de 100, 128 y 64 unidades LSTM. Finaliza con una función lambda.

El uso de unidades LSTM o *Long Short-Term Memory* se elige debido a que es un tipo especial de red recurrente donde la información persiste y se toma en cuenta instancias de tiempo anteriores hasta por más de 1000 instancias[43]. Un problema contemplado en esta etapa fue la elección de la función de activación de las unidades LSTM. Se comienza eligiendo función de activación ReLU o *Rectified Linear Activation Function*. Al recibir un valor negativo esta función retorna 0, mientras que para valores positivos retorna el valor. Este método es muy potente ante problemas como el desvanecimiento de gradiente. Esto ocurre cuando las funciones de activación producen valores muy cercanos a 0, por lo que el aprendizaje durante el *backpropagation* se torna muy lento o se estanca. Para nuestro set de datos y problema específico este método de activación terminó matando el aprendizaje casi instantáneamente convergiendo sin aprender nada. Esto es conocido como el *dying ReLU problem* y es recurrente en redes con muchas capas o, como nuestro caso, en RNNs donde se emplean gradientes de capas más profundas y de pasos de tiempo previos. Finalmente se opta por la función de activación *tanh*,

donde se confirma el aprendizaje en la mejora del *accuracy* a medida que pasan las iteraciones de aprendizaje.

Una vez finalizado el procesamiento en la primera capa, se retorna la secuencia final de salida de cada una de las unidades para usarse como input en la siguiente capa LSTM. Este proceso se repite hasta llegar la última capa donde se obtienen ambos vectores de salida y se comparan mediante una función lambda que retorna la distancia entre estos. Para la función lambda se prueban dos funciones de distancia: Euclídea y Manhattan.

La distancia Euclídea es muy utilizada al calcular la distancia entre puntos en un espacio vectorial. El cálculo de esta distancia se realiza como la raíz cuadrada de la suma del cuadrado de resta coordenada a coordenada:

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (y_i - x_i)^2}$$

Esta distancia no resultó útil, ya que no contribuyó al entrenamiento del modelo como se puede apreciar en la primera gráfica del Cuadro 4.2.

La distancia Manhattan o geometría del taxi, a diferencia de la euclidiana, es más útil para describir objetos en una grilla uniforme. La idea intuitiva es el cálculo de la distancia más corta que tomaría un taxi para llegar de un punto a otro en un mapa con manzanas. Se calcula como la suma del valor absoluto de la diferencia coordenada a coordenada:

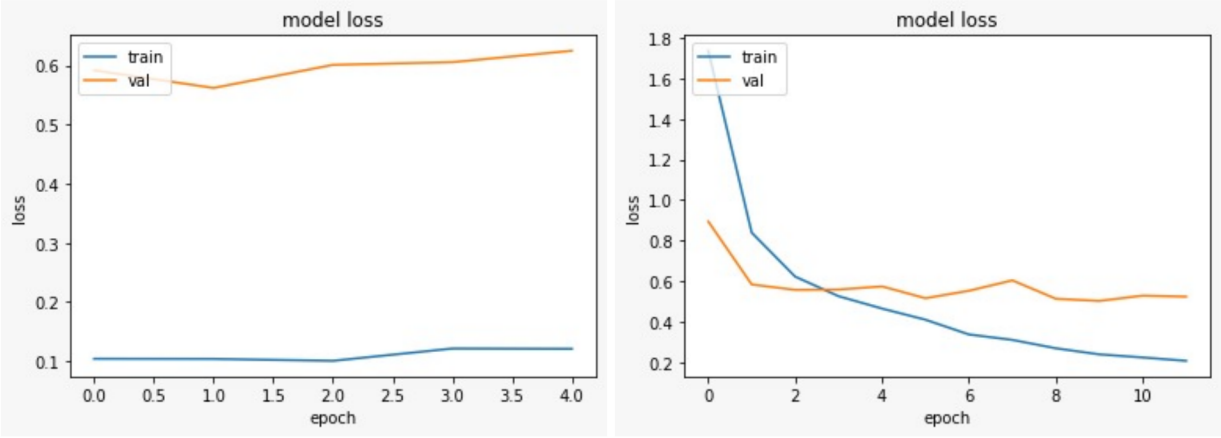
$$d(x, y) = \sum_{i=1}^{n} |y_i - x_i|$$

Esta función de distancia parece contribuir al aprendizaje del modelo ya que a mayor número de *epochs* el *loss* disminuye, esto se ve en la segunda gráfica del Cuadro 4.2.

La fluctuación del valor de pérdida entre la primera y la última iteración de pérdida es visible en la tabla del Cuadro 4.3. Esto da a entender que el modelo aprende o no dependiendo del uso de una distancia o la otra. Aunque los motivos para que funcione con una sola de las distancia son contraintuitivos y no claros, se selecciona la distancia Manhattan por ser la que funciona.

Al compilar el modelo se prueban varias funciones de costo o *loss function.* Estas computan qué tan buena es la predicción del modelo sobre las entradas. El objetivo del entrenamiento es encontrar el conjunto de *weights* que minimiza el *loss* promedio de todo el conjunto, se calcula al final de cada iteración y los pesos de cada capa se actualizan mediante *backpropagation*. Se prueban varias funciones, entre ellas *contrastive loss.* Esta función toma la salida de la red para un ejemplo positivo, calcula su distancia a otro ejemplo de la misma clase y contrasta con la distancia a un ejemplo negativo. Esto no resultó útil y el entrenamiento tuvo resultados erráticos. Una hipótesis de por qué esto ocurre es que en nuestro problema no hay clases definidas. No se puede generalizar una entrada dentro de un conjunto similar y no similar. Cada caso tiene

ejemplos específicos que son similares, pero no existe un conjunto global similar entre sí. Finalmente, se utiliza *binary cross entropy*, método que compara la probabilidad predicha con la clase real, 0 o 1, luego se calcula el score que penaliza la probabilidad basado en la distancia al valor esperado.



Cuadro 4.2: Comparación de las gráficas de *loss* del modelo haciendo uso de la distancia Euclídea (izquierda) o de la distancia Manhattan (derecha). *Loss* es la penalización aplicada por una mala predicción y cada epoch es una iteración completa del corpus de datos.

Distancia	**Fluctuación val_loss**
Manhattan	Desde 0.89 hasta 0.52
Euclidea	Desde 0.59 hasta 0.62

Cuadro 4.3: Fluctuación del valor de *loss* a medida que se realizan iteraciones para cada una de las distancias seleccionadas.

Por último se prueban dos *optimizers* para el modelo, *nadam* y *opt*. En este caso, *nadam*, también estanca el aprendizaje, mientras que *opt* contribuye.

Corpus

La obtención del corpus para el entrenamiento del modelo tuvo dos instancias. El corpus inicial de entrenamiento es el provisto para la competencia de SemEval[14]. Contiene 1555 pares de oraciones en español obtenidas de Newswire[1] y Wikipedia[2] junto con un valor de similitud entre 0 y 4. A continuación se presentan ejemplos.

[1]https://www.newswire.com/
[2]https://www.wikipedia.org/

Ejemplo 4.1

O1) El "Parlamento de Fiyi" ("Parliament of Fiji" en inglés), es bicameral y lo forman la Cámara de Representantes y el Senado de Fiyi.
O2) El Parlamento de Fiyi nace el 10 de octubre de 1970, cuando Fiyi consiguió la independencia del Reino Unido.
- Similitud semántica: 1.6

Ejemplo 4.2

O1) La ONU condena el ataque a una escuela de Nigeria y pide que se haga Justicia Naciones Unidas, 10 nov .
O2) El secretario general de la ONU, Ban Ki-moon, condenó hoy el atentado suicida perpetrado en una escuela en Nigeria, en el que murieron al menos 48 personas, y exigió que se lleve ante la Justicia a los responsables.
- Similitud semántica: 3

Dado que nuestro modelo clasifica entre similar o no similar, se decide traducir el valor de similitud semántica a:

no similar o **0** si x<3
similar o **1** si x>=3

Esto causa que el conjunto de datos quede desbalanceado con un 75% de ejemplos no similares y únicamente un 25% de ejemplos semánticamente equivalentes. Este conjunto de datos se termina descartando debido a que su formato no el problema de encontrar preguntas semánticamente equivalentes.

El hecho de que el problema presentado sea en español resulta una complicación a la hora de obtener material de entrenamiento, por lo que el corpus final se termina fabricando. Para esto se emplea el conjunto de pares de preguntas similares de Quora[3] en inglés. Este corpus tiene el mismo formato que nuestro problema, un par de preguntas con un valor de duplicado que varía entre 0 y 1. Este formato es visualizable en los siguientes recuadros.

Ejemplo 4.3

C1) What is the best travel website in spain?
C2) What is the best travel website?
- Similitud semántica: 0

[3]https://www.quora.com/

Ejemplo 4.4

> **C1)** How should I prepare for CA final law?
> **C2)** How one should know that he/she completely prepare for CA final exam?
> **-** Similitud semántica: 1

Se traducen 10.000 pares haciendo uso del traductor de Google. Se confirma que la traducción haya quedado acertada para ciertas preguntas al azar y se emplea ese corpus. Dado que no se cambia el modelo, se realiza una prueba preliminar para verificar que el modelo se mantiene útil para el nuevo corpus.

Para realizar esta prueba, se inicializa el modelo con valores aleatorios, se toman 5.000 ejemplos balanceados, 2500 pares similares y 2500 no similares. Cada par se corre por el *pipeline* ya presentado en la sección de Arquitectura en la página 28. Se emplean 4000 de esos ejemplos para entrenar, 500 para validar y 500 para *test*. Los resultados presentados en la tabla del Cuadro 4.4 muestran una clara mejoría de pérdida y precisión.

Formato	**loss**	**acc**
Sin entrenar	1.06	0.48
Entrenando con 4000 ejemplos	0.80	0.59

Cuadro 4.4: Resultados de prueba del modelo neuronal, se prueba con el mismo lote de pares de preguntas, sobre un modelo inicializado aleatoriamente y otro entrenado sobre instancias distintas.

Esto confirma que a priori el modelo consigue aprender sobre la tarea en cuestión y da para intuir que a mayor cantidad de ejemplos, los resultados deberían mejorar. Por lo que a continuación se toman 10.000 ejemplos y se entrena en batches, se toma el 80% del conjunto para entrenar, el 10% para validar y el 10% para test.

Entrenamiento y resultados

Para entrenar el modelo, se divide el corpus en baches de 1000 pares de preguntas y se entrena guardando el modelo intermedio. Esto dado que el tamaño excede la RAM disponible en nuestros equipos haciendo que el proceso de entrenamiento sufra un *crash* antes de terminar. Se evidencia como el poder de cómputo requerido para entrenar modelos que tomen decisiones sobre el lenguaje es considerable.

La salida del modelo es un número real entre 0 y 1 que representa la distancia semántica entre las dos oraciones. A mayor cercanía a 1, más similares son los inputs que se ingresan, mientras que cuanto más cercano a 0 implica no estar relacionados.

Se realizan pruebas sobre este sistema cuya conclusión no resulta alentadora. Estas se detallan en la sección de análisis experimental 4.3. Posteriormente, se incluye como soporte a la detección del modelo un filtrado previo con la herramienta Elasticsearch que mejora los resultados, pero aún así estos no son del todo satisfactorios.

Originalmente la idea de este método incluía que luego de la fase de instalación en el servidor, el cliente fabrique una cantidad considerable de ejemplos de prueba. Se desarrolló una funcionalidad de guardado en la nube, guardándose una tripla que contiene la pregunta ingresada, la devuelta por el servidor y si este par era satisfactorio o no, exactamente como el conjunto de entrenamiento. De esa forma, se esperaba juntar ejemplos suficientes como para crear un nuevo corpus específico del área, así pudiendo re-entrenar el modelo e idealmente mejorar los resultados. En la realidad, hubo un retraso considerable al momento de dejar la herramienta disponible y una vez se dejó disponible, esta no fue utilizada lo suficiente, por lo que la cantidad de ejemplos recabados no fue numerosa.

Debido a que el resultado final no resultó satisfactorio, se decide desarrollar otro enfoque donde se incrementa la simpleza y precisión de la solución final.

4.2 Detección mediante sentence embeddings

Después de leer un artículo[44] escrito por Julie Tibshirani, ingeniera de Elasticsearch, donde se presenta la noción de búsqueda en Elasticsearch por *embeddings* y luego de ver que la herramienta arroja en general resultados de recuperación de información muy precisos, surge la idea de aplicar búsqueda de similitud por *embeddings* en Elasticsearch pero empleando un único vector que represente la totalidad de la pregunta o un *sentence embedding.*

Sentence Embeddings

De la misma manera que los *embeddings*, los *sentence embeddings* son vectores de largo fijo. Luego de calcular un vector por palabra, se realiza un cómputo entre los vectores que componen la oración obteniendo otro vector que es su representación. Para su obtención, se hace uso del modelo desarrollado por Google universal-sentence-encoder-multilingual v3[40]. Este recibe una frase de largo variable y retorna un vector de 512 dimensiones. Existe una versión del modelo

más grande, pero dado que los recursos donde debe correr el sistema son limitados y el modelo chico parece funcionar acorde a la tarea, se opta por hacer uso del modelo de menor tamaño.

Búsqueda por vectores

Desde Elasticsearch 7.14, se introduce una nueva clase llamada *dense_vector*, que permite realizar búsquedas en dicha clase mediante scripts pre definidos. Se decide emplear similitud coseno, una de las opciones disponibles que es regularmente utilizada en recuperación de información. Allí se retornan las preguntas que tengan su *sentence embedding* más cercano a lo preguntado.

De esta forma se evita tratar con modelos, corpus, sesiones de entrenamiento, etc. Se hace uso de modelos ya entrenados por grandes compañías tecnológicas combinado con herramientas de recuperación de información populares en la actualidad, pudiendo así resolver el problema de obtención de preguntas similares semánticamente.

Los resultados que se describen en profundidad en la siguiente sección, alcanzan la mayor precisión de entre los dos enfoques para la tarea de similitud entre preguntas. Casi en la totalidad de ejemplos, siempre retornan una consulta relevante.

4.3 Comparación y resultados

A continuación se presenta el proceso de pruebas llevado a cabo para ambos enfoques de detección de similitud semántica entre preguntas. Se presentan resultados intermedios para las arquitecturas en cuestión y finalmente el resultado global.

Resultados intermedios para redes siamesas

Para mejorar el entrenamiento, se puede utilizar el recurso de entrenamiento sobre los mismos datos, o entrenamiento por una cantidad de *epochs*. Hay que encontrar el número indicado ya que este recurso es útil hasta cierta medida si se quiere reforzar el aprendizaje sin agregar nuevos datos, pero no se debe abusar pues podría resultar en sobreajuste. La tabla del Cuadro 4.5 muestra los resultados del entrenamiento sobre la cantidad de epochs, dejando en evidencia que 5 es el número de iteraciones que se deben realizar sobre el conjunto de entrenamiento. Se toma ese modelo para el desarrollo del servidor.

Epochs	**loss**	**acc**
1 epoch	0.74	0.57
2 epochs	0.73	0.60
5 epochs	**0.72**	**0.63**
10 epochs	0.79	0.59

Cuadro 4.5: Resultados de precisión y pérdida sobre las instancias de prueba luego de entrenar el modelo sobre los mismos datos una cantidad de iteraciones variable.

Resultados para similitud semántica entre preguntas

El conjunto de datos consta de preguntas semánticamente equivalentes específicas del dominio y es fabricado haciendo uso de los recursos provistos por el usuario responsable. Las preguntas originales se sacan del documento de FAQ. Este conjunto de datos se presentan en los ejemplos 4.5 a 4.8, incluyendo la pregunta original seguida de la fabricada.

Ejemplo 4.5

P1) Mi empresa me mandó al seguro de desempleo, ¿Cuáles son los requisitos para acceder a un curso por INEFOP?
P2) Estoy desempleado, que requisitos debo cumplir para acceder a un curso?

Ejemplo 4.6

P1) Estoy en el seguro de desempleo por el Covid-19, ¿tengo derecho a realizar un curso?
P2) Estoy desempleado debido al coronavirus, puedo realizar un curso?

Ejemplo 4.7

P1) ¿Debo asumir algún costo durante el desarrollo del curso?
P2) Hay costos asociados a los cursos?

Ejemplo 4.8

> **P1)** ¿Puedo hacer dos cursos a la vez?
> **P2)** ¿Puedo hacer varios cursos a la vez?

Para evaluar la correctitud del primer enfoque, se empareja cada una de las preguntas **fabricadas** contra las 35 preguntas **originales**. Estos 35 pares de preguntas, se pasan a través del *pipeline* que finaliza con el *output* del modelo, un número real entre 0 y 1. Este es el valor de similitud final para cada uno de los pares. Estos valores se ordenan y se toman las primeras tres preguntas con valor de similitud más elevado. Si la pregunta original correspondiente se encuentra entre estas tres preguntas se considera el intento acertado. El margen de 3 preguntas dado se basa en que es normal mostrarle al usuario más de una posible pregunta similar a la que ingresó, es una práctica usual hoy en día en la internet y al ser un prototipo universitario de soporte a dudas tampoco se ve obligado a responder a la perfección.

El porcentaje de acierto en esta primera pasada, es del **28%** de aciertos. 10 preguntas de 35 son correspondidas correctamente, en la tabla del Cuadro 4.6 se presentan algunos ejemplos.

Pregunta fabricada	**1º**	**2º**	**3º**
Como hago para agendar una entrevista?	**¿Qué pasos debo seguir para agendar una entrevista?**	¿Cómo me enteró si me aprobaron el voucher?	¿Cuánto duran los cursos?
Hay costos asociados a los cursos?	¿Cuántos cursos puedo hacer si estoy en el seguro?	**¿Debo asumir algún costo durante el desarrollo del curso?**	¿En qué Instituto/Entidad de capacitación puedo hacer el curso?
Los certificados se tienen que pagar?	¿Qué pasos debo seguir para agendar una entrevista?	¿Cómo, cuándo y dónde se entregan los certificados?	¿Cuánto duran los cursos?
Luego de hacer un curso, me dan trabajo?	Llamé al call center para agendarme pero no les figura la información actualizada de mi seguro, ¿qué debo hacer?	La institución me dijo que llame a INEFOP para pedir un voucher, ¿qué pasos debo seguir?	¿Los certificados tienen un costo para los usuarios?

Cuadro 4.6: Presentación de las primeras 3 preguntas ordenadas por valor de similitud, en negrita en las primeras dos filas se pueden ver las preguntas originales correctas.
*Para los casos de no acierto las preguntas originales eran:

- ¿Los certificados tienen un costo para los usuarios?
- ¿Si hago el curso, ya me dan un trabajo?

En este punto la precisión inicial tan baja genera sospecha de que el modelo no es suficiente para la tarea de similitud de preguntas, por lo que se prueba incluyendo a la arquitectura otro input en forma de tripla donde cada elemento corresponde a la medida de la distancia Damerau Levenshtein[1], Jaro Winkler[2] y Hamming[3] entre las entradas textuales. Esta tripla se agrega a la arquitectura, concatenando esta a la salida de las redes siamesas y se inserta en una capa densa. Esto parece aportar ruido ya que no ocurre aprendizaje aparente. Un motivo puede ser la diferencia de rango en los valores, de números enteros a números racionales entre 0 y 1, o quizás simplemente no sean medidas de mucha importancia para calcular la similitud semántica. Esto finalmente se descarta y se continúa con el modelo previo.

Aquí surge la idea de mejorar los resultados del modelo original a través del *fine-tuning*. Esto implica conseguir un conjunto de datos específicos del área, pares de preguntas sobre temas de relevancia para INEFOP con su indicativo de similitud y entrenar el modelo sobre estos datos. De esta manera se debería incrementar su precisión. Pero conseguir estos datos no es sencillo. Una posibilidad es escribir este corpus a mano pero el no conocimiento del área puede llegar a perjudicar el proceso. Dado que el tiempo requerido para escribir un conjunto de datos es considerable, se optó por descartar esta opción y apelar a la recolección mediante el uso que le diera el usuario responsable.

En las semanas posteriores a estos resultados, se comienza el desarrollo del back-end del servidor, lo que incluye la instalación de Elasticsearch así como el almacenamiento de los datos. Esto facilita un paso intermedio que no se había tenido en cuenta hasta ahora. Mediante Elasticsearch se pueden recuperar aquellas preguntas que probablemente sean relevantes. La herramienta retorna una lista de entradas ordenadas mediante un score siendo las de mayor score aquellas más “similares”. Esto abre la posibilidad a un filtrado inicial donde se toma únicamente las X entradas con mayor score y se pasa esas por el modelo. Se decide tomar todas las entradas devueltas por Elasticsearch, estas varían desde 2 hasta 6+ entradas.

---

[1]Damerau Levenshtein es la cantidad de operaciones necesarias para transformar una cadena de caracteres en otra. Siendo las operaciones inserción, eliminación, sustitución y transposición de caracteres.

[2]Jaro Winkler es una métrica de cadena que mide la distancia de edición entre dos secuencias pero otorgando calificaciones superiores a las cadenas que coinciden desde el principio.

[3]Hamming también es una distancia de edición que cuenta la cantidad de sustituciones requeridas para pasar de una cadena de caracteres a otra.

Lo devuelto por Elasticsearch es razonable, se ve una gran mejora en los resultados generales. Casi el **70%** de las preguntas retornadas son correctas. Esto es visible en el Cuadro 4.7. Se evidencia como la recuperación de información facilitada por Elasticsearch filtra las entradas alivianando el peso que se impone sobre el modelo al distinguir la similitud entre pares. Esto genera la duda de si la recuperación de Elasticsearch es suficientemente buena como para deshacernos de la red neuronal.

Pregunta fabricada	**1º**	**2º**	**3º**
Ya se que curso quiero, tengo que hacer la entrevista igual?	En la entrevista, quedé anotado en lista de espera, ¿cuánto tiempo tengo que esperar?	**Si tengo claro el curso que quiero realizar, ¿igualmente debo participar de una entrevista?**	-
Me pagan por hacer el curso?	¿Debo asumir algún costo durante el desarrollo del curso?	¿Si hago el curso, ya me dan un trabajo?	¿**INEFOP me paga por realizar el curso?**
Que institutos de capacitacion brindan los cursos?	¿En todos los cursos se entregan materiales?	¿Cuánto duran los cursos?	¿Puedo hacer dos cursos a la vez?
Si el curso es online, me pagan el internet?	¿Debo asumir algún costo durante el desarrollo del curso?	¿INEFOP me paga por realizar el curso?	¿INEFOP paga viáticos para poder hacer el curso?

Cuadro 4.7: Presentación de las primeras 3 preguntas ordenadas por valor de similitud, en negrita en las primeras dos filas se pueden ver las preguntas originales correctas.
*Para los casos de no acierto las preguntas originales eran:

- ¿En qué Instituto/Entidad de capacitación puedo hacer el curso?
- Si hago un curso online, ¿me pagan por el internet que necesite?

Luego de comprender que el conjunto de datos para *fine-tuning* no se obtendría ya que el uso que se le dio a la herramienta fue mínimo, se agotaron las ideas para la mejora del modelo. Empujados por la no conformidad con los resultados se cambia radicalmente la arquitectura. Aquí se introducen los *sentence embeddings*, se descarta el modelo neuronal y se efectúa una recuperación de información utilizando los vectores. Se llevan a cabo las mismas pruebas pero tomando los primeros tres resultados arrojados por Elasticsearch. La precisión final obtiene un **97%** de aciertos y se equivoca tan solo en 1 de las preguntas. Los resultados se presentan en el Cuadro 4.8.

Pregunta fabricada	1º	2º	3º
Que cursos hay disponibles?	**¿Qué cursos puedo hacer?**	-	-
En el call center me preguntan datos que no quiero brindar, puedo hacer los cursos igual?	**Llamé al call center pero me preguntan datos que no quiero brindar, ¿igual puedo hacer el curso?**	-	-
Como justifico el abandono de un curso?	¿Qué sucede si abandono el curso sin previo aviso o fuera del plazo estipulado para hacerlo?	Presenté el justificativo para abandonar el curso, ¿debo quedarme con una constancia de ello?	**¿Cuáles son las causas por las que puedo justificar el abandono del curso?**
Estoy desempleado debido al coronavirus, puedo realizar un curso?	¿Si hago el curso, ya me dan un trabajo?	¿Puedo hacer dos cursos a la vez?'	¿Qué debo hacer si mientras estoy haciendo el curso, consigo un trabajo que me coincide con el horario del curso?

Cuadro 4.8: Presentación de las primeras 3 preguntas ordenadas por valor de similitud, en negrita en las primeras dos filas se pueden ver las preguntas originales correctas.
*Para los casos de no acierto las preguntas originales eran:

- Estoy en el seguro de desempleo por el Covid-19, ¿tengo derecho a realizar un curso?

Estos resultados presentados en el Cuadro 4.8 evidencian como la combinación de recuperación de Elasticsearch sobre los *sentence embeddings* de las preguntas es la mejor de las opciones. Ambas herramientas, Elasticsearch y el modelo de *sentence embedding,* son desarrolladas por entidades particulares. Esto confirma como recursos desarrollados que ya se encuentran en el mercado resultan fiables en la búsqueda de soluciones. La adaptación de las herramientas a los problemas puntuales mediante un proceso creativo es clave para encontrar una solución óptima. Como última comparación, se presenta en el Cuadro 4.9 lo retornado para 4 preguntas en ambos enfoques.

Enfoque	Pregunta fabricada	1º	2º	3º
Siamesas	En el call center me preguntan datos que no quiero brindar, puedo hacer los cursos igual?	Llamé al call center para agendarme pero no les figura la información actualizada de mi seguro, ¿qué debo hacer?	**Llamé al call center pero me preguntan datos que no quiero brindar, ¿igual puedo hacer el curso?**	-
Sentence embeddings	En el call center me preguntan datos que no quiero brindar, puedo hacer los cursos igual?	**Llamé al call center pero me preguntan datos que no quiero brindar, ¿igual puedo hacer el curso?**	-	-
Siamesas	Si la entrevista es despues de que termine mi seguro, puedo hacer el curso igual?	Si tengo claro el curso que quiero realizar, ¿igualmente debo participar de una entrevista?	Estoy en el seguro de desempleo por el Covid-19, ¿tengo derecho a realizar un curso?	¿Qué pasos debo seguir para agendar una entrevista?
Sentence embeddings	Si la entrevista es despues de que termine mi seguro, puedo hacer el curso igual?	**Si la fecha de entrevista es posterior a la finalización de mi seguro, ¿igual puedo hacer el curso?**		
Siamesas	Que hago si el curso que me interesa lo brinda una institucion que no esta en la pagina?	¿Qué debo hacer si mientras estoy haciendo el curso, consigo un trabajo que me coincide con el horario del curso	¿INEFOP paga viáticos para poder hacer el curso?	¿INEFOP me paga por realizar el curso?
Sentence embeddings	Que hago si el curso que me interesa lo brinda una institucion que no esta en la pagina	**¿Qué debo hacer si me interesa hacer un curso pero la Institución no figura en el listado que figura en la web?**		

Cuadro 4.9: Comparación de una misma pregunta para cada enfoque, en negrita se pueden ver las preguntas originales correctas y los enfoques si las respuestas retornadas se consideran satisfactorias.

5. Recuperación de información en la web de INEFOP

Este capítulo describe el proceso seguido para responder preguntas con información obtenida de la página web de INEFOP. Se detalla como se obtuvieron todos los datos relevantes, el procesamiento necesario tanto de la web como el realizado a las preguntas que intentan obtener esta información y se presenta la evaluación realizada al modelo obtenido con los resultados de las diferentes variantes.

5.1 Scrapping de la página web de INEFOP

Dado que se deseaba total control de los enlaces a visitar e información a obtener, se realizó el scraping mediante solicitudes GET y un algoritmo DFS (Depth First Search). La página web de INEFOP[1] contiene gran cantidad de información; mucha de esta se encuentra obsoleta o es irrelevante. Para lidiar con este inconveniente se excluyó todo enlace por fuera de los dominios www.inefop.org.uy/, https://www.empresas.inefop.uy y https://oportunidades.inefop.org.uy, la sección de noticias y algunas urls particulares. Se eligieron algunos enlaces como una muestra para analizar y tomar ciertas decisiones que ayudarán a eliminar información irrelevante. Con la ayuda de BeautifulSoup[36] se eliminó todo texto que estuviera por fuera de *tags headers* (H1, .. H6) títulos y párrafos.

En las primeras aproximaciones al algoritmo final, se observó que requería de mucho tiempo. Esto era debido a que se descargaban gran cantidad de archivos multimedia. Esto fue solucionado evitando la descarga de los archivos verificando si el enlace finaliza con alguna extensión no deseada. Ejemplos de estas extensiones son: .mp4, mp3, zip, .ppt, .jpg, entre otras. Dado que mucha información importante se encuentra en archivos word y pdf, se agregaron estas extensiones a las permitidas y se utilizaron diferentes librerías con el fin de poder recuperar esta información.

5.2 Procesamiento de preguntas e información de la web

Se presta especial atención al procesamiento de las preguntas ya que es donde se espera que haya mayor cantidad de errores ortográficos. Se filtran los términos de la pregunta de manera que queden únicamente nombres propios, sustantivos, adjetivos y verbos. Este último fue incorporado debido a los buenos resultados obtenidos en la evaluación presentada más adelante en la sección 5.3. Se realiza una búsqueda de los términos en las entradas de datos de las distintas webs y se retorna los resultados ordenados según la cantidad de ocurrencias y cantidad de términos similares.

Dada la posibilidad de que la pregunta tenga errores de tipeo, se hace uso de *fuzzy query*, una funcionalidad facilitada por Elasticsearch donde los términos pueden diferir algunos grados

según la distancia de Damerau-Levenshtein. Esta distancia, ya descrita anteriormente, mide la cantidad de cambios de a caracter necesarios para llegar de una palabra a otra. Estos cambios incluyen:

- Cambiar una letra: **c**el → **t**el
- Eliminar una letra: curso**s** → curso
- Insertar una letra: curso → curso**s**
- Transponer dos caracteres contiguos: tra**sn**porte→ tra**ns**porte

La distancia utilizada es la recomendada por Elasticsearch; si la palabra tiene 2 o menos caracteres la distancia es 0, si la palabra tiene entre 3 y 5 caracteres la distancia es 1 y si la palabra tiene más de 5 caracteres la distancia es 2.

Tanto a la información importada de la web como a la pregunta procesada que se busca en Elasticsearch se le aplica *stemming*, esto es aplicado utilizando el módulo *SnowballStemmer* obtenido de NLTK. Esto se incluyó con el fin de que las búsquedas pudieran generalizar entre las categorías léxicas que contienen la misma raíz y además evitaran perderse diferentes conjugaciones verbales o sustantivos con distinto número/género por ejemplo. Pero debido a la existencia del *fuzzy query* y/o mecanismos internos desconocidos de indexación de Elasticsearch, la mejora fue apenas notoria.

5.3 Comparación y resultados

Para evaluar la recuperación de información en la web en función de la pregunta ingresada por el usuario, se buscó una medida que tomara en cuenta la posición en la que se retorna el primer enlace que contenga una respuesta válida para dicha pregunta. Es por esto que se decidió utilizar MRR (Mean Reciprocal Rank).

La medida MRR consiste en obtener el promedio del inverso del rank recíproco de cada respuesta válida y se calcula con la siguiente fórmula:

$$\frac{1}{n}\sum_{i=1}^{n}\frac{1}{r_i}$$

n - Cantidad de preguntas a evaluar.
i - Número de pregunta.
r_i - Rank recíproco del primer enlace válido.

Para clarificar el concepto de rank recíproco y entender mejor el resultado final, hay que notar que:

- Si la primera respuesta válida está en primer lugar, el rank recíproco es igual a 1/1 = 1.
- Si la primera respuesta válida está en segundo lugar, el rank recíproco es igual a ½ = 0.5.

Para la evaluación se seleccionaron 20 preguntas o solicitudes de información de un conjunto de preguntas enviadas por usuarios de INEFOP mediante un sistema de gestión que manejan actualmente. Al momento de la evaluación se tenían 582 links como posibles respuestas en el sistema.

Se evaluaron dos variaciones del algoritmo en función de la tarea POS-tagging. En la primera variación del algoritmo (Algoritmo 1) se tomaron en cuenta determinantes, adjetivos y sustantivos. Para la segunda variación del algoritmo (Algoritmo 2) se incluyeron además los verbos.

En el Cuadro 5.1 se muestran ejemplos de preguntas o mensajes con su correspondiente respuesta y ranking en cada algoritmo.

Pregunta / Mensaje	Respuesta	Posición en el ranking **Algoritmo 1**	Posición en el ranking **Algoritmo 2**
Como puedo hacer para contactarme?	http://www.inefop.org.uy/Contacto/Contacto-uc1372	85	1
Quiero saber información sobre los cursos de gastronomía disponibles.	https://oportunidades.inefop.org.uy/index.php/producto/auxiliar-de-cocina/	2	1
Desde hace más de 15 años ejerzo como tarotista, enseño tarot y otras terapias alternativas. Quisiera saber cómo hacer para certificar mi experiencia porque varias alumnas me lo piden y es momento de hacerlo.	http://www.inefop.org.uy/Capacitarse/Si-queres-certificar-tus-competencias-laborales-uc1709	30	3

Cuadro 5.1: Preguntas y mensajes de ejemplo con su correspondiente respuesta y la posición de la respuesta en el ranking para cada algoritmo

Luego de la evaluación se obtuvieron los resultados presentados en el Cuadro 5.2.

Algoritmo	MRR
Algoritmo 1 (No incluye verbos)	0,436
Algoritmo 2 (Incluye verbos)	0,728

Cuadro 5.2: Resultados de los algoritmos en base a la medida MRR.

Si bien el Algoritmo 2 muestra predominancia sobre el Algoritmo 1, en ambos se obtienen muy buenos resultados. Se observa que para el segundo algoritmo se obtiene en promedio una respuesta que se encuentra en la primera o segunda posición, para el primer algoritmo la respuesta se encuentra en promedio apenas debajo del segundo lugar.

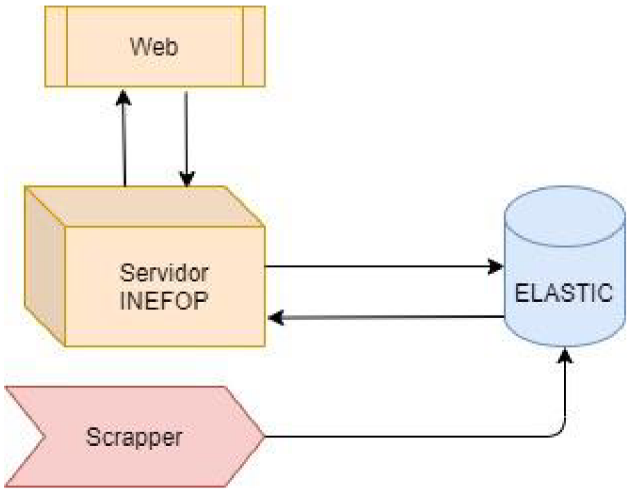
6. Implementación

Este capítulo introduce el sistema solución creado. Se incluye una descripción de la arquitectura general y de cada componente específico. Se incluyen las especificaciones del servidor donde se levanta el sistema y se detalla el procedimiento llevado a cabo.

La arquitectura se divide en 4 componentes principales. La interacción entre ellos es visible en la figura del Cuadro 6.1. Los componentes son los siguientes:

- ❏ Base de datos Elasticsearch
- ❏ Un módulo empleado para el scraping de la web
- ❏ Backend (API)
- ❏ Frontend (Web)

La parte del backend o la API difiere dependiendo del enfoque utilizado ya que se afecta las especificaciones de algunos componentes. Para el enfoque de detección mediante redes siamesas el desarrollo llevó aproximadamente el 90% del tiempo de proyecto pero su eficiencia resultó menor. El enfoque de detección mediante *sentence embeddings* se realiza en un tiempo considerablemente más corto, pero con resultados superiores, como se vio en la sección 4.3. Creemos conveniente presentar ambas arquitecturas.



Cuadro 6.1: Presentación de los distintos componentes principales y la interacción que ocurre entre ellos.

6.1 Base de datos Elasticsearch

Elasticsearch[35], como se menciona en la sección 2.2.5, es un motor de analítica y análisis distribuido, gratuito y abierto para varios tipos de datos, entre ellos textuales, numéricos,

geoespaciales, estructurados y no estructurados. Está desarrollado a partir de Apache Lucene y ofrece API REST simples dentro un sistema distribuido, veloz y escalable.

Es posible realizar y combinar muchos tipos de búsquedas: estructuradas, no estructuradas, geográficas, métricas, etc. Los tiempos de respuesta son extremadamente rápidos gracias al diseño que incluye índices invertidos, árboles KDB (árboles balanceados de k dimensiones) y más. La escalabilidad es posible y eficiente, simplemente se deben agregar más nodos de forma horizontal mientras que la comunicación se mantiene como si fuera un solo nodo. También está preparado para manejar errores humanos como por ejemplo errores tipográficos. Por todos estos motivos, se considera esta herramienta suficientemente buena para alojar los datos y realizar las búsquedas necesarias.

Se instala Elasticsearch, en principio con un solo nodo y un solo índice, INEFOP. Este se divide en 2 *shards* con una sola réplica, siendo un shard una instancia del índice independiente que funciona como motor de búsqueda y administra un subconjunto de datos. Se declara el mapeo de los parámetros como dinámico en caso de que haya que agregar nuevos campos. A continuación se presentan los campos empleados:

- **question**, de tipo *text*, donde se aloja una pregunta del estilo de las encontradas en los documentos de FAQ. La búsqueda de preguntas se basa en este campo únicamente para la versión de detección mediante siamesas, igualmente es imprescindible ya que identifica la pregunta a retornar.
- **question_vector***,* de tipo vector, donde se guarda la pregunta codificada en un *sentence embedding* de 512 dimensiones calculado por un modelo particular. Este campo es probado y empleado en la versión de detección por *embeddings*. La búsqueda por vectores mejora ampliamente los resultados retornados.
- **answer**, de tipo *text*, que contiene la respuesta a la pregunta indicada y es necesario a la hora de contestar la pregunta que introduce el usuario.
- **entry**, de tipo *text*, cuyo valor puede ser de tipo [*question, website, document*] y distingue el tipo de esa entrada. Los campos disponibles varían dependiendo del tipo, por lo que es necesario identificar qué se está buscando.
- **link**, de tipo *text*, empleado para guardar el url de la web o del documento en cuestión.
- **data**, de tipo *text*, que contiene la información de la web o del documento encontrados en la url dada. Esta información se limpia previo a su guardado, para lo que se eliminan por ejemplo tags html.
- **stems**, de tipo *text*, este campo incluye exactamente la misma información que *data* pero ha pasado un proceso de *stemming*. Inicialmente se esperaba que haciendo uso de este campo, las búsquedas se vieran generalizadas y resultaran más exitosas, pero al parecer Elasticsearch ya emplea herramientas internas de este estilo por lo que parecería ser un recurso duplicado.

6.2 Modulo de Scraping

Su función es la de almacenar la información que será utilizada por el servidor dentro de Elasticsearch, en particular debe realizar el *scraping* completo de la web de INEFOP[1] y almacenarla en un formato estructurado tal como se detalla en la sección 5.1. Aparte, debe ocuparse de almacenar las FAQ que fueron facilitadas por el usuario responsable.

En Elasticsearch se inserta la cuádrupla (*entry, link, data, stems*), siendo *entry* de tipo *website* o *document*, dependiendo si se está trabajando con una web o un doc/pdf, *link* la URL que se acaba de scrapear, *data* la información scrapeada y *stems* el proceso de stemming de *data*. Apache Tika[37] fue utilizado para leer archivos PDF, al ser una herramienta en Java, se utilizaron los *bindings* para Python y se levanta un servidor mediante una directiva incluida en los bindings. Al no encontrarse una herramienta óptima para la lectura de archivos word en múltiples formatos, se optó por utilizar diferentes librerías para los archivos doc y docx; estas fueron textract[38] y docx2txt[39] respectivamente. Finalmente se termina eliminando Textract debido a que la cantidad de .doc es reducida, desactualizada y las dependencias entraban en conflicto con otras librerías empleadas en el modelo de *embeddings*.

Para las FAQ, se inserta la cuádrupla (*entry, question, answer, question_vector*) siendo este último introducido en el enfoque final. Las preguntas con su respuesta son leídas de un archivo .tsv en el cual se almacenó lo obtenido de los documentos de preguntas relevantes. Suman un total de 34 preguntas.

6.3 Backend

El siguiente componente a describir es el servidor que contiene el backend del proyecto. Se implementa en modo RESTful API, o sea que conforma una arquitectura REST permitiendo la comunicación con los servicios mediante solicitudes http.

Dado que hoy en día las bibliotecas predominantes de PLN están implementadas en Python, se decide utilizar este lenguaje como base, el cual tiene a disposición una gran cantidad de librerías, entre ellas Flask, un micro framework web que permite levantar un servidor y habilitar *endpoints*; aparte que soporta el agregado de librerías externas. Negocios conocidos como LinkedIn[1] o Pinterest[2] utilizan Flask.

[1]Pinterest: https://www.pinterest.com/
[2]LinkedIn: https://www.linkedin.com/

Se exponen 4 *endpoints* principales cuyas especificaciones son visibles en la tabla del Cuadro 6.2. Se incluye una descripción para cada uno a lo largo de esta sección. Para el *endpoint* de preguntas similares se describen los dos enfoques vistos, detección mediante redes siamesas y mediante embeddings, ya que la implementación varía considerablemente de uno a otro.

Endpoints	**Descripción**
/similar_questions	Recibe una pregunta en texto plano y retorna 3 preguntas junto con su respuesta
/similar_webs	Recibe una pregunta en texto plano y retorna 3 links de webs/documentos
/question_feedback	Recibe dos preguntas en texto plano y un número que indica similitud
/link_feedback	Recibe una pregunta en texto plano, un link, un número que indica similitud, y un campo de texto para agregar un comentario.

Cuadro 6.2: Breve descripción de los argumentos que reciben o retornan los *endpoints* principales expuestos por la API.

similar_questions

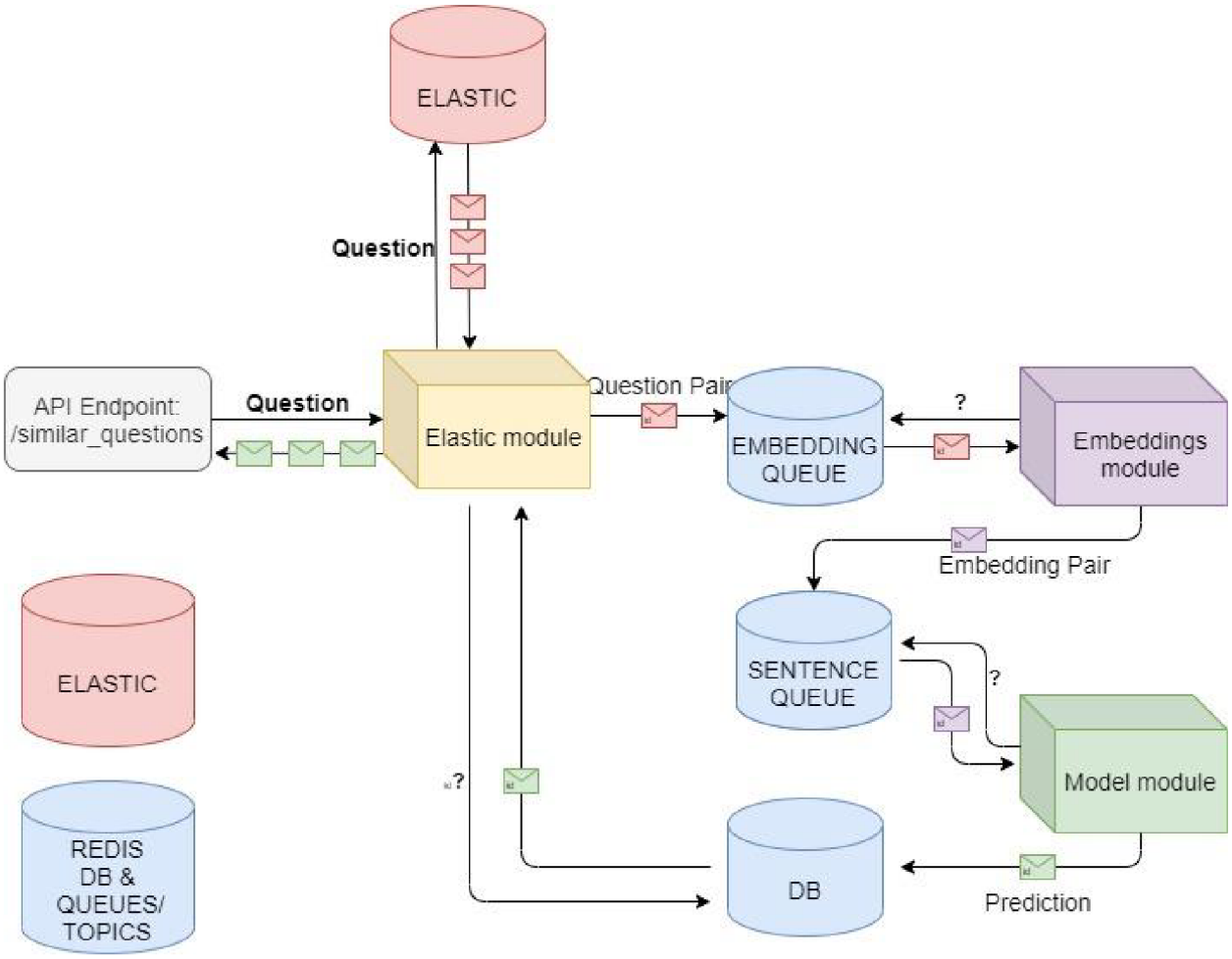
Este endpoint recibe una pregunta enviada por el usuario y debe retornar aquellas preguntas que más se parezcan, con el fin de encontrar una pregunta equivalente a la ingresada cuya respuesta conteste la pregunta original.

El proceso completo es descrito en la sección 4.1, este es visible en el Cuadro 6.3 donde se presentan los módulos y la forma en que interactúan. Existen 3 módulos principales.

El primer módulo, encargado de la interacción con Elasticsearch , toma la pregunta como se ingresa y realiza una recuperación sobre las entradas de tipo *entry-question* ingresadas allí. Se recuperan aquellas 5 preguntas con mayor puntuación.

Previo a realizar la búsqueda, se le aplica un preprocesamiento al texto ingresado. Se obtiene un modelo entrenado para el español de Spacy[24]. Se elige el modelo más chico entre cuatro modelos de diferentes tamaños asumiendo que existen restricciones en el espacio que pueda llegar a tener el usuario responsable en su computadora o el servidor donde se instale el sistema. El preprocesamiento se inicia con la eliminación de *stopwords*, un conjunto de palabras usadas comúnmente en el lenguaje que no contienen información de relevancia, por ejemplo pronombres, determinantes, preposiciones, etc. Luego, haciendo uso del modelo elegido, se

realiza la *tokenización* y consecuentemente el *etiquetado POS*. Esto consiste en distinguir cada una de las palabras y la categoría gramatical a la que pertenecen. A continuación se filtran aquellas palabras que no pertenezcan a las categorías gramaticales: nombre propio, sustantivo, adjetivo y se realizan pruebas filtrando o no los *verbos*. El resultado del preprocesamiento se presentan en los ejemplos 6.1 y 6.2.



Cuadro 6.3: Presentación de los distintos módulos en forma de prisma cuadrangular y las herramientas de almacenamiento en forma de cilindros. Las flechas indican las interacciones y los sobres son una representación de las preguntas incluídas en las respuestas, cada una identificada por un id.

Ejemplo 6.1

¿Cuántos cursos puedo hacer a la vez? → cursos / cursos puedo

Ejemplo 6.2

¿Cuáles son los teléfonos de inefop? → teléfonos inefop

Palabras como *hacer*, *vez* o *son* entran en la categoría de stopwords. Este resultado es lo que se utiliza para la búsqueda en Elasticsearch. Se obtienen aquellas 5 preguntas con mayor ranking y se ingresan emparejadas con el input original y un *uuid* para facilitar su identificación en el tópico de *embeddings*.

La herramienta Redis[1] se utiliza para generar tópicos a los cuales se suscribirían los módulos, logrando la comunicación en forma de paradigma pub/sub[2]. Esto debido a que existió una confusión con la sesión de los modelos que debía mantenerse activa, y la única forma que se encontró fue levantando un proceso en paralelo que consultara un tópico constantemente. Esto resultó precario y agregó complejidad innecesaria.

Luego de que las 5 preguntas con mayor puntuación son ingresadas en el tópico, el módulo de Elasticsearch, que guardó los *uuid*[3], queda consultando por los mismos en la base de datos de Redis, donde se escribirán los resultados finales.

El segundo módulo, se encarga de la conversión de la tira de palabras, a una tira de *embeddings*. El modelo empleado es descrito en el capítulo 4. El resultado del par de preguntas se inserta, con su *uuid* correspondiente, en un tópico consultado por el último módulo.

El último módulo, se ocupa de realizar la predicción de similitud sobre las tiras de *embeddings* y escribe el resultado final identificado por el id en la base de datos. Este es accedido y retornado por el primer módulo que retorna un conjunto de preguntas ordenadas por similitud.

Los resultados del enfoque mediante redes siamesas no resultaron satisfactorios, la arquitectura es exagerada, la forma en la que se procesa el texto no parece adecuada y no aparenta que haya mejora posible. Se plantea el nuevo enfoque cuya arquitectura puede verse en el Cuadro 6.4.

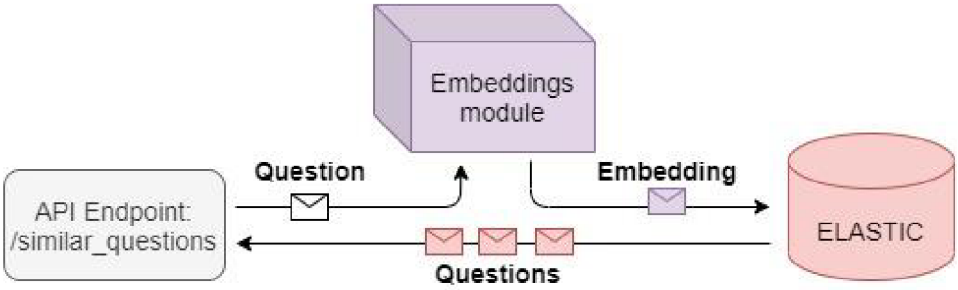
El paso de preprocesamiento se mantiene igual. El módulo de *embeddings* se conserva pero realizando un cambio en el modelo utilizado. En vez de retornar una lista de vectores se retorna un solo vector que representa toda la frase. Finalmente, la búsqueda que se lleva a cabo en la base de datos se hace empleando este vector. Se descarta el uso de comunicación mediante tópicos, ya que se establece una clase para contener al modelo que se inicializa durante el *setup* y se mantiene en una variable a lo largo de la sesión.

---

[1]https://redis.io/

[2]Paradigma b/sub: emisores (pub) no envían sus mensajes a receptores (sub) específicos, sino que publican mensajes en un canal, y los receptores se suscriben a uno o varios canales sin conocimiento de quién emite los mensajes. Este paradigma habilita una mayor escalabilidad y dinamismo en la red.

[3]uuid: identificador único universal.



Cuadro 6.4: presenta el segundo enfoque donde se reduce el tamaño de la arquitectura. Se emplea únicamente un módulo y Elasticsearch.

similar_webs

Para este endpoint se envía una pregunta y se debe retornar links a la web de INEFOP donde potencialmente se encuentre la respuesta, o información relevante al texto ingresado. Cuando la pregunta llega al servidor se realiza el proceso indicado en la sección 5.2 sobre el contenido de cada una de las páginas que se encuentran en el sitio web y se retornan los enlaces asociados.

Este endpoint, aparte de las webs, aplica el mismo proceso por separado para los documentos y retorna los más similares para ambos, distinguidos en webs y docs.

question_feedback

Este endpoint recibe 3 parámetros, siendo dos de ellos preguntas y el tercero un número que debe ser 0 o 1 e indica si estas dos preguntas son similares semánticamente. Estos parámetros son ingresados a una *sheet* de Google contenida en la nube, quedando almacenados para su posterior uso.

Inicialmente, se propone este endpoint para juntar un set de datos con el cual entrenar el modelo presentado en la primera versión. Las sheets son fácilmente exportables a formatos como *.csv*, *.tsv*, comúnmente utilizados para el almacenamiento de corpus ya que resulta sencillo importarlos a estructuras de datos, por ejemplo tablas.

Se contaba con que el usuario responsable utilizara el sistema de prueba y recabara suficientes ejemplos, pero eso no ocurrió. Luego de efectuado el cambio de enfoque, este *endpoint* deja de ser relevante, dado que no se utiliza el modelo que requiere entrenamiento propio de este corpus. Igualmente se mantiene el endpoint para visualizar el uso y resultados de la aplicación.

link_feedback

Similar al endpoint anterior este se ocupa del almacenamiento de los resultados para la sección de búsqueda en webs. Recibe como parámetros la pregunta original, un link, un valor de éxito que debe ser 0 o 1 y un argumento opcional, cuyo fin era almacenar un extracto de texto encontrado en el link que respondiera la pregunta ingresada. Estos datos también quedan registrados en una *sheet* de Google.

El objetivo era generar y almacenar un corpus con el cual afinar el entrenamiento de un hipotético modelo de *question answering* pre entrenado. El uso que se le da es mínimo, apenas cuenta con algunas entradas ingresadas. De manera similar, éste modelo nunca es implementado.

6.4 Web

La interfaz de usuario fue desarrollada como una página web utilizando Angular 11.2.2 y es desplegada en un servidor NodeJS. Consta de una sola página HTML que contiene tres secciones disponibilizadas mediante la manipulación de la visibilidad de los componentes. Esto fue decidido así ya que se está constantemente navegando entre las secciones y este manejo permite que el único tiempo de espera sea el necesario para renderizar los componentes.

La primera sección y también pantalla de inicio al ingresar a la página, contiene una caja de texto donde el usuario puede ingresar una pregunta y un botón que envía la pregunta al servidor.

La segunda sección contiene las posibles preguntas similares a la ingresada en la sección anterior, permite al usuario seleccionar una o más preguntas, o en su defecto, establecer que ninguna de las opciones se corresponde con su pregunta.

La tercera sección presenta enlaces relativos a http://www.inefop.org.uy/ donde puede estar la respuesta a la pregunta del usuario. Se tiene la posibilidad de elegir una de estas opciones o en su defecto, puede ingresar manualmente la respuesta en texto plano o el enlace donde se encuentra su respuesta.

Cada una de estas decisiones del usuario es enviada al servidor ya que es información valiosa para alimentar el sistema posteriormente. Para estas solicitudes se utiliza el prefijo /api, de esta manera se evita el intercambio de recursos de origen cruzado. Estas solicitudes son recibidas por un proxy dentro del servidor NodeJS y redirigidas a la api.

6.5 Servidor

Ante la necesidad de encontrar una forma anticipada de hacer llegar el sistema al usuario responsable, se propone dejar disponible un servidor. De esta forma podrían probar la herramienta así como incluir ejemplos de uso. A continuación se mencionan algunas maneras de hacer esto.

Primero se plantea como posibilidad instalar el software en la infraestructura del usuario. Esto implica tener que instalar las herramientas empleadas tal como Elasticsearch o Redis en un sistema operativo diferente al utilizado en desarrollo y levantar todo el sistema de forma local, lo que incluye realizar y almacenar el proceso de scraping y almacenar los modelos de procesamiento de lenguaje de Spacy y de *embeddings*. Todo esto tiene un tamaño no despreciable que a priori se desconocía si el usuario tendría disponible. El poder de cómputo requerido para ejecutar los modelos de similitud también es considerable y el acceso a hardware suficientemente potente tampoco se podía garantizar. Finalmente, también se incluye el hecho de que toda la instalación se debía hacer de manera remota porque nos encontrábamos en un momento grave de la pandemia por coronavirus, esto a pesar de no ser un problema mayor ya que existen hoy en día herramientas que te permiten realizar conexiones remotas a equipos, significaba un escalón extra al proceso.

Como segunda opción, se piensa en las distintas posibilidades de *cloud-computing* disponibles hoy en día, como Google Cloud, Amazon AWS, Heroku, entre otras. Estas plataformas ofrecen posibilidades de alojamiento y almacenamiento para sistemas, el uso es sencillo y proveen opciones de escalado, se pueden aumentar o quitar requerimientos a gusto. A pesar de tener una opción inicial gratuita, lo que se provee no es suficiente, por lo que levantar el sistema necesario implica un costo que no corresponde dado que esto es una tesis de grado de una prueba de concepto.

La tercera opción discutida, es la de alojar el sistema en hardware propio. En particular, uno de los compañeros cuenta con una torre desocupada cuyas especificaciones son suficientes para levantar y almacenar el sistema. La necesidad por disponibilidad implicaría dejar la torre constantemente prendida atendiendo llamados externos, actuando como servidor durante un tiempo prolongado. Esto conlleva un gasto eléctrico que se prefiere evitar con la proposición de la siguiente opción.

Finalmente, la opción elegida es propuesta por los tutores de la tesis. Recientemente el InCo compró un servidor llamado Boole que se destina a proyectos como tesis de grado o proyectos internos de facultad, este servidor se facilita a demanda de estudiantes. Luego de gestionar especificaciones y procedimientos con un integrante de la Unidad de Recursos Informáticos, el sistema es albergado en una máquina virtual en boole.fing.edu.uy. Las especificaciones incluyen:

- Memoria RAM: 6GB
- Cores (CPU): 4
- Espacio en disco: 20GB
- Sistema operativo: CentOS 8 (Instalación mínima)

6.6 Dificultades Técnicas

A lo largo de la gestión y desarrollo de este proyecto, se presentan dificultades técnicas que confirman que procesos de este tipo no serán fluidos ni ágiles. Se encontraron problemas de carácter tanto tecnológico como social. Aquellos problemas de carácter social se describen en las conclusiones en la sección 7.

Llevar a cabo una prueba de concepto trae por defecto dificultades a superar. Su función es confirmar que es posible llevar a cabo un proyecto con resultados prometedores. Parte de la tecnología empleada en este proyecto, como lo son las redes neuronales, continúa en desarrollo/estudio y a pesar de ser una tecnología madura, su uso en general es a prueba y error. Esto se vio reflejado al momento de seleccionar y desplegar las características específicas y parámetros de la arquitectura, donde la cantidad de opciones era tan significativa que resultó abrumador. La selección, implementación y entrenamiento de la red fue el mayor reto tecnológico ofrecido por este proyecto. El tiempo dedicado entre lectura de material y desarrollo superó un semestre, difiriendo del plan de trabajo pactado inicialmente.

A nuestro parecer, la situación que causó el mayor atraso del plan de trabajo fue el tiempo que tomó la gestión y configuración del sistema en el servidor de la FING. El tiempo transcurrido entre que se transmite al cliente en una reunión que se dejaría la herramienta disponible y que la herramienta queda efectivamente accesible fue entre 2 y 3 meses.

Otro ejemplo de dificultad técnica de carácter tecnológico y a su vez social, vino del lado del usuario responsable. El uso de la plataforma una vez disponible resultaba indispensable. Esta era la única manera de obtener datos para re-entrenar los modelos buscando el *fine-tuning* y así logrando una mejora en su rendimiento. Se estableció en reuniones que el usuario responsable se encargaría de encomendar a operarios ocupados de responder consultas dentro de INEFOP el uso de la plataforma, aportando tanto nuevos datos como retroalimentación, comentarios u opiniones de los futuros usuarios resultan de utilidad cuando se busca mejorar un producto. Ocurrió que la cantidad de datos obtenida fue escasa. También se propusieron instancias de reunión con el personal de IT de INEFOP que nunca se concretaron. Existían deseos por parte del usuario responsable que existiera un canal de comunicación entre el personal de IT y nosotros. Esto refleja el hecho que al haber un proceso burocrático de por medio, por más que haya intención, a veces las cosas no se pueden llevar a cabo de forma veloz o directamente no se pueden llevar a cabo.

7. Conclusiones

En este último capítulo se describen las conclusiones sacadas del proceso que llevó a la realización del proyecto, así como del proyecto en sí. También se incluye trabajo e ideas de implementación futura.

Nos gustaría comenzar este capítulo agradeciendo la oportunidad presentada por los docentes y el personal de INEFOP. Se dio una oportunidad de resolver un problema interesante estudiando un campo de nuestro interés y haciendo uso de tecnologías en el estado del arte. El trayecto no fue simple pero resultó gratificante.

En primer lugar, se quiere destacar los buenos resultados obtenidos. No sólo las métricas obtenidas son satisfactorias, sino que también basta con utilizar el producto final para comprobar su eficiencia. Por otro lado, no deja de ser una opinión personal y no validada, hubiera sido enriquecedor evaluar el sistema con usuarios finales pero esto no fue posible debido a inconvenientes mencionados anteriormente.

Un punto importante a tener en cuenta sobre el software empleado, es la necesidad de tener disponible hardware decente donde correrlo. El hecho de emplear tecnología como modelos neuronales o de lenguaje resultó en una demanda de recursos que no había sido prevista. Las redes neuronales, a pesar de originarse en los 80s, continúan evolucionando y con esto surgen nuevos algoritmos de aprendizaje cada vez más potentes pero con una demanda incrementada de poder de cómputo. Fue necesario realizar el entrenamiento de nuestra red neuronal en etapas porque los recursos computacionales no eran suficientes.

La siguiente conclusión a mencionar se relaciona con la estimación y evolución del proyecto. Existieron retrasos debido a falta de cooperación o quizás expectativas elevadas a la hora de plantear cómo y cuándo se harían las tareas, el ejemplo más claro fue el tiempo que tomó disponibilizar la herramienta. Otro ejemplo es el uso prácticamente nulo que se le dio a esta luego de disponible cuando lo esperado era lo contrario. Muchas veces las características de un proyecto pueden ir cambiando sobre la marcha, en estos casos la adaptación es esencial. El trato con personas conlleva interpretaciones que pueden variar de individuo a individuo, propuestas o promesas que no se lleven a cabo y falta de disponibilidad. Aparte de los recién descritos, existieron varios motivos sociales que perjudicaron el cumplimiento del itinerario, situaciones imprevistas del día a día.

Como último punto social a destacar quizás no tan importante pero definitivamente influyente, está la situación sanitaria que transcurrió durante todo el proceso. Forzosamente todas las reuniones, gestiones y trabajo compartido fue de carácter virtual, trayendo esto los clásicos problemas de falta de conexión, fallos en dispositivos de comunicación, demoras en respuestas a peticiones y más.

El desarrollo de varios enfoques para solucionar un problema nos permitió realizar una comparación entre ellos. Se nos presentó una situación donde innovar desarrollando una arquitectura propia no dio los mejores resultados. En cambio emplear herramientas desarrolladas por grandes empresas como el modelo neuronal de Google en combinación con el uso innovador que se le da a la herramienta de RI, si resultó en una solución con una precisión inesperada. Al momento de desarrollar una solución no siempre es necesaria la creación de algo desde cero. El proceso creativo puede implicar moldear de manera ingeniosa herramientas ya existentes, encontrando usos inusuales que aporten a la solución.

Siguiendo la línea de utilizar herramientas ya desarrolladas, el web scraping se implementó desde cero. Si bien esta decisión fue tomada con la intención de tener total control sobre qué páginas eran necesarias y qué páginas debían ser descartadas, posteriormente se descubrieron herramientas que podrían haber realizado un trabajo similar. Esta implementación dejó un gran aprendizaje pero también llevó un considerable tiempo de desarrollo, esto fue debido principalmente a la necesidad de mejorar el tiempo de ejecución, lo cual fue logrado con éxito.

En cuanto a la recuperación de información, se obtuvieron muy buenos resultados. El objetivo de entregar un link que contenga la respuesta correcta a una pregunta se cumple en la mayoría de los casos. Existen algunos casos donde existe una respuesta válida y no se entrega correctamente, esto muestra que la herramienta aún puede mejorar. Esta mejora otorgaría más confianza al usuario cuando se le indica que no existe una respuesta para su pregunta en la web de INEFOP.

Como trabajo futuro, se puede incluir el desarrollo de un modelo de Q&A sobre la web de INEFOP. Este puede emplear como base un modelo de análisis en español y luego ser entrenado más específicamente sobre las posibles consultas a responder en la web. Para esto se requieren una gran cantidad de nuevos ejemplos derivados de las distintas páginas dado que existen diversos dominios donde se abordan variedad de temas. Si se quiere acotar esta idea, se puede descartar la totalidad de la web, planteando este mismo proceso únicamente sobre el documento de bases generales descrito en el capítulo 3.2.

Otra prueba interesante a considerar es la verificación de que tan generalizable es el proceso desarrollado a otras realidades. El costo que podría tener la adaptación de la solución tecnológica a un problema similar pero con distintos parámetros, como otro lenguaje o un área diferente a consultas sobre procesos burocráticos.

8. Referencias bibliográficas

[1] INFOP - Instituto Nacional de Empleo y Formación Profesional, URL http://www.inefop.org.uy/home
[2] Christoph Bartneck, Christoph Lütge, “What Is AI?”, An Introduction to Ethics in Robotics and AI, pp.5-16, 2021.
[3] Graham Oppy, David Dowe, “The Turing test”, Stanford Encyclopedia of Philosophy, 2003
[4] Ajit Jaokar, “Twelve types of Artificial Intelligence (AI) problems, 2017, Data Science Central
[5] Elon Musk: ‘Mark my words — A.I. is far more dangerous than nukes’, CNBC
[6] Daniel Jurafsky, James H. Martin. “Speech and Language Processing - An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition”, Prentice-Hall, 2000.
[7] Tom M. Mitchell, “Machine Learning”, WCB/McGraw-Hill, 1997.
[8] Yann LeCun, Yoshua Bengio, Geoffrey Hinton, “Deep Learning.” Nature, vol. 521, no. 25 May, 2015, pp. 436-444, 2005.
[9] Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean, “Efficient Estimation of Word Representations in Vector Space”, Proceedings of Workshop at ICLR. 2013.
[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”, Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), 2019.
[11] Jose Cañete, Gabriel Chaperon, Rodrigo Fuentes, Jou-Hui Ho, Hojin Kang, Jorge Pérez, “Spanish Pre-Trained Bert Model and Evaluation Data”, Universidad de Chile, 2020.
[12] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwa,l Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, Dario Amodei, “Language Models are Few-Shot Learners”, OpenAI, 2020.
[13] OpenAI’s API Now Available with No Waitlist, https://openai.com/blog/api-no-waitlist/
[14] Semantic Textual Similarity Wiki - STS Resources, URL http://ixa2.si.ehu.eus/stswiki/index.php/Main_Page
[15] Eneko Agirrea, Carmen Baneab, Claire Cardiec, Daniel Cerd, Mona Diabe, Aitor Gonzalez-Agirrea, Weiwei Guof, Inigo Lopez-Gazpioa , Montse Maritxalara, Rada Mihalceab, German Rigaua, Larraitz Uriaa, Janyce Wiebe, “SemEval-2015 Task 2: Semantic Textual Similarity, English, Spanish and Pilot on Interpretability”, Proceedings of the 9th International Workshop on Semantic Evaluation, 2015, pp. 252–263, 2015.

[16] Wanpeng Song, Min Feng, Naijie Gu, Liu Wenyin, “Question Similarity Calculation for FAQ Answering”, Semantics, Knowledge and Grid, Third International Conference on, 2007.
[17] WordNet, large lexical English Database, URL https://wordnet.princeton.edu/
[18] Paul Neculoiu, Maarten Versteegh, Mihai Rotaru, “Learning Text Similarity with Siamese Recurrent Networks”, Proceedings of the 1st Workshop on Representation Learning for NLP, pp. 148–157, 2016.
[19] João Vitor Andrioli de Souza, Lucas Emanuel Silva E Oliveira , Yohan Bonescki Gumiel , Deborah Ribeiro Carvalho, Claudia Maria Cabral Moro, “Exploiting Siamese Neural Networks on Short Text Similarity Tasks for Multiple Domains and Languages”, Pontifical Catholic University of Paraná, Springer Nature Switzerland, pp. 357–367, 2020.
[20] J. Mueller, A. Thyagarajan, “Siamese recurrent architectures for learning sentence similarity”.
AAAI, pp. 2786–2792, 2016.
[21] Gensim - Biblioteca de Python para modelado de tópico, indexado de documentos y recuperación de similitudes, URL https://pypi.org/project/gensim/
[22] Wenhao Zhu, Tengjun Yao, Jianyue Ni, Baogang Wei, Zhiguo Lu, “Dependency-based Siamese long short-term memory network for learning sentence representations”, PLOS ONE 13, 2018.
[23] Sentences involving compositional knowledge, SICK data set, http://marcobaroni.org/composes/sick.html
[24] Spacy, https://spacy.io/usage/linguistic-features#dependency-parse
[25] Sameera A. Abdul-Kader, John Woods, “Question answer system for online feedable new born Chatbot”, 2017 Intelligent Systems Conference (IntelliSys), pp. 863-869, 2017.
[26] Girish Kumar, Matthew Henderson, Shannon Chan, Hoang Nguyen, Lucas Ngoo, “Question-Answer Selection in User to User Marketplace Conversations”, 9th International Workshop on Spoken Dialogue System Technology, 2019.
[27] Henderson, M., Al-Rfou, R., Strope, B., Sung, Y., Lukács, L., Guo, R., Kumar, S., Miklos, B., & Kurzweil, R, “Efficient Natural Language Response Suggestion for Smart Reply.” ArXiv abs/1705.00652, 2017.
[28] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin, “Attention Is All You Need”, Advances in Neural Information Processing Systems 30 (NIPS 2017), 2017.
[29] Carlos Segura, Àlex Palau, Jordi Luque, Marta R. Costa-Jussà, Rafael E Banchs, “Chatbol, a Chatbot for the Spanish “La Liga””, 9th International Workshop on Spoken Dialogue System Technology pp 319-330, 2019.
[30] Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze, Introduction to Information Retrieval, ch. 1, pp. 1, Cambridge University Press, 2008.
[31] Jian-Yun Nie, Traditional IR models, Montreal University
[32] Andrzej Białecki, Robert Muir, Grant Ingersoll, Lucid Imagination, “Apache Lucene 4”, Proceedings of the SIGIR 2012 Workshop on Open Source Information Retrieval, 2012.

[33] Brian D. Sloan, “ES-ESA: An Information Retrieval Prototype Using Explicit Semantic Analysis and Elasticsearch”, City University of New York, 2017.
[34] Diagrama web INEFOP
https://drive.google.com/file/d/1az1tpmhX8BWYfEHvTBHoxKRvhxYtwHss
[35] Elasticsearch, https://www.elastic.co/es/what-is/elasticsearch
[36] BeautifulSoup, https://www.crummy.com/software/BeautifulSoup/bs4/doc/
[37] Apache Tika, https://tika.apache.org/
[38] Textract, https://textract.readthedocs.io/en/stable/
[39] Docx2txt, https://pypi.org/project/docx2txt
[40]Universal-sentence-encoder-multilingual, CNN Model, Google,
https://tfhub.dev/google/universal-sentence-encoder-multilingual/3
[41] Wiki40b-lm-es, Transformer XL, Google, https://tfhub.dev/google/wiki40b-lm-e1
[42] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le, Ruslan Salakhutdinov, “Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context”,Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, 2019.
[43] Ralf C. Staudemeyer, Eric Rothstein Morris, “Understanding Long Short-Term Memory Recurrent Neural Networks”, Schmalkalden University of Applied Sciences, Singapore University of Technology and Design, 2019.
[44] Julie Tibsharani, Text similarity search with vector fields , Elastic, 2019,
https://www.elastic.co/blog/text-similarity-search-with-vectors-in-elasticsearch

9. Glosario

accuracy Exactitud
API REST Es un estilo de arquitectura creado para guiar el desarrollo de software orientado a la Internet, define reglas para sobre las que diseñar el comportamiento de un sistema web
backpropagation Es un algoritmo con el cual entrenar redes neuronales durante el aprendizaje supervisado
bag of words Forma de representación de documentos en la cual se define un vector donde cada posición hace referencia a un token particular sin importar el orden. Luego, la representación de un documento se computa indicando 1 en la posición del vector si el token está contenido en el documento o 0 si no.
chatbot Software que simula una conversación humana mediante texto o voz
crash Cuando un programa computacional deja de funcionar correctamente de repente
embeddings Forma de representación empleada en el análisis de texto, consta de vectores numéricos que representan variable discretas, como palabras
endpoint Nodo de comunicación dentro de una red, puede exponer una interfaz respondiendo a peticiones externas
epochs Hiperparámetro que define la cantidad de veces que pasaremos un mismo conjunto de datos por un algoritmo o modelo, reforzando el aprendizaje
FAQ Frequently Asked Questions
fine tuning es una forma de mejorar la precisión de una red neuronal integrando lo aprendido por otro modelo. Se utilizan los parámetros internos del modelo ya entrenado para inicializar, así haciendo el entrenamiento más eficiente y corto.
INEFOP Instituto Nacional de Empleo y Formación Profesional
keywords palabras o conceptos importantes
lemmatization proceso mediante el cual se agrupan las formas flexionadas de una palabra, permitiendo el análisis como un único ítem
Distancia de Levenshtein cantidad de operaciones necesarias para transformar una cadena de caracteres en otra. Las operaciones pueden ser inserción, eliminación o sustitución
LinkedIn red social para profesionales, orientada a dar apoyo al proceso de contratación y visualización del perfil profesional
loss función de error que se busca minimizar para optimizar el aprendizaje de un modelo
LSTM Long Short Term Memory
Newswire Una agencia de noticias
open Source código diseñado de manera pública, cualquiera es capaz de visualizar, acceder o incluso modificar el código
output salida o valor resultante
Pinterest Red social enfocada a la distribución y exposición de imágenes y videos
pipeline Flujo de trabajo
PLN Procesamiento de Lenguaje Natural

precisión Porcentaje de instancias relevantes sobre instancias obtenidas
Q&A Question and answers
queries Consultas
Quora Red social enfocada en preguntas y respuestas hechas por los usuarios
readme Archivo que contiene información sobre el resto de los archivos encontrados en el directorio
recall Porcentaje de instancias relevante obtenidas sobre total de instancias relevantes
Redis Software open source de almacenamiento en memoria empleado como base de datos, caché, o broker de mensajes.
ReLU Rectified Linear Activation Function
RI Recuperación de Información
RNN Recurrent Neural Network
scraping Extraer contenido y datos de un sitio web automáticamente
sentence embeddings Forma de representación vectorial para conjuntos de palabras que conformen una frase u oración
stem Base o raíz de una palabra
stemming Procedimiento mediante el cual se remueve parte de una palabra reduciéndose a su raíz (stem)
stopwords Palabras que no tienen significado por si mismas pero que se emplean en la comunicación ya sea para hacer referencias, acompañar o modificar otras palabras.
tahn Hyperbolic Tangent
weights Parámetros de aprendizaje dentro de un modelo, son los valores numéricos para el peso de cada neurona en las distintas capas
wikipedia Enciclopedia online gratuita creada por voluntarios ubicados en todo el mundo

10. Anexo

10.1 Resúmenes de trabajos seleccionados

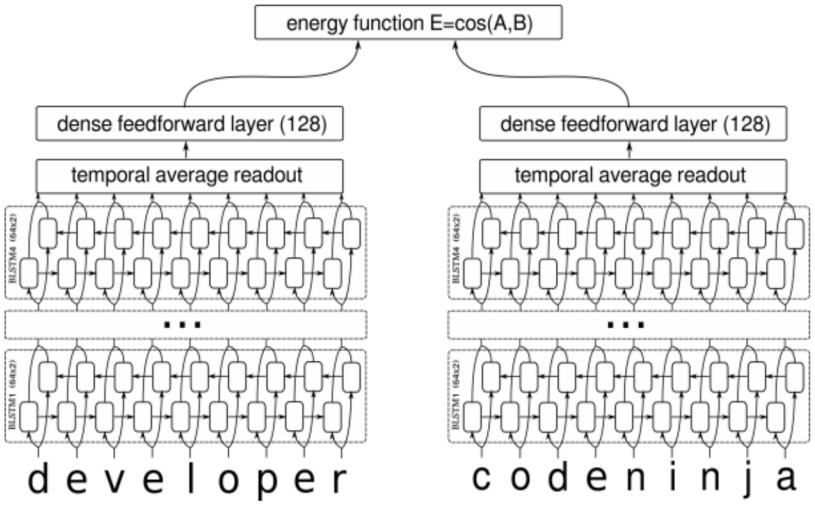
P. Neculoiu et al.

P. Neculoiu et al. [18] trabajan en un sistema para clasificar títulos de puestos. Se describe la normalización de textos como la tarea de transformar un texto en su forma canónica. La tarea en cuestión incluye recibir un título y mapearlo dentro de un conjunto predefinido de puestos de trabajo, al más parecido.

Mencionan que a esta tarea se le puede encarar y usualmente es vista como un problema de clasificación multi-clase, pero que en este estudio el enfoque que se le da es el de aprender un espacio de representaciones en los textos tal que títulos de trabajos similares estén cercanos.
El enfoque es flexible, ya que las representaciones aprendidas pueden emplearse como entrada de un modelo subsecuente que clasifique, o simplemente se pueden utilizar para encontrar títulos relacionados cercanos, o explorar clusters de títulos. En este enfoque de comparación de pares, el modelo empleado para su resolución es una red siamesa y se puede visualizar en el Cuadro 10.1.

Dado que las entradas son secuencias de tokens, se emplea una red neuronal capaz de procesar estas entradas, en particular, una variante de las redes neuronales recurrentes, la LSTM bidireccional. Esta es una red neuronal con memoria, que toma en cuenta contexto pasado y futuro ya que procesa la entrada en ambas direcciones.

Cada lado en la red siamesa produce un vector de largo fijo, que pasa por una capa densa y se finaliza realizando distancia coseno entre las salidas, esto permite entrenar con entradas de tipo <input1, input2, y>, siendo y=1 si los input son equivalentes, 0 si no.



Cuadro 10.1: Detalle la arquitectura empleada en el trabajo, una red neuronal siamesa que compara entre títulos de puestos de trabajo, ambos lados se componen por redes con capas LSTM (con memoria) finalizando con una capa densa cuyos resultados se introducen en una función de distancia de vectores, en este caso distancia coseno.

Para los experimentos, se obtiene el vector de salida de la LSTM entrenada, y se busca aquel más cercano dentro del espacio de vectores generado. En pruebas también se le realizan un par de modificaciones al set de datos para favorecer la robustez ante errores de tipeo, sinónimos o palabras superfluas. Obtienen en general resultados con accuracy mayores a 0.8.

J. V. Andrioli de Souza et al.

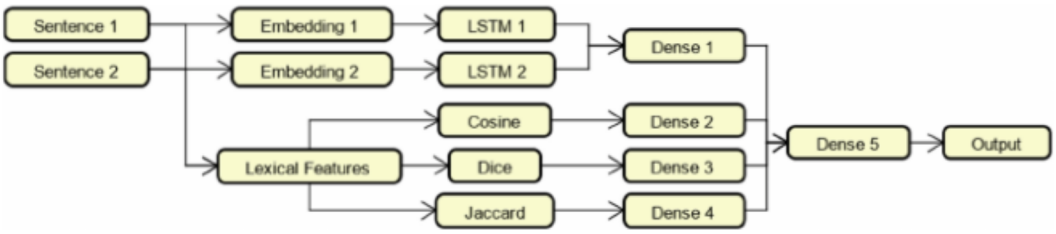
El siguiente trabajo de J. V. Andrioli de Souza et al. [19] busca desarrollar una única metodología que emplee la misma arquitectura y características para resolver la similitud entre un conjunto de textos extraídos de una base clínica en inglés, y otros extraídos de dominio periodístico en portugués. También emplean redes siamesas, argumentando que requieren menos datos de entrenamiento y que son menos susceptibles al *overfitting*, aparte que es conocido su éxito en cómputo de similitud entre dos instancias. Proponen complementar el modelo con ciertos parámetros y *word embeddings* pre entrenados.

Comienzan replicando la arquitectura utilizada en [20] , realizando un par de cambios:
- Primero modifican la función sigmoide de activación de la red por ReLU, esto debido a que las salidas de la función sigmoide están en el rango de 0 a 1, y ellos requerían al menos entre 1y 5.
- Además incluyen tres capas que procesan características léxicas en los textos, en particular se calculan la cantidad de tokens en común y se representan mediante similitud coseno[1], coeficiente de Dice[2], e índice de Jaccard[3], utilizando cada representación en una capa distinta. Al parecer,

estudios preliminares arrojaron una mejora del 5% haciendo uso de las tres medidas, en vez de una sola. Las características léxicas parecen ser un paso sencillo que podría ser incluido.

En cuanto al uso de *word embeddings*, prueban con unos pre-entrenados para Portugués y otros para Inglés, y aparte, generan sus propios *word embeddings* mediante la funcionalidad *word2vec* de la librería *gensim* [21], incluyendo los corpus de entrenamiento y testeo. La arquitectura de la red neuronal se presenta en el Cuadro 10.2.

Testean sobre 3 corpus distintos, obteniendo algunos resultados mejores y otros peores que otros trabajos con los que se comparan. Se cuestiona qué tan bien representan los conjuntos de entrenamiento a los de test y se destaca que tanto el uso de los *word embeddings* pre-entrenados como la inclusión de las características léxicas, en general mejora los resultados.
También se concluye que el hecho de requerir grandes modelos pre-entrenados que produzcan buenos embeddings para cada lenguaje es una limitación pero es necesario si se quiere obtener mejores resultados. Se menciona que una opción podría ser el uso de un modelo multi-lenguaje como BERT.



Cuadro 10.2: Composición de las diferentes capas que conforman la red neuronal. Hay dos flujos principales, uno que compara *embeddings* y otro que compara características léxicas, uniendo todo en una última capa densa.

[1]Similitud coseno es una medida de similitud entre dos vectores no nulos donde se calcula el coseno del ángulo entre ellos.
[2]Coeficiente de Dice es una medida de similitud para dos conjuntos de datos, la fórmula para calcularlo es 2 * |X ∩ Y| / (|X| + |Y|).
[3]Índice de Jaccard es otra medida de similitud para conjuntos de datos, la fórmula para calcularlo es |X ∩ Y| / |X ∪ Y|.

W. Zhu et al.

En el trabajo de W. Zhu et al. [22], se busca producir una representación vectorial de una oración particular también mediante LSTMs, pero agregando una *feature* de dependencias. Es posible visualizar la arquitectura en el Cuadro 10.3.

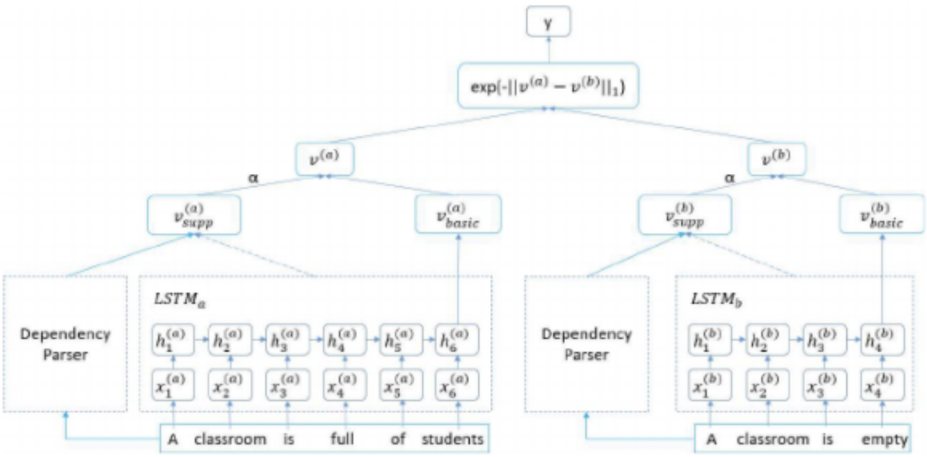
La D-LSTM emplea un parser de dependencias pre entrenado para obtener información y componentes relevantes sumado a un modelo LSTM estándar. Los componentes relevantes de una oración son principalmente el sujeto, predicado y objeto. También incluyen y experimentan con un factor de peso α entre [0,1] para el componente de dependencias del sistema, este balancea qué tanto aporta el componente al resultado final.

Explican que su modelo estándar, al igual que el trabajo anterior, se inspira en el modelo desarrollado por J. Mueller et al[20], donde se presenta una red siamesa de LSTM para detectar similitud entre oraciones, haciendo uso de *word embeddings* e información de sinónimos.
Destacan el uso de la métrica Manhattan[1] como finalización de la salida y mencionan que obtienen resultados que superan el estado del arte.

En las pruebas, experimentan con el sistema incluyendo ambos componentes, así como solo el componente base. Se emplean las medidas de evaluación: correlación de Pearson[2], correlación de Spearman[3] y *mean squared error*[4]. Terminan concluyendo que α entre [0.5, 0.6] arroja los mejores resultados. Por lo que el componente de dependencias parecería funcionar a la hora de mejorar el modelo.

La idea de usar un *parser* de dependencias presentada en este artículo, podría ser aplicable a nuestro trabajo. Actualmente existen parsers de dependencias para el español basados en modelos de lenguaje. Un ejemplo es Spacy[24], una biblioteca de código abierto muy popular empleada en el procesamiento de lenguaje natural.

[1]Manhattan es una medida de distancia entre dos puntos que equivale a la suma de la diferencia entre sus coordenadas cartesianas. Se presenta la definición en la sección 4.
[2]Pearson mide la relación estadística entre dos variables continuas. Puede tomar un rango de valores de +1 a -1. Un valor de 0 indica que no hay asociación entre las dos variables. Un valor mayor que 0 indica una asociación positiva y un valor menor que 0 indica una asociación negativa
[3]Correlación de Spearman es una medida de correlación entre dos variables aleatorias.
[4]Mean squared error es un estimador que mide el promedio de los errores al cuadrado.



Cuadro 10.3: Presentación de la arquitectura empleada en el trabajo que consta de un conjunto de neuronas LSTM y un procesamiento de características de dependencia. Para cada parte existe un factor de importancia ya que el trabajo realizado por las LSTM debe tener más peso.

Sameera A. Abdul-Kader et al.

Sameera A. Abdul-Kader et al. [25] presentan un trabajo dedicado a diseñar un chatbot alimentado con información de la web. Hacen hincapié en la importancia del procesamiento de la información y la extracción de características para poder obtener buenos resultados.

Lo interesante de la investigación es la variedad de métodos que se utilizan para la extracción de características, presentan la combinación de éstos como un nuevo método para cuantificar las respuestas del chatbot.
Entre los métodos utilizados se destacan

- Selección de características a utilizar: Importante para lograr simplificar el problema y eliminar información poco relevante.
- N-Match: Cantidad de palabras o frases en común entre una pregunta y una posible respuesta. Lo ideal es obtener una métrica que permita detectar qué tan similar es una palabra a otra y no una simple comparación de igualdad. Las dos opciones utilizadas para comparar resultados son Similitud Coseno y Coeficiente de Jaccard.
- POS tagging: Técnica que categoriza cada palabra o frase en su categoría gramatical, es útil para comparar frases sintácticamente.
- Similitud semántica: Cálculo matemático en función del significado de las palabras o frases para medir la similitud entre dos oraciones.

La arquitectura final consiste en preprocesar el texto obtenido de la web con técnicas conocidas como la tokenización tanto a nivel de oración como a nivel de palabra, eliminación de símbolos

redundantes y en distinto idioma, POS tag, etc. Luego se realiza el análisis de características entre pregunta y respuesta, el puntaje final se obtiene con la suma del puntaje semántico más el puntaje sintáctico más el puntaje léxico, los cuales se calculan concurrentemente.

En cuanto a los resultados, se muestran ejemplos de preguntas que devuelven la respuesta correcta en el ranking número 1. Para evaluar el sistema se utiliza el Mean Reciprocal Rank que consiste en ponderar una cantidad de preguntas determinada según el ranking de la respuesta correcta. Se obtienen muy buenos resultados presentados en el Cuadro 10.4, en especial para la Similitud Coseno.

No.	Combination	MRR score
1.	Combination using Jaccard's coefficient	37.13
2.	Combination using cosine similarity	52.37
3.	A. Moschitti, and S. Quarteroni [4].	43.25

Cuadro 10.4: Resultados del trabajo de Sameera A. Abdul-Kader et al. en sus tres variaciones.

Girish Kumar et al.

Girish Kumar et al. [26] plantean el problema de la cantidad exagerada de preguntas que se le hacen a un vendedor sobre sus productos publicados en un mercado en línea. Comentan que gran parte de las preguntas pueden ser respondidas utilizando la descripción del producto.

Para atacar este problema buscan extender un modelo presentado por Matthew Henderson et al. [27] en el cual se utiliza el producto vectorial de dos redes neuronales recurrentes, una para codificar pregunta y otra para generar la respuesta.
Este trabajo propone agregar redes LSTM y mecanismos de atención al modelo mencionado.

El procedimiento consiste en asignar una probabilidad a cada posible respuesta usando el producto vectorial de las redes. Para la comparación de resultados, utilizan distintas técnicas de codificación:

- Capa LSTM: Utilizada para agregar información secuencial, en particular utilizan una LSTM bidireccional.
- Contexto de conversación: Dado que el modelo utiliza solo la pregunta, para esta variante utilizan un modelo LSTM que incorpora todos los mensajes anteriores a la pregunta al vector final de pregunta.
- Capa de atención: Utilizan el modelo presentado en Tensor2Tensor [28] para mejorar la representación de las respuestas candidatas.

Los resultados alcanzados son bastante buenos y se presentan en el Cuadro 10.5.

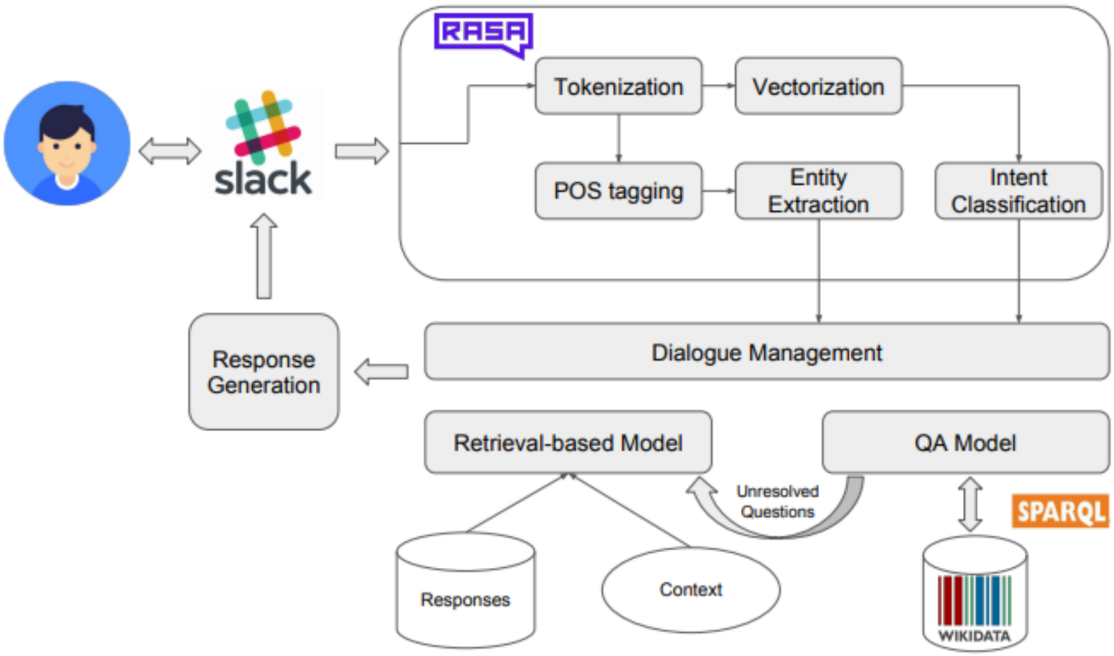
	Overall Accuracy	Positive Accuracy	Trigger Accuracy
Baseline	0.592	0.528	0.731
+ Pretraining	0.678	0.604	0.783
+ LSTM Layer	0.688	0.618	0.786
+ Attention	0.694	0.620	0.788
+ Conv. Context	0.710	0.624	0.804

Cuadro 10.5: Resultados del trabajo de Girish Kumar et al. en sus cuatro variaciones.

Overall Accuracy: Precisión sobre todo el conjunto.
Positive Accuracy: Precisión solo sobre un conjunto que contiene una respuesta dentro del texto.
Trigger Accuracy: Precisión sobre el conjunto que devuelve una respuesta.

Carlos Segura et al.

Carlos Segura et al. [29] presentan un trabajo basado en la construcción de un chatbot que conteste preguntas en español sobre "La Liga", la liga española de fútbol. Utilizan una arquitectura simple (Cuadro 10.6) basada en el modelo Rasa NLU, útil para procesar y buscar la información necesaria.



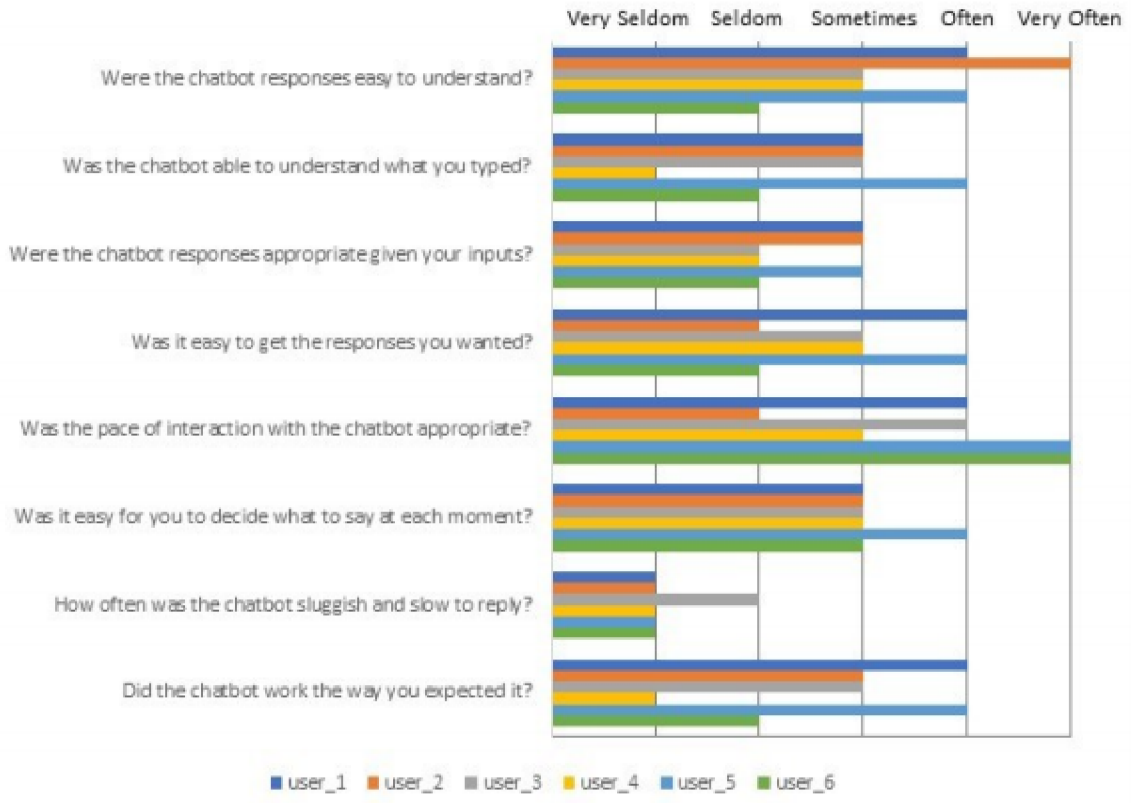
Cuadro 10.6: Arquitectura de "Chatbol" mostrando por dentro Rasa NLU.

Rasa NLU utiliza distintos mecanismos y técnicas de procesamiento explicadas anteriormente sobre la pregunta, para luego buscar una posible respuesta en Wikipedia. Para la búsqueda se utiliza SparQL, un lenguaje de consulta. Si no se encuentra una respuesta válida, busca en un corpus de conversaciones de OpenSubtitles para darle una respuesta amigable al usuario. Solo les sirvieron 94 archivos de subtítulos (Debían contener la palabra fútbol) así que incorporaron sesiones de chats del canal #futbol de IRC-HISPANO (Slack).

Es de interés el análisis de resultados ya que también se enfocan en el trato con el usuario, estos resultados se presentan en los Cuadros 10.7 y 10.8.

System	Responses		
	Valid	Acceptable	Invalid
QA model	34 (53%)	12 (19%)	18 (28%)
Retrieval model	50 (43%)	28 (25%)	37 (32%)

Cuadro 10.7: Evaluación de las respuestas generadas por “Chatbol” para cada modelo.



Cuadro 10.8: Evaluación de “Chatbol” con perspectiva de los usuarios finales.

10.2 README

A continuación se presenta el README del proyecto, encontrado en el repositorio.

Este es un proyecto de grado realizado por estudiantes de la facultad de ingeniería UdelaR. Propone la aplicación de técnicas de Procesamiento de Lenguaje Natural (PLN) para la construcción de una herramienta que de soporte a un sistema de búsqueda de información.

Dependencias

Para su correcto funcionamiento, se debe tener instalada la versión **7.9.2** de Elasticsearch local en el puerto default, 9200. Python debe encontrarse instalado en su versión 3.8. También se debe tener Pipenv, para facilitar el manejo de los distintos environments empleados.

Scraping

Luego se debe popular la base de datos con los datos encontrados en la web y las preguntas similares encontradas en el **tsv**. Para esto se levanta el env ubicado en la carpeta de Scraping y se instalan las dependencias.

```
$ cd tesis-inefop/Docker/Scrapper_pipenv/
$ pipenv shell
$ pipenv install
```

Luego se debe correr el script principal, este puede tardar un tiempo considerable.

```
$ python main.py
```

Back-end

Para levantar el back-end, se levanta el env y se instalan las dependencias de la misma manera.

```
$ cd tesis-inefop/Docker/QSimilarity_sv//
$ pipenv shell
$ pipenv install
```

El script principal deja levantado el servidor de back end.

```
$ python run_server.py
```

Front-end

Para levantar la página web se utiliza un comando tanto para hacer el build como para hostearla.

```
$ ng serve
```