UNIVERSIDAD TECNOLÓGICA NACIONAL

Facultad Regional Rosario

Maestría en Ingeniería en Sistemas de Información

Tesis de Maestría

"Desarrollo de una medida de similaridad para Sistemas de Recomendación en sitios de Community Question Answering. Análisis desde un enfoque Big Data y usando un método de ensamble de clustering"

Ing. Federico Tesone

Director: Dr. Guillermo Leale

Co-director: Dra. Soledad Ayala

Rosario, Santa Fe, Argentina. Julio de 2021

A mi persona favorita.

Resumen

Los *sistemas de recomendación* (Recommender Systems o RS) tienen la tarea de recomendar ítems a los usuarios de un sitio o aplicación. Los mismos pueden ser aplicados a sitios de preguntas y respuestas colaborativas, llamados *Community Question Answering* (CQA por sus siglas en inglés) y las preguntas que realizan los usuarios de la aplicación pueden considerarse como los ítems a recomendar. En este trabajo, son de interés las preguntas pendientes de ser respondidas, ya que la tarea de recomendar otras preguntas similares que hayan sido formuladas por otros usuarios y tengan la respuesta deseada, puede ser realizada por un RS, minimizando así el tiempo en que un usuario puede encontrar lo que estaba buscando.

Un buen RS debería utilizar una medida de *similaridad* confiable entre preguntas, por lo cual proponemos crear una nueva medida combinada de distancia para textos a través de un método de ensamble de clustering basado en acumulación de evidencias, utilizando una arquitectura Big Data. Para este fin, dispondremos de un conjunto de datos de pares de preguntas reales, extraídos del sitio web Quora. Se realizará un análisis comparativo entre el método de ensamble de clustering y las medidas de similaridad utilizadas como punto de partida del mismo.

Este tipo de enfoque es necesario para trabajar con grandes conjuntos de datos y así recuperar, analizar y procesar los mismos con precisión, variabilidad y velocidad, con el propósito de encontrar una medida de similaridad que pueda presentarse como una alternativa a las actuales en términos de mejorar la experiencia del usuario en sitios de CQA, mejorar las medidas de rendimiento y reducir las probabilidades de error en la búsqueda de preguntas similares.

Palabras clave: Community Question Answering, Recommender Systems, Big Data, Ensemble Clustering, Evidence accumulation.

Índice General

Índice de Tablas

Índice de Figuras

Agradecimientos

Capítulo 1. **Introducción**

1.1. Área temática

Los sitios de *Community Question Answering* (CQA) brindan servicios que permiten a los usuarios formular y contestar preguntas sobre temas de cualquier índole. Miles de nuevas preguntas son subidas diariamente en sitios de CQA como Yahoo! Answers[1] , Stackexchange[2] , Stackoverflow[3] , o Quora[4] . Estos son portales muy populares donde los usuarios suben diariamente una cantidad importante de preguntas de varios dominios para obtener respuestas de otros usuarios de la comunidad (Anuyah et al., 2017). Del análisis de sitios de CQA, puede observarse que muchas de las preguntas no están respondidas correctamente o no tienen respuestas específicas, ya que, en estas comunidades, hay típicamente un pequeño número de expertos entre la gran población de usuarios (Yang et al., 2013). Por lo tanto, cuando un usuario realiza una pregunta, es de interés buscar si esa misma interrogación ha sido formulada por otro usuario con anterioridad y para verificar que tenga la respuesta buscada. Con esto, el usuario podría leer las respuestas a dicha pregunta sin tener que esperar que la misma sea respondida. Esto no siempre es una tarea fácil, ya que podría darse la situación en la que la pregunta sí exista previamente en el sitio y haya sido respondida y, sin embargo, esté formulada de una manera completamente diferente en el sentido léxico. Por esta razón, una correspondencia exacta (o casi exacta) no es aplicable. Consideremos el siguiente ejemplo, a partir del análisis de dos preguntas cuya respuesta final es la misma: *¿Cómo elijo una revista para publicar mi artículo?* y *¿Dónde publico*

[1] Yahoo! Answers: `https://answers.yahoo.com/`. Último acceso: Julio 2021.
[2] Stackexchange: `https://stackexchange.com/`. Último acceso: Julio 2021.
[3] Stackoverflow: `https://stackoverflow.com/`. Último acceso: Julio 2021.
[4] Quora: `https://www.quora.com/`. Último acceso: Julio 2021.

mi artículo?[5] . Entre estas dos interrogaciones, existe apenas una superposición de palabras, sin tener en cuenta los *stopwords*[6] . Sin embargo, ambas preguntas tienen la misma respuesta, que referirá a revistas o sitios donde publicar un artículo científico.

Con el fin de comparar dos preguntas, se establece entonces una medida de similaridad que se puede considerar como máxima cuando son idénticas y que es inversamente proporcional a las diferencias entre ellas (Lin et al., 1998). Pero teniendo en cuenta que una medida de similaridad de texto entre preguntas basada en características léxicas no las detectaría como preguntas iguales. Esto deja en evidencia la necesidad de utilizar enfoques que, además, consideren características semánticas.

A partir de lo expuesto anteriormente, puede decirse que la tarea de encontrar preguntas similares en sitios de CQA puede ser llevada a cabo por un Sistema de Recomendación. Los Sistemas de Recomendación (Recommender Systems o RS) son herramientas de software y técnicas que proveen sugerencias de ítems, bajo el supuesto de que es posible que los usuarios tengan la intención de utilizar dichos ítems (Ricci et al., 2011). Las sugerencias relacionan una variedad de procesos de toma de decisiones, como por ejemplo qué artículos comprar o qué música escuchar. El término general usado para denotar lo que los RS recomiendan a los usuarios es el vocablo "*Ítem*". Los ítems son objetos que pueden estar caracterizados por su valor o utilidad. El valor de un ítem puede ser positivo si el ítem es útil para el usuario y negativo si no es apropiado. En este último caso, el usuario tomaría una mala decisión al seleccionarlo. A partir de la dinámica que se construye en los RS, las recomendaciones al usuario pueden ser personalizadas o no personalizadas. Las primeras, se basan en los comportamientos del usuario o en grupos de usuarios para encontrar sugerencias adecuadas a sus preferencias; las segundas, efectúan recomendaciones que son inherentes a los ítems que el RS sugerirá. Cada una de estas estrategias de recomendación se elabora a partir de los diferentes conocimientos y datos recopilados por el sitio o el sistema donde el RS esté aplicado. Algunos ejemplos de tales aplicaciones incluyen la recomendación

[5] Traducción de las preguntas "How do I choose a journal to publish my paper?, Where do I publish my paper?" extraídas desde el conjunto de datos de Quora que se utilizará en el presente trabajo de tesis `https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs`. Último acceso: Julio 2021.

[6] En informática, se llama stopword a palabras que se filtran antes o después del procesamiento de datos del lenguaje natural (Leskovec et al., 2014).

de libros, películas, o ítems de compra de productos y/o servicios (Adomavicius y Tuzhilin, 2005). En particular, para los sitios de CQA, los algoritmos de recomendación se aplican principalmente a elementos de texto. Este trabajo se centrará en ese tipo de recomendaciones, que pueden estar clasificadas dentro de RS basados en contenido de texto no personalizados, ya que las mismas están basadas únicamente en la estructura sintáctica y semántica de las preguntas existentes en los sitios de CQA. Los usuarios pueden navegar esas recomendaciones. Luego, pueden aceptarlas o no y, además, proveer inmediatamente o en un paso posterior, una retroalimentación explícita o implícita.

A consecuencia de lo expuesto anteriormente, y sumado a que se dispone de un gran conjunto de datos, se diseñó e implementó una propuesta de arquitectura para utilizar Big Data con el fin de crear una medida de similaridad de texto que alimente a un RS especializado en la tarea de encontrar preguntas similares en sitios de CQA basado en análisis de contenido de texto. Este tipo de enfoque es necesario para procesar una gran cantidad de datos y, de esta manera, optimizar el procesamiento de los mismos, logrando velocidad y con la ventaja de poder aprovechar toda la variabilidad que provee un conjunto de datos de gran volumen. Luego de obtener esta nueva medida de similaridad, se realizará un análisis comparativo de la misma contra las medidas subyacentes utilizadas como entrada del método en cuestión, utilizando la arquitectura en Big Data propuesta.

1.2. Tema específico

Con el fin de dejar en claro el alcance de este trabajo, se toma como punto de partida el trabajo de investigación de la Universidad Tecnológica Nacional, Facultad Regional Rosario: "Comparative Analysis on Text Distance Measures Applied to Community Question Answering Data" (Gonzalez et al., 2017), el cual se centra en el Paso 1 de la secuencia de pasos (o *pipeline*) que se describe en la Figura 1.

Este proceso descrito en tres pasos, tiene como objetivo construir un RS basado en una medida novedosa de similaridad de texto. En el Paso 1 se realiza un análisis comparativo desarrollado a partir de medidas basadas en distancia; en el Paso 2, se crea una nueva medida construyendo una matriz de similaridad basada en ensamble de clustering, como una de las propuestas en los métodos

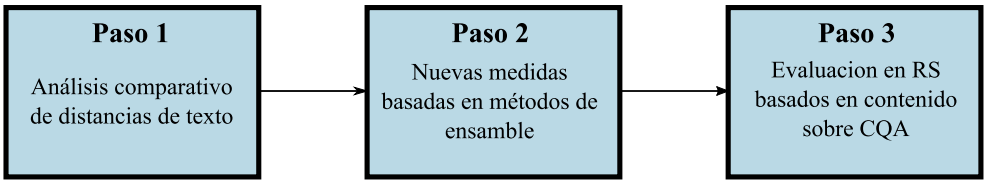

Figura 1: Pipeline para un RS basado en contenido de CQA y en una nueva medida de similaridad.

Algoritmo de Particionamiento de Similitud basado en Cluster (Cluster-based Similarity Partitioning Algorithm o CSPA) (Strehl y Chosh, 2002) y *Clustering de Acumulación de Evidencias* (Evidence Accumulation Clustering o EAC) (Fred y Jain, 2005). El método EAC, que será descrito en detalle más adelante, intenta mejorar la calidad de salida para una representación de similaridad basada en texto; por último, en el Paso 3 se debe aplicar la matriz de similaridad obtenida en el Paso 2 en un RS basado en contenido, con el fin de evaluar su eficacia en sitios de CQA.

Si tomáramos el conjunto completo de datos Quora (404301 pares de preguntas, es decir, 808602 preguntas totales), y quisiéramos generar una sola matriz de distancias cruzando par a par todas las preguntas entre sí, estaríamos calculando $\frac{n(n+1)}{2} = 326919001503$ distancias, donde $n = 808602$ y el resultado es la cantidad de elementos en la triangular superior de la matriz cuadrada donde las preguntas corresponden a la vez a las filas y a las columnas. El ejemplo anterior sirve para tomar dimensión del volumen de datos que es necesario manejar con este enfoque basado en ensamble de clustering. Esta matriz considera sólo una de las técnicas de distancia de texto del estado del arte, por lo cual si deseamos combinar varias técnicas mediante un método de ensamble, deberíamos generar al menos una matriz por cada una de ellas, y luego usar las mismas para aplicar EAC. Por lo anterior, estaríamos generando un número total de cálculos considerablemente mayor al que puede procesar una computadora clásica, en un tiempo aceptable. Es necesario además tener en cuenta que las distancias pueden llegar a considerar características diversas entre sí, como morfología, sintaxis y semántica de los textos, lo cual añade complejidad y variedad al volumen considerado. Esta situación plantea la necesidad de considerar una arquitectura Big Data, conjuntamente con optimización de código y técnicas de ejecución paralela entre un gran número de tareas de procesamiento. Con respecto al espacio de almacenamiento, debe notarse que las matrices requieren doble precisión para la representación interna de cada uno de sus elementos (distancias), es decir 750 KB cuando están

almacenados en un archivo, por lo cual, si consideramos la matriz ejemplificada anteriormente, necesitaríamos aproximadamente 12 TB para almacenarla, y sería es necesario un esquema de almacenamiento optimizado para la implementación de este método. Con respecto a la velocidad de procesamiento, pruebas preliminares en las técnicas del estado del arte, realizadas con un procesador potente, brindan una estimación de alrededor de 3 años para completar el procesamiento. Con el problema planteado de esta forma, es indispensable aplicar un enfoque de Big Data para satisfacer los requerimientos de volumen, variedad y velocidad que requiere el contexto de análisis, de tal forma de brindar resultados veraces, y de esa forma cumplir con las premisas de las "V" del Big Data[7] .

Este trabajo de tesis apunta entonces a construir una medida de similaridad novedosa desde un enfoque Big Data, tal como se describe en el Paso 2 del pipeline, para luego compararla con las técnicas existentes. Para tal fin, se crea un nuevo software basado en una arquitectura y patrones de Big Data, tomando como punto de partida el desarrollo del estado del arte a partir del trabajo mencionado.

1.3. Objetivo general

El presente trabajo de investigación tiene como objetivo construir una arquitectura Big Data que incluye la posibilidad de ser aplicada a grandes conjuntos de datos en el ámbito de CQA y, a partir de esta arquitectura, implementar y evaluar nuevas medidas de similaridad entre textos que puedan ser utilizadas en sistemas de recomendación.

1.4. Objetivos específicos

Se detallan a continuación, los objetivos específicos que son necesarios para lograr el objetivo principal.

- Diseñar y desarrollar una arquitectura Big Data para cálculo de similaridad en grandes matrices, que requerirá nuevas estrategias para recolectar, procesar y manejar grandes volúmenes de datos.

[7] Las "V" del Big Data refieren a Volumen, Variedad y Velocidad. También se consideran los conceptos de Valor y Veracidad con respecto al resultado de la aplicación del enfoque Big Data (Gandomi y Haider, 2015).

- Identificar medidas de similaridad de texto existentes y un método efectivo de aplicación de las mismas en grandes volúmenes de datos.

- Evaluar el comportamiento de medidas de similaridad de texto del estado del arte respecto al manejo del volumen, variedad, velocidad y veracidad inherentes a grandes volúmenes de datos, en particular en el ámbito de CQA.

- Proponer una nueva medida que permita integrar las medidas de similaridad del estado del arte mediante una arquitectura de software basada en Big Data y que sea extensible a otras medidas existentes en el estado del arte.

- Brindar conclusiones, pautas y recomendaciones para trabajar con medidas de comparación de textos en grandes volúmenes de datos utilizando arquitecturas basadas en Big Data y su aplicación en sistemas de recomendación.

Capítulo 2. **Fundamentación**

2.1. Motivación de la tesis

La calidad de un RS tiene una relación directa con los datos de entrada que se han generado para alimentarlo. Con el fin de generar una entrada basada en medidas de similaridad, es necesaria la comparación de preguntas formuladas en sitios de CQA usando técnicas de análisis de texto. Un problema importante inherente al análisis de texto, con el fin de cuantificar relaciones entre distintos fragmentos o documentos, es encontrar la medida apropiada de representación (Gonzalez et al., 2017). Algunas medidas de similaridad resultantes de algoritmos de recomendación en análisis de texto son obtenidas mediante algoritmos puramente sintácticos, léxicos, tales como: Term Frequency (Salton y McGill, 1983), Term Frequency/Inverse Document Frequency (Baeza-Yates et al., 1999), basados en ventanas como FastText (Joulin et al., 2016) o Word2Vec (Mikolov et al., 2013), o semánticos, como Semantic Distance (Li et al., 2006). Los algoritmos puramente sintácticos como Term Frequency y Term Frequency/Inverse Document Frequency tienen conocidos problemas, tales como ser invariantes respecto al orden de las palabras o ser sensibles a stopwords, por lo cual, necesitan un gran trabajo de pre-procesamiento. Adicionalmente, no tienen en cuenta la semántica de las palabras y sus relaciones. FastText y Word2Vec están fuertemente afectados en el orden en el cual aparecen las palabras. Semantic Distance, por su lado, según el trabajo tomado como estado del arte, tampoco alcanza medidas de rendimiento apropiadas para un RS en un sitio de CQA, por ejemplo posee un pobre desempeño en términos de eficiencia debido a su complejidad inherente.

Los resultados experimentales de medidas de rendimiento obtenidas en el trabajo que se toma como punto de partida de esta tesis, arrojan entre un 66,85 % y un 67,97 % de exactitud (entre un 32,03 % y un 33,15 % de error) usando cada uno de los algoritmos de recomendación descritos anteriormente. Estos valores son considerados prometedores, ya que las medidas de rendimiento son consis-

tentes en todos los algoritmos seleccionados, lo que denota que la complejidad inherente del conjunto de datos no afecta significativamente el rendimiento de cada uno de ellos. Además, los resultados de prueba no varían significativamente con respecto a los resultados de validación. Dicho esto, una de las motivaciones de este trabajo de tesis es la creación de un método novedoso que combine medidas de similaridad existentes, que puede además aplicarse como entrada para un RS con el fin de ser implementado en sitios de CQA. Conjuntamente con esta motivación, se propone una arquitectura de software que soporte el procesamiento del método propuesto de una forma eficiente y escalable. Para tal fin, se crearon matrices de distancias (o similaridad), usando cada una de las preguntas de un conjunto de datos que se tomó como objeto de estudio, para luego combinarlas usando métodos de ensamble de clustering. La propuesta de esta combinación surge a raíz del supuesto de que, como existen múltiples algoritmos de clustering, es difícil identificar uno solo de estos algoritmos que pueda manejar todos los tipos de formas y tamaños de cluster, e incluso decidir qué algoritmo sería el mejor para un conjunto de datos en particular.

Los autores en (Fred y Jain, 2005) introducen el concepto de Clustering de Acumulación de Evidencias (EAC), que mapea las particiones de datos individuales en un ensamble de clustering dentro de una nueva medida de similaridad entre elementos, resumiendo la estructura entre elementos percibido a partir de esos clusters. La partición de datos final es obtenida aplicando el método de clustering *single-linkage* a la nueva matriz de similaridad. El resultado de este método muestra que la combinación de algoritmos de clustering "débiles", tales como el *k-means*, podrían conducir a una mejor identificación de los clusters subyacentes verdaderos, teniendo en cuenta formas, tamaños y densidades arbitrarias. Por lo cual, teniendo en cuenta diferentes particiones creadas con el método de ensamble desde los mismos datos originales, objetos de textos similares probablemente pertenecerán al mismo cluster.

El desarrollo de matrices de similaridad para la aplicación del EAC que se utilizarán en este trabajo como entrada de RS, claramente implica manipular un gran volumen de datos complejos y realizar un elevado número de cálculos en tiempo real, ya que nos estamos refiriendo a conjuntos de datos cuyo tamaño supera la capacidad de las herramientas tradicionales de bases de datos en las tareas de recopilar, almacenar, gestionar y analizar la información (De Battista et al., 2016). Esto implica, en principio, considerar una *matriz de co-asociación*

entre elementos realizando varias series de corridas y aplicación de clustering. Cada una de esas series está basada en una de las medidas de similaridad analizadas en este trabajo. El resultado será un valor adimensional e insesgado que puede mejorar la representación para la estructura subyacente de relaciones de texto. El volumen de datos ejemplificado en las secciones anteriores, deja expuesta la necesidad de investigar y desarrollar el tema aquí propuesto con un enfoque distinto al tradicional. Esto implica realizar un muestreo aleatorio de pares de preguntas dentro de una arquitectura que permita generar la mayor cantidad posible de subconjuntos de datos extraídos aleatoriamente. Además, posibilitará que cada uno de ellos sea lo más grande posible para aprovechar toda la *variedad* de los datos. Mientras más se aproveche la variedad de los datos (más subconjuntos de datos y de mayor tamaño), se afectará negativamente en el tiempo de procesamiento, razones por las cuales se hace necesaria una arquitectura e infraestructura preparada para tal desafío, con una velocidad que haga posible obtener resultados en un período de tiempo razonablemente corto. Un enfoque Big Data es imprescindible para este tipo de procesamiento de datos. No solo si se desea hacer referencia a la gran cantidad y complejidad de los datos, sino también a las herramientas utilizadas para procesarlos y las posibilidades de extraer conocimiento útil a partir del análisis de los mismos. Estos procesos y herramientas son el eje central de la definición original de Big Data de la consultora Gartner (2012), la cual hace foco en los procesos para manipular activos de gran volumen y variedad con una gran velocidad. Por lo cual, si bien Big Data se refiere a estos activos, demanda formas innovadoras y efectivas de procesarlos que habiliten tomas de decisiones y automatización de procesos.

Por todos estos motivos, se propone la elaboración de una arquitectura que soporte un nuevo método, que genere una entrada de datos correctamente estructurada para RS y que pueda ser utilizada en sitios de CQA de una forma eficiente y eficaz, además del análisis comparativo de dicho método con otros métodos seleccionados del estado del arte.

2.2. Importancia científico-tecnológica

Con respecto a los sitios de CQA en particular, la importancia de este trabajo radica en la posibilidad de ofrecer herramientas para construir un RS que, desde el punto de vista del usuario, sirva tanto para reducir el tiempo promedio en que se encuentra una respuesta como para mejorar la experiencia del sitio. En este

sentido, en la mayoría de los casos no será necesario escribir múltiples versiones de la misma pregunta y los lectores podrán encontrar rápidamente la respuesta que están buscando. Por otro lado, se podría evitar, si se desea, que se creen preguntas duplicadas, lo que significaría un aumento considerable en la calidad y cantidad de la base de conocimiento del sitio, construyendo una relación biunívoca entre una pregunta y su correspondiente respuesta. Además, se logrará optimizar el tamaño de la base de datos, la integridad de la información, mejorar la velocidad en búsquedas e incrementar la satisfacción y fidelidad del usuario (Ricci et al., 2011).

Por otro lado, con respecto a RS en general, en este trabajo se transitan desafíos que tienen que ver con la extracción, transformación y carga de datos, y con la aplicación de distintos modelos y algoritmos de comparación de texto y búsqueda de información para Sistemas de Recomendación. Además, se explorarán diferentes soluciones posibles en cuanto a la arquitectura y diseño de un RS en tiempo real, que servirán para futuras investigaciones y como bibliografía para desarrolladores e investigadores del área.

Adicionalmente, como extensión futura de este trabajo, el resultado de la presente investigación también puede ser utilizado para sitios que son fuente de consulta para diversos investigadores dentro del ámbito de la Universidad Tecnológica Nacional, Facultad Regional Rosario, tales como bibliotecas virtuales o foros de consulta para investigaciones científicos-tecnológicos que incluyan I+D+i. Esto permitiría no solo conocer los intereses de otros investigadores y en qué términos formularon sus interrogaciones, sino también conocer quién o quiénes elaboraron las respuestas a dichas preguntas y a qué campo disciplinar pertenecen.

2.3. Formación de recursos humanos

El presente trabajo de tesis, en relación a la formación de recursos humanos, tiene los siguientes objetivos:

- Capacitar a un grupo de estudiantes de la UTN FRRo, en el Departamento Ingeniería en Sistemas de Información, con elementos para la investigación y desarrollo en aplicaciones Big Data.
- Realizar grupalmente conocimiento científico, con base teórica sustentable y ejemplos empíricos de aplicaciones funcionales, para presentar en con-

gresos tales como AGRANDA[8] , CONAIISI[9] , o RecSys[10] ; o bien eventos relacionados con Ingeniería en Sistemas de Información.

- Elaborar material de estudio relacionado con la temática de la minería de datos para materias de grado, tales como la electiva Minería de Datos, en quinto año de la carrera de Ingeniería en Sistemas de Información en UTN FRRo, y/o posgrado, tales como la Maestría y Especialización en Ingeniería en Sistemas de Información de dicha Facultad.

- Lograr que los estudiantes puedan entender cómo está formado en la actualidad el estado del arte sobre el presente tema y que esto sirva de base para futuras investigaciones en UTN FRRo, ya sean proyectos de investigación, tesis de maestría o de doctorado, mediante su difusión al cuerpo de docentes, alumnos y graduados con vínculo al Departamento Ingeniería en Sistemas de Información de la UTN FRRo.

- Desarrollar insumos para el armado de cursos tanto de formación académica como abiertos a la comunidad relacionados con Big Data o análisis de texto.

2.4. Importancia socio-económica

El tema posee una importancia social y económica que permitirá construir contactos y alianzas -económicas, académicas y de naturaleza mixta- con instituciones nacionales y extranjeras. En otras palabras, a nivel social podría utilizarse para actividades de investigación y en los diferentes niveles educativos, según se adecúen tanto las explicaciones y el vocabulario utilizado, como las actividades y los diversos usos. La búsqueda de información en bibliotecas digitales y virtuales, en bases de datos científicos, repositorios digitales, foros especializados de temáticas específicas y diversas o plataformas educativas, son algunos de los sitios donde los RS pueden ser utilizados y aplicados para determinadas actividades cognitivas. Además, el tema puede ser complementado en un futuro con otras líneas de investigación, tales como políticas educativas para la alfabetización mediática, análisis y datos online, fuentes abiertas o la relación entre tecnología y democracia. Estas líneas, de prioridad en la agenda de ciencia y tecnología de países del primer mundo, están siendo desarrolladas entre academia, instituciones de gubernamentales y policy-makers, de manera interdisciplinaria y con el

[8] AGRANDA: Simposio Argentino de GRANdes DAtos.
[9] CONAIISI: Congreso Nacional de Ingeniería Informática - Sistemas de Información.
[10] RecSys: The ACM Conference Series on Recommender Systems.

objetivo de mejorar las herramientas que poseen los ciudadanos en relación a la cultura digital y sus mecanismos de funcionamiento estrictamente técnicos y los aspectos culturales que la atraviesan.

Por otro lado, en el marco económico, los resultados de la presente investigación posibilitarán continuar con futuras indagaciones referidas a la temática y diseñar/construir nuevas herramientas de software adecuadas en función de ciertos usos y usuarios específicos, especialmente en contextos de Big Data. Estas acciones permitirían llevar adelante: nuevos proyectos de investigación interdisciplinarios y con subsidios de naturaleza mixta (público-privada), formación de formadores, pequeños emprendimientos para estudiantes avanzados y/o la postulación a becas de formación (nacionales e internacionales).

Capítulo 3. **Marco teórico**

En este capítulo se explicarán los conceptos utilizados en el presente trabajo de tesis, estructurados en cinco grandes aristas: sitios de CQA, Sistemas de Recomendación, Big Data, medidas de similaridad y Clustering. Todos estos conceptos se combinarán para luego, en el siguiente capítulo, abordar el problema de investigacion.

3.1. Sitios de CQA

Los servicios de Community Question Answering CQA, son un tipo especial de servicios de *Question Answering* (QA), los cuales permiten a los usuarios registrados responder a preguntas formuladas por otras personas. Los mismos atrajeron a un número creciente de usuarios en los últimos años (Li y King, 2010). Una pregunta formulada en el sitio Web Quora, y respondida por su fundador y CEO, Adam DÁngelo, revela que el sitio recibe más de 200 millones de visitantes únicos mensualmente (información actualizada a Junio de 2017), lo que denota la popularidad de este tipo de portales[11] . Desde la creación de este tipo de servicios, se han aplicado diferentes técnicas y herramientas de software para que los usuarios encuentren respuestas a sus preguntas en el menor tiempo posible y aprovechar al máximo el valor de las bases de conocimiento. Algunas de estas herramientas son: un framework para predecir la calidad de las respuestas con características no textuales (Jeon et al., 2006); la propuesta de incorporar información de legibilidad en el proceso de recomendación (Anuyah et al., 2017); una forma de encontrar a los expertos apropiados (Li y King, 2010); o bien una recomendación para la mejor respuesta a una pregunta dada, entre otros. Sin embargo, el mecanismo existente por el cual se responden las preguntas en los sitios de CQA todavía no alcanza a satisfacer las expectativas de los usuarios por varias razones: (I) baja probabilidad de encontrar al experto: una

[11] Pregunta formulada en el sitio Quora "How many people use Quora?": `https://www.quora.com/How-many-people-use-Quora-3`. Último acceso: Febrero 2021.

nueva pregunta, en muchos casos, puede no encontrar a la persona con la habilidad de responder de manera correcta, resultando en respuestas tardías y que distan de ser óptimas; (II) respuestas de baja calidad: los sitios de CQA suelen contener respuestas de baja calidad, maliciosas y spam. Estas suelen recibir baja calificación de los miembros de la comunidad; (III) preguntas archivadas y poco consultadas: muchas preguntas de los usuarios son similares. Por lo anterior, es posible pensar que antes de formular una pregunta, un usuario podría beneficiarse a partir de buscar otras preguntas ya formuladas, y con ello, encontrar sus respuestas a partir de las existentes correspondientes a dichas preguntas (Yang et al., 2013).

3.2. Sistemas de recomendación

3.2.1. Contexto Histórico

Es muy frecuente tener que tomar decisiones sin la suficiente experiencia personal sobre las alternativas disponibles. En la vida cotidiana, confiamos en recomendaciones de otras personas, ya sea de boca en boca o cartas de recomendación, reseñas de libros y películas, o encuestas generales. Los sistemas de recomendación o RS asisten este proceso natural en el ámbito de los sistemas de información (Resnick y Varian, 1997). El primer RS, Tapestry (Goldberg et al., 1992), fue un sistema experimental destinado a resolver el problema de manejar grandes cantidades de correos electrónicos filtrando según cuán interesantes eran los documentos, utilizando un enfoque *basado en el contenido* de los mismos y también *filtros colaborativos*. Se ha trabajado mucho en mejorar y desarrollar nuevos enfoques con respecto a los RS en los últimos años, y el interés en esta área sigue vigente debido a la abundancia de aplicaciones prácticas en las cuales es necesario ayudar a los usuarios a lidiar con la sobrecarga de información[12] y proveer para este fin recomendaciones personalizadas, contenidos, y servicios. Sin embargo, a pesar de todos estos avances, la generación actual de RS todavía requiere mejoras para que los métodos de recomendación sean más efectivos y aplicables a una gama más amplia de sistemas y/o sitios. Aunque las raíces de los RS se remontan a trabajos en ciencia cognitiva (Rich, 1979), teoría de aproximación (Powell, 1981), recuperación de información (Salton, 1989), ciencias de las

[12] El concepto de sobrecarga de información, del inglés information overload, hace referencia a cuando los usuarios reciben demasiada información, por lo cual, la precisión en sus decisiones empieza a decrecer (Eppler y Mengis, 2004).

predicciones (Armstrong, 2001), ciencias de la gestión (Murthi y Sarkar, 2003), y también al modelado de la elección de consumidor en marketing (Lilien et al., 1992), los RS recién surgen como un área de investigación independiente en la década de 1990, cuando los investigadores comenzaron a centrarse en problemas de recomendación que se basan específicamente en *calificaciones* (Adomavicius y Tuzhilin, 2005). En su formulación más común, el problema de recomendación se reduce a estimar calificaciones para los ítems que no han sido vistos por un usuario.

3.2.2. Funciones de un Sistema de Recomendación

Como se mencionó anteriormente, un RS es un conjunto de herramientas de software que sugiere ítems a un usuario, quien posiblemente utilizará algunos de ellos. Haciendo énfasis particularmente en un RS comercial, probablemente la función más importante es incrementar el número de ítems vendidos, lo cual es posible porque el RS ofrecerá los ítems sobre los cuales el usuario tiene más probabilidades de deseo o necesidad. Esto implica aumentar el denominado *ratio de conversión*, es decir, la cantidad de ventas que es posible efectuar sobre un ítem sobre el total de veces que un usuario selecciona el mismo. Por ejemplo, para un sitio de delivery de comida online, esta medida corresponde a cuántas veces un usuario realiza un pedido en un restaurante en particular, sobre el total de veces que observó el menú del mismo. Indudablemente, la conversión va a ser mayor si el usuario recibe recomendaciones de restaurantes que están más cercanos a su gusto personal. Otra función de un RS comercial complementaria y muy relacionada a la anterior es vender productos más diversos, ya que sería muy difícil para un usuario encontrarlos sin una recomendación precisa.

Desde el punto de vista del usuario, un conjunto de recomendaciones precisas y relevantes aumentarán su satisfacción y fidelidad. El usuario disfrutará el uso de un sistema donde cada ítem o característica que utiliza está diseñada teniendo en cuenta sus intereses. Aumentar la fidelidad del usuario con el sitio significa que habrá mucha más interacción y, por lo tanto, el modelo de recomendación se volverá más refinado. Este conocimiento, a su vez, puede ser utilizado para mejorar otros sistemas y procesos relacionados, como el sistema de control de stock o la publicidad (Ricci et al., 2011).

3.2.3. Técnicas de Recomendación

Para implementar su función principal, un RS debe *predecir* si vale la pena recomendar un ítem en particular. Para esto, este sistema debe ser capaz de predecir la utilidad de algunos de los ítems, o al menos poder comparar la utilidad entre algunos de ellos, y entonces decidir qué ítems recomendar basándose en esta comparación. Para realizar estas comparaciones, los RS basan sus estrategias de recomendaciones en 6 técnicas básicas (Ricci et al., 2011):

3.2.3.1. Basados en contenido

Los RS basados en contenido intentan recomendar ítems similares a los que el usuario eligió anteriormente. Como su nombre lo indica, el proceso básico llevado a cabo por estos RS consiste en hacer coincidir atributos del perfil de usuario que posean preferencias e intereses en la búsqueda actual con los atributos del ítem que se va a recomendar (Lops et al., 2011). Este tipo de RS es especialmente útil cuando se conocen características de los ítems a recomendar pero no se conocen características del usuario. En otras palabras, estos sistemas intentan recomendar ítems similares a los que el usuario ha elegido anteriormente.

Uno de los limitantes conocidos de estos RS es que recomiendan ítems del mismo tipo al que el usuario está solicitando. Por ejemplo, no sería posible recomendar música, utilizando videos ya que el perfil de contenido es distinto. Para solucionar esto, muchos de los RS basados en contenido están utilizando algoritmos híbridos con otro tipo de técnicas de recomendación.

3.2.3.2. Filtrado Colaborativo

El Filtrado Colaborativo (Collaborative Filtering en inglés o CF) es el proceso de filtrado o evaluación de ítems usando las opiniones de los demás (Schafer et al., 2007). El Filtrado Colaborativo es el enfoque original y el más simple de todas las técnicas de recomendación, y el más utilizado. Se basa en recomendar al usuario activo los ítems que otros usuarios con gustos similares eligieron en el pasado.

3.2.3.3. Demográficos

La mayoría de los RS utilizan enfoques basados en conocimiento o en contenido. Esto implica que se necesita la suficiente información o un conocimiento

adicional para poder llevar a cabo las recomendaciones. Los RS *demográficos* hacen recomendaciones basadas en clases demográficas, por ejemplo, una ciudad en común. La ventaja es que la información histórica no es necesaria. Por ejemplo, una aplicación para este enfoque podría ser la de utilizar información demográfica para predecir el rating de distintos turistas a atracciones, basándose en enfoques predictivos de *Machine Learning* (Wang et al., 2012). Las técnicas demográficas forman correlaciones "persona-a-persona", como los sistemas colaborativos, pero utilizando una naturaleza distinta para los datos, en este caso, el perfil demográfico del usuario.

3.2.3.4. Basados en conocimiento

Los RS *basados en conocimiento* (Knowledge-based en inglés) usan el conocimiento acerca de los propios usuarios y los ítems a recomendar para generar la recomendación, razonando acerca de qué ítems satisfacen los requerimientos del usuario (Burke, 2000).

Los RS de filtrado colaborativo, al utilizar datos de otros usuarios, deben ser inicializados con un conjunto de datos considerablemente grande, ya que un sistema con una base de datos pequeña es improbable que sea útil. Además, la precisión del sistema es muy sensible al número de ítems asociados con un usuario dado (Shardanand y Maes, 1995). Esto conlleva a un problema de inicialización: hasta que no exista un número considerable de usuarios cuyas elecciones y hábitos sean conocidos, el sistema no será útil para un nuevo usuario. Lo mismo sucede para los RS que toman enfoques de Machine Learning. Típicamente, este tipo de sistemas se convierten en buenos clasificadores una vez que han aprendido desde una gran base de datos. Los RS basados en conocimientos evitan estas desventajas. No existe un problema de inicialización, ya que las recomendaciones no dependen de un conjunto de datos grande. Para este tipo de RS, no es necesario recolectar información acerca de un usuario en particular porque las recomendaciones que realizan son exclusivamente basadas en las elecciones de ese usuario. Estas características no solo hacen a este tipo de RS muy valioso en sí mismo, sino también como complemento de otros RS que utilicen distintas técnicas.

3.2.3.5. Basados en comunidades

Un *RS basado en comunidades* (community-based en inglés) hace uso del método "boca a boca" digital para construir una comunidad de individuos que comparten opiniones personales y experiencias relacionadas con sus recomendaciones de ítems. Estos sistemas presentan y agregan opiniones generadas por los usuarios en un formato organizado, las cuales son consultadas a la hora de tomar decisiones (por ejemplo, comprar un producto) (Chen et al., 2009). La evidencia sugiere que las personas están más inclinadas para seguir una sugerencia de sus amigos que una sugerencia similar que viene desde una persona anónima (Sinha et al., 2001). Este tipo de RS toma importancia cuando se tiene en cuenta la creciente popularidad de las redes sociales abiertas, tal que estos sistemas también son conocidos como *Sistemas de Recomendación Sociales*.

3.2.3.6. Sistemas Híbridos

Una variedad de técnicas fueron propuestas como base de los sistemas de recomendaciones. Cada una de ellas tiene desventajas conocidas, como el ya mencionado problema de inicialización de los sistemas colaborativos y basados en contenido. Un *RS Híbrido* combina múltiples técnicas de recomendación para encontrar sinergia entre las mismas. Por ejemplo, un sistema basado en conocimiento puede compensar el problema de inicialización de los sistemas colaborativos para nuevos perfiles de usuario; así como también, el componente colaborativo puede utilizar sus habilidades estadísticas para encontrar pares de usuarios que compartan preferencias no esperadas, las cuales no podrían haber sido predichas por habilidades basadas en conocimiento (Burke, 2007).

3.3. Big Data y Arquitecturas

3.3.1. Contexto Histórico

Al igual que todos los términos que surgen a partir de avances tecnológicos, no existe un consenso claro de cómo definir *Big Data*. En (Manyika et al., 2011) los autores definen este concepto como los conjuntos de datos cuyo tamaño está más allá de la habilidad de las herramientas software de base de datos para capturar, almacenar, gestionar y analizar los datos. Nótese que esta definición no hace referencia a un tamaño mínimo del conjunto de datos, sino que asume que la tecnología avanza constantemente (como así también las herramientas), por

lo cual, la definición se "mueve" con el tiempo. Por otro lado, también es interesante tomar otra arista en la definición de este concepto: la consultora Gartner en su sitio web[13] lo define de la siguiente forma: "Big Data son activos de información caracterizados por su alto volumen, velocidad y variedad que demandan formas innovadoras y rentables de procesamiento de información para mejorar la compresión y la toma de decisiones", haciendo énfasis en la multiplicidad de características del concepto de Big Data.

El comienzo de sobrecarga de información, recientemente mencionado, data del año 1880, cuando el censo de los Estados Unidos tarda 8 años en organizarse y tabularse. Ante esta situación Herman Hollerith inventó la máquina tabuladora eléctrica basada en tarjetas perforadas[14] . El censo en 1890 fue un éxito rotundo e, incluso, la máquina diseñada fue utilizada para los censos de Canadá, Noruega y Austria al año siguiente. En el año 1941, los científicos empiezan a utilizar el término "explosión de la información", que fuera citado en el periódico The Lawton Constitution[15] , haciendo alusión a la dificultad de administrar toda la información disponible. Gradualmente, se identificaron avances concretos en materia de procesamiento de datos y criptografía, motivados particularmente por los sucesos bélicos de la época. Un ejemplo es el dispositivo llamado Colossus (Copeland, 2004) que buscaba e interceptaba mensajes a una tasa de miles de caracteres por segundo. Unos años más tarde, en 1951, el concepto de *memoria virtual* es introducido por el físico alemán Fritz-Rudolf Güntsch, como una idea que trataba el almacenamiento finito como infinito.

A partir de la década del 80', los avances tecnológicos, especialmente en sistemas MRP (planificación de recursos de fabricación), permitieron nuevas formas de organizar, almacenar y generar datos. En este sentido, IBM se destaca y define una arquitectura para los informes y análisis de negocio (EBIS)[16] , que se convierte en la base del almacenamiento de datos en forma centralizada para usuarios finales (Devlin y Murphy, 1988); es decir, el *data warehousing*. Hacia finales de los 80', Tim Berners-Lee, inventa la *World Wide Web* (Berners-Lee y Cailliau, 1992), invento que provocaría el impacto más grande hasta la actua-

[13] Concepto de Big Data en el glosario de Gartner: `https://www.gartner.com/en/information-technology/glossary/big-data`. Último acceso: Febrero 2021.

[14] Herman Hollerith, US Census Boureau: `https://www.census.gov/history/www/census_then_now/notable_alumni/herman_hollerith.html`. Último acceso: Febrero 2021.

[15] The Lawton Constitution: `http://www.swoknews.com/`. Último acceso: Febrero 2021.

[16] Acrónimo para EMEA (Europe, Middle East and Africa) Business Information System.

lidad con respecto a la generación, identificación, almacenamiento y análisis de grandes volúmenes de datos de diversa naturaleza.

El inicio de los años 90' marca un antes y un después en lo relativo al tratamiento y almacenamiento de datos. El crecimiento tecnológico fue explosivo, tal es así que el almacenamiento digital empieza a ser más conveniente y rentable que el papel para almacenar datos (Morris y Truskowski, 2003). Es en 1990 cuando surgen las plataformas de *Business Intelligence* (BI) y los rediseños de software al estilo *Enterprise Resource Planning* (ERP). En este contexto, en (Cox y Ellsworth, 1997) se afirma que el crecimiento de la cantidad de datos que debe manejar un sistema de información empieza a ser un problema en materia de almacenamiento y visualización de los datos, situación que denominaron como "el problema del Big Data". Así, 1997 es un año clave, en el que se realiza una gran cantidad de estudios y publicaciones que se enfocan en averiguar cuánta información hay disponible a nivel mundial y su crecimiento[17] y, en consecuencia, se estima que el crecimiento de Internet es aproximadamente del 100 % anual, y que superaría al tráfico de voz para el año 2002 (Coffman y Odlyzko, 1998).

En el año 2001, se introduce el concepto de *Las 3 V's: Volumen, Velocidad y Variabilidad de los datos* (Laney, 2001), fundantes sobre la temática y que sería mundialmente aceptado una década más tarde. Por otro lado, también en 2001, aparece el concepto de *Software como un Servicio* (SaaS) (Hoch et al., 2001), un modelo disruptivo de servicios centralizados y acceso a los mismos mediante clientes finos (típicamente exploradores web), dando la posibilidad del escalamiento horizontal de sistemas de información y la generación de estándares de comunicación. Esta situación provocó que empresas como Oracle[18] , SAP[19] y Peoplesoft[20] empiecen a centrarse en el uso de servicios web, permitiendo así la generación de datos en forma masiva por usuarios finales. Así, en 2006, nace Apache Hadoop[21] , una solución de código abierto que permite el procesamiento en paralelo y distribuido de enormes cantidades de datos en forma escalable. Posteriormente, en 2008, se empieza a pensar al Big Data como la mayor innovación en informática en la última década, ya que ha transformado la forma en que los

[17] Michael Lesk publica *"How much information is there in the world?"* (1997): `http://www.lesk.com/mlesk/ksg97/ksg.html`. Último acceso: Febrero 2021.

[18] Oracle: `https://www.oracle.com`. Último acceso: Febrero 2021.

[19] SAP: `https://www.sap.com`. Último acceso: Febrero 2021.

[20] Peoplesoft: adquirida por Oracle en Enero de 2005.

[21] Apache Hadoop: http://hadoop.apache.org/. Último acceso: Febrero 2021.

motores de búsqueda acceden a la información, las actividades de las compañías, las investigaciones científicas, la medicina, y las operaciones de defensa e inteligencia de los países, entre otras tantas actividades. Más aún, se ha comenzado a ver su potencial para recopilar y organizar datos en todos los ámbitos de la vida cotidiana (Bryant et al., 2008), tales como redes sociales, estadísticas deportivas, o avances médicos y genéticos.

3.3.2. Map-Reduce

Map-reduce es un modelo de programación popular para el procesamiento de grandes cantidades de datos mediante computación distribuida (Condie et al., 2010). En pocas palabras, se especifica una función *map* que procesa pares clave-valor para generar un conjunto de pares clave-valor intermedios, y una función *reduce* que combina todos los valores intermedios asociados con la misma clave (Dean y Ghemawat, 2008). Con este concepto, en lenguajes de programación funcionales así como también lenguajes de alto nivel modernos, es posible escribir expresiones de estilo *lambda*[22] , en las cuales es posible paralelizar y ejecutar programas en clusters de computadoras[23] distribuidos sin la necesidad de tener en cuenta los detalles de partición de datos y subprocesamiento.

3.3.2.1. Arquitectura Hadoop

La librería de software Apache Hadoop es un framework que posibilita el procesamiento de grandes conjuntos de datos entre clusters de computadoras usando modelos de programación simples. Para posibilitar esto, Hadoop se basa en una arquitectura de archivos propia, llamada HDFS[24] (Sistema de Archivos Distribuido Hadoop, por sus siglas en inglés). En la mayoría de los procesos basados en Hadoop, HDFS es utilizado para almacenar la entrada del paso "map" y la salida del paso "reduce", pero no los resultados intermedios, ya que ellos se almacenan en el sistema de archivo de cada uno de los nodos (Condie et al., 2010). Según el sitio oficial[25] , HDFS es altamente tolerante a fallos y está diseñado para ser ejecutado en computadoras de bajo costo. Además, HDFS provee gran

[22] Las funciones Lambda no tienen definido un identificador. Son también conocidas como funciones anónimas.

[23] Red de computadoras de alta velocidad que se comportan como si fuesen un único servidor. No confundir con la definición de "cluster" en algoritmos de clustering.

[24] Siglas en inglés para Hadoop Distributed File System.

[25] Arquitectura HDFS: https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html. Último acceso: Febrero 2021.

productividad accediendo a los datos de una aplicación y es posible usarlo en grandes conjuntos de datos.

HDFS tiene una arquitectura maestro-nodo[26] . Un cluster HDFS (conjunto de nodos maestro-nodos) consiste en un *NameNode*, que es un servidor maestro que maneja el espacio de nombres del sistema de archivos y regula el acceso a archivos. Además, hay un número de *DataNodes*, usualmente uno por cada nodo en el cluster, que maneja el almacenamiento de datos en archivos. Internamente, un archivo es dividido en uno o más bloques, y esos bloques son almacenados en DataNodes. Por otro lado, el NameNode ejecuta operaciones tales como abrir, cerrar, y renombrar archivos y directorios.

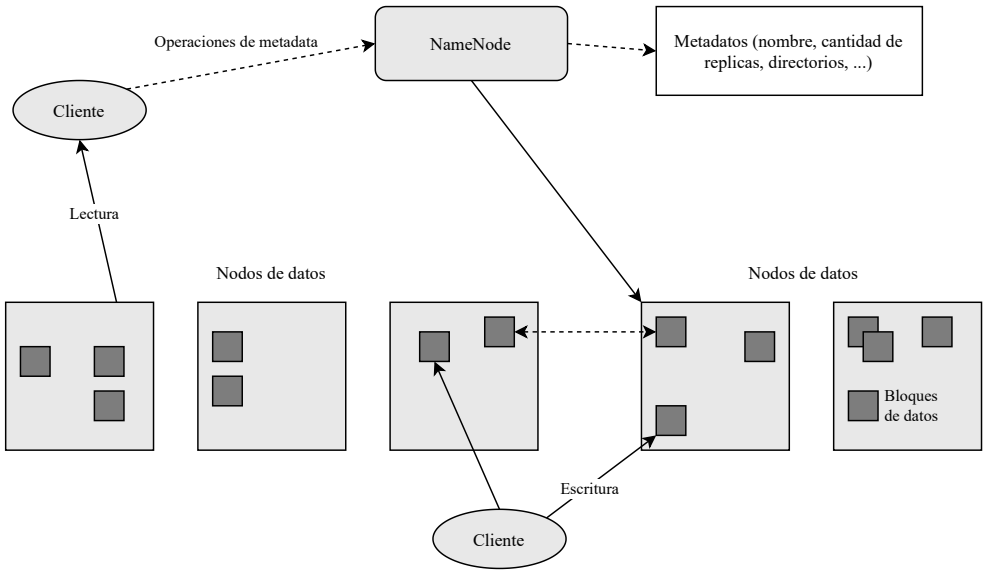

Figura 2: Arquitectura HDFS.

Tanto el NameNode como el DataNode son piezas de software diseñadas para correr, típicamente, en sistemas operativos GNU/Linux. Además, como HDFS está construido utilizando el lenguaje de programación Java, puede ser desplegado en un rango amplio de máquinas. La Figura 2 muestra un esquema acerca de como un programa cliente interactúa con los nodos de datos para escritura y lectura, mientras que también interactúa con el NameNode para realizar operaciones de metadata.

[26] Del inglés, master-node architecture.

3.3.2.2. Apache Spark

Apache Spark es una plataforma de código abierto escrita en Scala[27] , para procesamiento de datos a gran escala y preparado para tareas de Machine Learning iterativas (Meng et al., 2016). En alto nivel, una aplicación Spark consiste en un programa "driver" que ejecuta la función "main" y varias operaciones en paralelo en un cluster de computadoras.

El paralelismo en un cluster de computadoras realizado por Spark se basa en particiones de datos. Por ejemplo, si se desea procesar un conjunto de datos de un millón de registros y se cuenta con un cluster de 4 ejecutores, cada uno de ellos procesará 250.000 registros. Para conseguir esto de una forma que pueda ser entendible para un desarrollador, la plataforma cuenta con una abstracción de datos llamada RDD o *Resilient Distributed Dataset* (o conjunto de datos distribuido resiliente), la cual es una colección de datos particionados a través de los nodos de cluster que pueden operar en paralelo. Los RDDs son creados desde un archivo HDFS o una colección Scala existente en el programa driver. También es posible almacenarlos en memoria, permitiendo que estas colecciones sean reusadas eficientemente entre operaciones paralelas.

Los RDDs permiten dos tipos de operaciones: *transformaciones*, las cuales crean un conjunto de datos desde uno existente, y *acciones* u *operaciones terminales*, que devuelven un valor al programa driver luego de ejecutar una cierta cantidad de cómputos en el conjunto de datos. Estas operaciones se condicen con el modelo de programación map-reduce en el cual se basa Hadoop. Por ejemplo, en este contexto, *map* es una transformación que opera en cada uno de los elementos del conjunto de datos y devuelve un nuevo RDD representando los resultados. Por otro lado, *reduce* es una operación que agrega todos los elementos de un conjunto de datos usando alguna función y devolviendo un resultado final.

Todas las transformaciones son *lazy*, esto es, secuencias de acciones imperativas que son retrasadas hasta que el resultado es requerido (Launchbury, 1993). Es decir, las transformaciones aplicadas a un conjunto de datos son recordadas hasta que es necesario devolver un resultado final al programa driver. Además, es posible almacenar un RDD en memoria (o disco rígido) logrando así mantener

[27] Sitio oficial del lenguaje de programación Scala: https://www.scala-lang.org/. Último acceso: Febrero 2021.

elementos disponibles en un cluster para un acceso rápido en caso de que una transformación sea requerida más de una vez.

3.4. Medidas de distancia de texto

3.4.1. Conceptos básicos

3.4.1.1. Information retrieval

Information retrieval (IR), traducido a menudo como "recuperación de información", se define la acción de como encontrar material (generalmente documentos) de una naturaleza desestructurada (generalmente texto) que satisfaga una necesidad de información de grandes colecciones (generalmente almacenadas en computadoras) (Schütze et al., 2008).

El IR utiliza técnicas probabilísticas, pero también, en los últimos años, varios investigadores se han centrado en técnicas basadas en conocimiento. Estas últimas, han hecho una significante contribución al IR "inteligente". Más recientemente, la investigación se ha volcado a nuevas técnicas de aprendizaje inductivo basadas en *inteligencia artificial* (IA), las cuales incluyen redes neuronales, aprendizaje simbólico y algoritmos genéticos (Chen, 1995). Cuando hablamos de aprendizaje, nos referimos a un fenómeno multifacético. Los procesos de aprendizaje incluyen la adquisición de un nuevo conocimiento declarativo, la organización del nuevo conocimiento, representaciones efectivas del mismo, y finalmente un descubrimiento de nuevos hechos y teorías a través de la observación y la experimentación. Desde el nacimiento de la era de las computadoras, estas capacidades han querido ser implantadas en las mismas. En este sentido, el estudio y el modelado en computadoras del proceso de aprendizaje en múltiples manifestaciones constituyen el propósito principal del Machine Learning (ML) (Mitchell et al., 2013).

Los Sistemas de Recomendación a los cuales se hace foco en este trabajo, aplican técnicas que no son más que conceptos derivados de IR y algunos de ML. Técnicas probabilísticas y determinísticas son utilizadas para extraer información relevante desde diferentes orígenes de datos, así como también para generar información valiosa a partir de ellos. Para este último propósito, se han desarrollado métodos basados en el aprendizaje inductivo que demostraron ser muy

efectivos y, en algunos casos, fueron útiles para modelos simples y aplicables que permiten desarrollar RS eficaces y escalables.

3.4.1.2. Unidad de documento

Una *unidad de documento*, es una secuencia de caracteres de longitud fija con las cuales se va a trabajar. Estas secuencias de caracteres pueden estar codificadas por uno o varios bytes o esquemas de codificación multibyte, como UTF-8 o varios estándares específicos de algún país o compañía. Una vez que la codificación esté determinada, se debe decodificar la secuencia de bytes a una secuencia de caracteres.

3.4.1.3. Stopwords

En informática, se llama stopword a una palabra que se filtra antes o después del procesamiento de datos del lenguaje natural (Leskovec et al., 2014). Generalmente, este tipo de palabras son extremadamente comunes, y podrían tener poco valor en el momento de seleccionar documentos que coincidan con las necesidades de un usuario (Schütze et al., 2008).

Entonces, es necesario seleccionar una lista de stopwords, que serán filtradas en el procesamiento de las unidades de documento. Estas listas son llamadas *stop lists*. La estrategia general para el armado de stop lists, es ordenar los términos por *collection frequency*, es decir, el número total de veces que aparece un término en la colección de documentos. Una vez hecho esto, es necesario tomar los términos más frecuentes, a veces filtrados a mano según su contenido semántico relativo al dominio de los documentos que están siendo indexados. A menudo las preposiciones son buenas palabras candidatas para ser consideradas como stopwords.

3.4.1.4. Tokenización

Dado una secuencia de caracteres y una unidad de documento definida, la *tokenización* es la tarea de dividirla en distintas piezas, llamadas *tokens* y, quizás en el mismo momento, desechar ciertos caracteres (como signos de puntuación). Un ejemplo de tokenización de una secuencia de caracteres en idioma inglés, podría ser:

```
Input: Friends, Romans, Countrymen, lend me your ears;
```

```
Output: [Friends] [Romans] [Countrymen] [lend] [me] [your] [ears]
```

Estos tokens, son frecuentemente confundidos con palabras, pero es importante hacer la distinción entre *token*, *tipo*, y *término*, para comprender la diferencia. Un token es una instancia de una secuencia de caracteres en un documento en particular que están agrupados juntos como una unidad semántica útil para procesamiento. Un tipo es la clase de todos los tokens que contienen la misma secuencia de caracteres. Un *término* es un tipo que está incluido en un diccionario de un sistema de IR. Por ejemplo, si un documento a ser indexado es "to sleep perchance to dream", existen 5 tokens, pero solo 4 tipos (ya que hay dos instancias del token "to"). Sin embargo, si "to" es desechada por ser un stopword, habrá entonces 3 términos: "sleep", "perchance" y "dream".

3.4.1.5. Similaridad

Es de interés poder cuantificar la relación entre los objetos de texto. Existen distintas maneras de medir esa relación. En general se habla de medidas de proximidad como una generalización para las medidas de similaridad y disimilaridad. En este trabajo utilizaremos medidas de similaridad comúnmente encontradas en la literatura (Gomaa y Fahmy, 2013; Harispe et al., 2015; Lin et al., 1998; Resnik, 1995). Debido al propósito de este trabajo, es de interés, en particular, la similaridad semántica en taxonomías (Resnik, 1995), que se basa en el contenido de información de las unidades documentales. Una taxonomía para un corpus de conceptos se estructura a la manera de un árbol, con un nodo raíz conectado mediante arcos a otros nodos.

Las medidas basadas en contenido de información de una unidad de documento (o concepto) en una taxonomía, utilizan el siguiente enfoque. Se define $p(t)$ como la probabilidad de un concepto t, formalizando:

$$p(t) = \frac{freq(t)}{N},$$

donde $freq(t)$ es la frecuencia que aparece un concepto, teniendo en cuenta el concepto en sí y todos sus antecesores en el árbol; y N la cantidad total de conceptos en el corpus. De esta forma, el nodo raíz tendrá $p(t) = 0$, mientras los nodos hojas, tendrán valores cercanos a 1. Se define el contenido de información $I(t) = 0$ como:

$$I(t) = -\log p(t).$$

Aplicando el logaritmo negativo a $p(t)$ se logra que los nodos hoja, siendo conceptos muy específicos, contengan mucha información, y que los nodos más genéricos que se encuentran cerca de la raíz posean un contenido de información que, por el contrario, tiende a cero.

En (Resnik, 1995) se define, dado un conjunto de conceptos C en una taxonomía *es-un* que permite herencia múltiple, que la similaridad entre dos conceptos es la medida en que ellos comparten información en común, indicado, en este tipo de taxonomías, por el nodo inmediato de más alto nivel que los subsume a ambos, el *subsumidor mínimo* (minimum subsumer o ms por sus siglas en inglés).

Los autores en (Resnik, 1995) definen, en un conjunto de términos C en una taxonomía con relaciones de tipo *es-un* que permite herencia múltiple, que la similaridad entre dos términos es la medida en que ellos comparten información en común, indicado, en este tipo de taxonomías, por el nodo inmediato de más alto nivel que los subsume a ambos, denominado el *subsumidor mínimo* (*minimum subsumer* o *ms* por sus siglas en inglés).

Considerando dos términos t_i y t_j, y a $S(t_i, t_j)$ al conjunto de ancestros comunes de t_i y t_j en la taxonomía, se define al subsumidor mínimo $ms(t_i, t_j)$, como al término de $S(t_i, t_j)$ que contiene el máximo contenido de información:

$$\max_{t \in s(t_i, t_j)} I(t) = I(ms(t_i, t_j)).$$

La medida de similaridad de Resnik (1995) S_R, es entonces, el contenido de información del subsumidor mínimo de dos términos:

$$S_R = I(ms(t_i, t_j)).$$

El enfoque anterior, posee la siguiente particularidad: no tiene en cuenta la similaridad de los nodos con respecto a su subsumidor mínimo. Es intuitivo pensar que dos términos que definen conceptos abstractos (nodos ubicados en posiciones más cercanas al subsumidor mínimo) son más parecidos entre sí que dos términos que definen conceptos específicos (más alejados del subsumidor mínimo) (Lin et al., 1998). Para solucionar esto, Lin et al. (1998) tiene en cuenta dos aspectos para calcular la similaridad entre dos términos en este tipo de taxonomías: (I) La cantidad de información; y (II) La ubicación relativa entre

los nodos hijos respecto al subsumidor mínimo. Esta medida se define como la similaridad de Lin:

$$S_L(t_i, t_j) = \frac{2S_R(t_i, t_j)}{I(t_i) + I(t_j)}.$$

El resultado de esta medida, estará normalizada en el rango $[0, 1]$ y obtiene el resultado de que nodos más generales, con menor cantidad de información, sean más similares entre sí, que dos nodos específicos (ya que la cantidad de información de los mismos aumentará el denominador de la ecuación, disminuyendo el resultado final). Se ha demostrado que esta definición de similaridad produce una correlación ligeramente mayor con el juicio o percepción sobre similaridad de conceptos dado por humanos.

3.4.1.6. Medidas de proximidad

Como mencionamos en el apartado anterior, proximidad es una generalización de similaridad y disimilaridad. La función disimilaridad, también conocida como función de distancia, en un conjunto de datos X, es definida para satisfacer las siguientes condiciones. Estas condiciones son las utilizadas en (Xu y Wunsch, 2008) y de relevancia en el presente trabajo:

1. Simetría,
$$D(x_i, x_j) = D(x_j, x_i);$$
2. Positividad,
$$D(x_i, x_j) \geq 0 \quad \forall x_i, x_j;$$

De forma análoga, la función de similaridad es definida satisfaciendo las condiciones:

1. Simetría,
$$S(x_i, x_j) = S(x_j, x_i);$$
2. Positividad,
$$0 \leq S(x_i, x_j) \leq 1, \quad \forall x_i, x_j.$$

Si bien el término matemático formal de distancia exige una serie de supuestos rigurosos (Xu y Wunsch, 2008), en este trabajo utilizaremos la noción de distancia y de disimilaridad en forma indistinta, y nos basaremos en las medidas de proximidad habitualmente utilizadas para comparación de texto. Por lo tanto, para transformar una medida de similaridad $S(x_i, x_j)$ en una de distancia

$D(xi, xj)$ que cumpla $0 \leq D(x_i, x_j) \leq 1$, haremos la normalización de la misma en el intervalo $[0, 1]$ y luego aplicaremos el cálculo $D(x_i, x_j) = 1 - S(x_i, x_j)$ y recíprocamente (Leale et al., 2013).

3.4.1.7. Modelo de espacio vectorial

En el modelo de *espacio vectorial*, un texto es representado como un vector de términos. Si las palabras son elegidas como términos, entonces cada palabra del vocabulario sería una *dimensión* independiente en el espacio vectorial (Singhal et al., 2001). Todo texto puede ser representado por un vector en este espacio dimensional. Si un término pertenece a un documento, este obtiene un valor distinto de cero en el vector, junto con la dimensión correspondiente al término. Como un documento contiene un conjunto limitado de términos (el vocabulario puede contener millones de términos), muchos de los vectores pueden ser muy dispersos. La mayoría de los sistemas basados en vectores trabajan en el cuadrante positivo, es decir, a ningún término se le asigna un valor negativo.

Como caso de aplicación, para asignar un valor numérico a un documento en una consulta (verificando si un documento en una consulta es similar a un documento existente), el modelo mide la similaridad entre el vector ingresado en ella y el vector del documento al cual se quiere consultar. La similaridad entre dos vectores no es inherente al modelo. Típicamente, el ángulo entre los dos vectores es usado como medida de divergencia entre los mismos, y el coseno del ángulo es usado como similaridad numérica, ya que el coseno tiene la propiedad de ser tener resultado 1 cuando los vectores son idénticos y 0 cuando los vectores son ortogonales (explicado en detalle más adelante). Como una alternativa, el producto escalar entre dos vectores es también usado como medida de similaridad. Si todos los vectores están forzados a tener longitud 1, es decir, vectores unitarios, entonces el coseno del ángulo entre los vectores tiene el mismo resultado que el producto escalar.

Formalizando, si $\overrightarrow{D}$ es el vector del documento y $\overrightarrow{Q}$ es el vector de la consulta, la similaridad entre $\overrightarrow{D}$ y $\overrightarrow{Q}$ es representada como:

$$S(\vec{D}, \vec{Q}) = \sum_{ti \in Q,D} W_{t_{iQ}}.W_{t_{iD}},$$

donde $W_{t_{i\vec{Q}}}$ es el valor de la componente i del vector $\overrightarrow{Q}$ y $W_{t_{i\vec{D}}}$ es el valor de la componente i del vector $\overrightarrow{D}$, también definido como el peso del término i en el

documento D. Cualquier término no presente en la consulta o en el documento tendrá valor cero en $W_{t_{i\vec{Q}}}$ o $W_{t_{i\vec{D}}}$, por lo cual es posible hacer la sumatoria solo de los términos en común entre la consulta y el documento.

3.4.1.8. Distancia del coseno

La *distancia del coseno* es posiblemente la que se aplica en mayor frecuencia en términos de similaridad en IR (Korenius et al., 2007). Al aplicar la distancia del coseno se obtiene un resultado que se encuentra en el rango $[-1, 1]$. El valor -1 significa que los vectores tienen la misma dirección, pero sentidos opuestos. El valor 1, por el contrario, significa que el ángulo comprendido entre los vectores es cero.

Particularmente en IR, es de interés el intervalo $[0, 1]$ ya que todos los componentes de un vector que representa a un documento, son no negativos. De esta interpretación, se deriva la definición de la distancia del coseno, restando la medida del coseno de su máximo valor:

$$d_c(\vec{D}_i, \vec{D}_j) = 1 - cos(\vec{D}_i, \vec{D}_j) = 1 - \frac{\vec{D}_i' \vec{D}_j}{\sqrt{\vec{D}_i' \vec{D}_i}\sqrt{\vec{D}_j' \vec{D}_j}},$$

donde $1 \leq i, j \leq n$ siendo n la cantidad de vectores en el corpus. Los símbolos $\overrightarrow{D_i}$ y $\overrightarrow{D_j}$ son documentos en forma de vectores y d_c es la distancia entre ellos.

Para simplificar, y teniendo en cuenta que estamos hablando de documentos en forma de vectores, la distancia del coseno se puede derivar de la fórmula del *producto escalar* (producto punto). Siendo $\overrightarrow{D_i}$ y $\overrightarrow{D_j}$ dos documentos en forma de vectores, se define el producto escalar entre ellos, como:

$$\vec{D}_i.\vec{D}_j = \left\|\vec{D}_i\right\| . \left\|\vec{D}_j\right\| .cos(\theta),$$

siendo $\left\|\overrightarrow{D_i}\right\|$ y $\left\|\overrightarrow{D_j}\right\|$ los módulos de los vectores $\overrightarrow{D_i}$ y $\overrightarrow{D_j}$ respectivamente, y θ el ángulo formado entre ellos. Entonces:

$$cos(\theta) = \frac{\vec{D}_i.\vec{D}_j}{\left\|\vec{D}_i\right\| . \left\|\vec{D}_j\right\|},$$

donde d_i y d_j son los componentes de los vectores $\overrightarrow{D_i}$ y $\overrightarrow{D_j}$ respectivamente.

3.4.2. Term Frequency

También conocido en la literatura como *Bag of words* (bolsa de palabras) o simplemente, de sus iniciales, bow. *Term Frequency*, o TF, es una técnica donde el orden exacto de los términos es ignorado, pero se basa en el número de ocurrencias de cada uno de ellos en un documento (Christopher et al., 2008). Los términos anteriormente mencionados como referencia a esta técnica se van a usar indistintamente en este documento.

Si comparamos las frases en inglés *"Mary is quicker than John"* y *"John is quicker than Mary"*, desde esta perspectiva (ignorando el orden), son idénticas. Esto lleva a ciertos problemas desde el punto de vista semántico, ya que, si bien el número de ocurrencia de términos es el mismo, el significado de las frases es opuesto. Las palabras se cuentan en una *bolsa*, lo cual difiere de la definición matemática de *conjunto*, que no considera los elementos repetidos. Cada término corresponde a una dimensión en el espacio de datos resultante y cada documento corresponde a un vector compuesto por valores no negativos en cada dimensión (Huang, 2008). Con los documentos representados como vectores, se mide el grado de similaridad de dos documentos como la correlación entre los vectores correspondientes, que puede ser cuantificada como el coseno del ángulo entre ellos. Entonces, es posible obtener la distancia entre dos documentos en forma de vectores $\overrightarrow{D_1}$ y $\overrightarrow{D_2}$ aplicando:

$$d(\vec{D}_1, \vec{D}_2) = 1 - \frac{\vec{D}_1 \vec{D}_2}{\left\|\vec{D}_1\right\| \left\|\vec{D}_2\right\|}.$$

3.4.3. Term Frequency/Inverse Document Frequency

Hasta ahora, no se ha mencionado qué implicancia tiene la frecuencia de cada término en uno o varios documentos de una colección. Se define *document frequency* df_t como el número de documentos en una colección que contienen el término t, en contraste con TF que indica que tan frecuentemente aparece un término en un documento dado. El propósito de este indicador es discriminar a nivel de documento para asignar un peso a cada uno de los términos existentes en la colección de documentos (Christopher et al., 2008), para lo cual se define *inverse document frequency*, o IDF, un indicador basado en la cantidad de documentos que contienen (o son indexados por) un término en cuestión (Robertson, 2004). La intuición se basa en que si un término de búsqueda se encuentra en

muchos documentos, no es un buen discriminador, y se le debe asignar menor peso que a un término que se encuentra en pocos documentos.

Este método, conocido como *Term Frequency/Inverse Document Frequency* (TF-IDF), está basado en multiplicar la medida IDF con la medida TF. Asumiendo que existen N documentos en una colección, y un término t_i aparece en n_i documentos (df_t), entonces la medida IDF propuesta es (Sparck Jones, 1972):

$$idf(t_i) = \log \frac{N}{n_i}.$$

Palabras con alto valor TF-IDF implican una fuerte relación con el documento en el cual aparecen, es decir, que TF-IDF mide la relevancia de un término en un documento dado. Estas palabras, que son comunes sólo en un documento o en un pequeño grupo de ellos, tienden a tener un valor TF-IDF más grande que palabras comunes como artículos y preposiciones (Ramos et al., 2003).

El procedimiento formal para implementar TF-IDF tiene algunas diferencias menores entre diferentes aplicaciones, pero básicamente consiste como se indica a continuación. Siendo t_i un término que se encuentra en un documento d_j, se calcula:

$$tfidf(t_i, d_j) = tf(t_i, d_j) \cdot idf(t_j),$$
$$tfidf(t_i, d_j) = tf(t_i, d_j) \cdot \log \frac{N}{n_j}.$$

Esta fórmula deriva en distintos valores que puede tomar el resultado final.

Teniendo en cuenta cada uno de los parámetros involucrados, se van a analizar algunas variaciones. Consideremos en principio que $N \cong n_j$, es decir que el término t_i aparece en casi todos los documentos. Esto implica que $1 < \log(\frac{N}{n_j}) < c$ para un c con un valor pequeño y el valor de TF-IDF será menor que el valor de IDF, pero aún positivo (ya que $0 \leq tf(t_i, d_j) \leq 1$). Esto implica que el término t_i es relativamente común entre todos los documentos pero aún posee cierta importancia sobre N. Este puede ser el caso de palabras muy comunes en documentos, como pronombres o preposiciones, las cuales no tienen ningún significado relevante por sí mismas, y recibirán un valor TF-IDF muy bajo. Finalmente, supongamos que el valor TF es alto, y n_j es un número pequeño. Entonces $\log(\frac{N}{n_j})$ será bastante grande y así también lo será el valor total TF-IDF. Este último caso es en el cual estamos interesados, ya que el caso en el que las palabras poseen alto TF-IDF implica que son importantes en n_j pero no son comunes en la

totalidad de los documentos. Se dice entonces que este término t_i tiene un gran *poder discriminatorio*.

TF-IDF es un algoritmo eficiente y simple para medir el peso que tiene una palabra en una consulta para un conjunto de documentos que son relevantes para esa consulta. Además, TF-IDF es simple de codificar, sentando las bases para algoritmos más complicados y sistemas de IR. Más allá de esas fortalezas, TF-IDF tiene sus limitaciones. En cuanto a sinónimos, no hace ninguna relación entre palabras. Es decir, que no considerará documentos en los cuales la palabra no se encuentra pero si se encuentra un sinónimo de la misma, ya que las palabras son tomadas como conceptos totalmente diferentes. El mismo problema ocurre cuando se trata de plurales o una variación del tiempo verbal de una palabra: por ejemplo, "país" y "países" serán categorizadas separadas, en lugar de categorizarlas como una sola palabra y calcular un valor TF-IDF más bajo.

3.4.4. Word2Vec

Word2Vec (W2V) presenta dos modelos basados en redes neuronales para computar representaciones de palabras como vectores continuos en grandes conjuntos de datos. La calidad de la representación es medida como similaridad (Mikolov et al., 2013). La forma de representar los vectores está basada en su contexto, el cual es usado para optimizar la representación del vector, es decir, palabras con significado similar son representadas con vectores que están más cerca unos de otros. Esto indica que, por primera vez en las medidas analizadas en el estado del arte de este trabajo, W2V toma un enfoque semántico.

3.4.4.1. Motivación de Word2Vec

Muchos sistemas de *procesamiento de lenguaje natural* (NLP por sus siglas en inglés correspondientes a Natural Language Processing), tratan a las palabras como unidades atómicas y no tienen noción de contexto entre las palabras. Este tipo de sistemas son limitados para varias tareas, ya que no proporcionan información sobre las relaciones que pueden existir entre las distintas entidades. Con el progreso del Machine Learning en los últimos años, ha sido posible entrenar modelos más complejos con grandes conjuntos de datos, y en ellos se observa un rendimiento mejor que en los modelos más simples. Probablemente, el concepto más exitoso es la *representación distribuida de palabras*. En estos modelos, fue

descubierto que la similaridad entre palabras va más allá que simples regularidades sintácticas, por el contrario es posible realizar operaciones algebraicas entre representaciones vectoriales de palabras, tal como:

```
vector("Rey") - vector("Hombre") + vector("mujer") = vector("Reina").
```

Los vectores adhieren muy bien a nuestra intuición, por ejemplo, palabras que son sinónimos tienden a tener vectores similares en términos de la distancia del coseno, mientras que antónimos, tienden a tener vectores disímiles. Por lo anterior, el objetivo es maximizar la precisión de estas operaciones entre vectores mediante arquitecturas de modelos que logren preservar regularidades lineales entre palabras así como también minimizar la complejidad computacional. Estos vectores son útiles por dos razones principales (McCormick, 2016):

1. Se puede medir la similaridad semántica entre dos palabras calculando la distancia del coseno entre los vectores correspondientes.

2. Es posible usar los vectores en distintas tareas de NLP, tales como clasificación de documentos o análisis de sentimientos.

Las arquitecturas propuestas por W2V se basan en dos modelos, *Skip-Gram* y *CBOW* (Continuous Bag of Words). Ambos modelos son simples redes neuronales con una capa oculta, y los resultados son vectores construidos con los pesos de la misma, luego de la aplicación de propagación hacia atrás y *descenso de gradiente estocástico* (SDG por sus siglas en inglés correspondientes a Stochastic Gradient Descent). La Figura 3 muestra la diferencia entre ambos modelos, los dos utilizando una capa oculta: a la izquierda, CBOW utiliza un conjunto de palabras y predice la palabra actual $w(t)$; a la derecha, Skip-gram predice palabras vecinas a partir de una palabra $w(t)$. El detalle de cada uno de los métodos es descrito a continuación.

3.4.4.2. Modelo Skip-Gram

El modelo Skip-gram se basa en lo siguiente: dado un conjunto de frases, el modelo analiza las palabras de cada sentencia y utiliza cada palabra para predecir las palabras que serán vecinas. Esta técnica se basa en la *propagación hacia atrás*. La entrada para este modelo es una sola palabra w_I y la salida, su contexto $\{w_{0,1}, ..., w_{0,C}\}$ definido por una ventana de tamaño C. Una potencial instancia de entrenamiento podría ser la palabra "auto" como entrada y las palabras {"Manejé","mi","hacia","la","tienda"} las salidas. El objetivo es, mediante

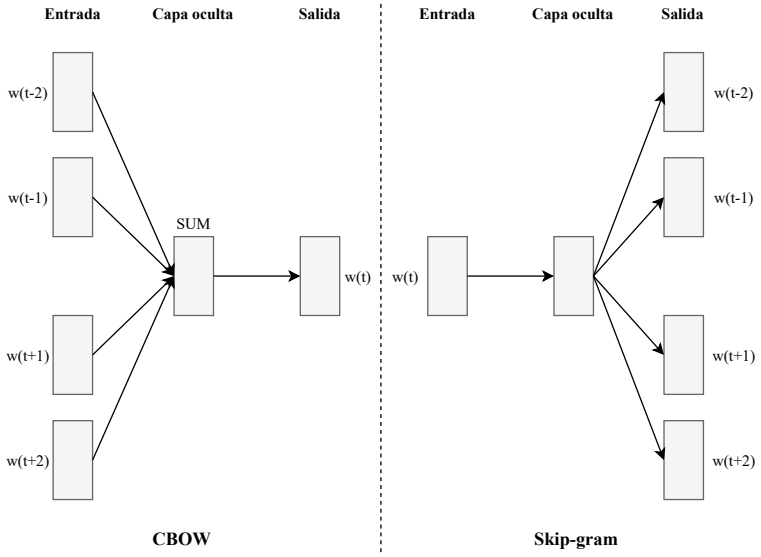

Figura 3: La arquitectura CBOW predice la palabra actual basándose en el contexto, mientras que la arquitectura Skip-gram predice palabras vecinas, dada una palabra como entrada (Mikolov et al., 2013).

el entrenamiento, generar los pesos de la capa oculta de la red neuronal y usarlos como los vectores que representarán las palabras. Mediante la palabra que es usada como entrada, el modelo busca palabras "vecinas" pertenecientes a su contexto y elige una de forma aleatoria. La red neuronal dará como resultado la probabilidad de cada palabra en nuestro vocabulario de ser *palabra vecina*. El concepto de palabra vecina, viene dado por el tamaño de ventana que hayamos elegido, si el tamaño de ventana es 5, significa que son tomadas en cuenta 5 palabras adelante y 5 palabras atrás (10 en total) (Minnaar, 2015a).

La entrada de la red neuronal será una palabra representada como *one-hot vector*, esto es, un vector con tantas posiciones como tamaño tenga el vocabulario; esto es, si tenemos un vocabulario de 10000 palabras, el vector tendrá el valor 1 en la posición donde se encuentre la palabra representada, y 0 en las 9999 restantes. La salida de la red neuronal es otro vector con las mismas 10000 posiciones, que en cada posición contiene las probabilidades de que cada una de las palabras sean vecinas de la palabra representada en la entrada. Las probabilidades son calculadas de la siguiente forma. Dado un one-hot vector x de tamaño V (cantidad de palabras del vocabulario) correspondiente a la palabra de entrada, y un conjunto de redes neuronales de la capa oculta de tamaño N, para una ventana de tamaño C, tendrémos $\{y_1, ..., y_C\}$ vectores one-hot correspondientes a las palabras de salida en la instancia de entrenamiento. La matriz $W = V \times N$ es la matriz de pesos entre la capa de entrada y la capa oculta, cuya fila i-ésima

representa los pesos correspondientes a la palabra i-ésima del vocabulario. Esta matriz W (inicializada aleatoriamente) contiene entonces la codificación como vector de cada una de las palabras del vocabulario (como filas), las cuales serán entrenadas mediante la red neuronal y, luego de varias iteraciones, se convertirán en los vectores que representen cada una de las palabras del vocabulario. El siguiente ejemplo ilustra, cómo la matriz W multiplicada por el vector de entrada, resulta en el vector que la representa.

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} 12 & 14 & 2 \\ 6 & 19 & 74 \\ 22 & 33 & 96 \\ 36 & 85 & 2 \\ 95 & 1 & 56 \end{bmatrix} = \begin{bmatrix} 36 & 85 & 2 \end{bmatrix}.$$

Se puede ver en este ejemplo cómo el vector resultado de tamaño $1 \times N$ es la 4 fila de la matriz W, ya que la palabra de entrada es un vector one-hot con el valor 1 en el cuarto elemento.

Una vez obtenido el vector de salida, éste es usado como entrada para la capa de salida de la red neuronal, usando un clasificador de regresión Softmax (Morin y Bengio, 2005), el cual es una generalización de la regresión logística para el caso en que se quieren manejar múltiples clases[28] . Cada una de las V neuronas de la capa de salida, producirá un valor entre 0 y 1. Además, la suma de todos estos valores, será 1, es decir, esta será una distribución de probabilidades. Específicamente, cada neurona de salida tiene un vector de pesos correspondiente a cada una de las palabras del vocabulario, que se multiplica con el vector que proviene de la capa oculta y se aplica una función Softmax. El resultado es la probabilidad de que aleatoriamente se seleccione la palabra de entrada y sea vecina a cada una de las palabras representadas por los vectores en las neuronas de salida. El objetivo del entrenamiento es minimizar la sumatoria de los errores a los largo de todas las palabras del contexto en la capa de salida, para luego, como objetivo final, ajustar los pesos de la capa oculta que se convertirán en la codificación vectorial de las palabras del vocabulario. La interpretación de lo anterior se resume a que si dos palabras diferentes tienen un contexto similar, el modelo tendrá una salida similar para las mismas. Una forma de que la red neuronal obtenga dos salidas similares es que sus vectores de palabras (definidos en la capa oculta)

[28] Tutorial en profundidad de regresión Softmax `http://ufldl.stanford.edu/tutorial/supervised/SoftmaxRegression`, Último acceso: Febrero 2021.

también sean similares. Por lo cual, si dos palabras tienen contextos similares, la red neuronal va a tender a tener representaciones vectoriales similares para dichas palabras.

3.4.4.3. Modelo Continuous Bag of Words (CBOW)

Es similar al modelo Skip-gram pero con las salidas y las entradas invertidas: dado un conjunto de palabras que definen un contexto, CBOW predice la palabra actual. La capa de entrada consiste en C palabras del contexto, codificadas como vectores one-hot, es decir un conjunto $\{x_1, ..., x_C\}$ cada uno de dimensión V. La capa oculta es un vector h de tamaño N. Finalmente, la salida es una palabra y también codificada como un vector one-hot. Los vectores de entrada son conectados con la capa oculta por una matriz $W = V \times N$, y esta capa oculta a su vez es conectada con la capa de salida vía otra matriz $W' = V \times N$ (Minnaar, 2015b).

La salida de la capa oculta h de la red neuronal es el promedio de los vectores de entrada, ponderados por la matriz W.

$$h = \frac{1}{C} W \cdot (\sum_{i=1}^{C} x_i).$$

Este cómputo es una de las diferencias principales con el modelo Skip-gram. Luego, se calculan las entradas de cada nodo de la capa de salida.

$$u_j = {v'_{w_j}}^T \cdot h,$$

donde v'_{w_j} es la j-ésima columna de la matriz W'. Finalmente, se calcula el resultado de la capa de salida. Dicha salida y_j es obtenida pasando la entrada u_j por la función Softmax.

$$y_j = p(w_{y_j}|w_1, ..., w_C) = \frac{exp(u_j))}{\sum_V^{j=1} exp(u'_j)}.$$

Se puede observar cómo funciona el clasificador Softmax (como también ocurre en el modelo Skip-gram). Específicamente, cada neurona de salida tiene un peso que se multiplica contra un vector de palabra de la capa oculta, y luego se aplica la función $exp(x)$ al resultado. Finalmente, para lograr que la sumatoria de las salidas sea 1, se divide este resultado por la sumatoria de los resultados de todos los nodos de salida. Entonces, la capa de salida es una palabra y_j codificada como un vector one-hot de dimensión V.

En resumen, Word2Vec utiliza dos enfoques de redes neuronales para la generación de la representación vectorial de las palabras de un vocabulario V: utilizando contexto para predecir una palabra objetivo (CBOW), o utilizando una palabra para predecir un contexto objetivo (Skip-gram). El último es especialmente utilizado porque produce resultados más precisos en grandes conjuntos de datos, y es el cual se utiliza en el presente trabajo. En ambos enfoques, una función de descenso de gradiente estocástico es utilizada para minimizar el error de las probabilidades en cuanto a las relaciones entre las palabras del contexto, y ajustar así los pesos de la capa oculta iterativamente, que darán como resultado a la representación vectorial de palabras utilizadas para el cálculo de similaridad en este trabajo.

3.4.5. FastText

Fastext[29] es una librería open-source que permite generar representaciones de texto y clasificadores de texto. Está basada en una técnica representación continua de palabras, que tiene en cuenta la morfología[30] de las mismas (Bojanowski et al., 2017). El modelo de FastText está basado en el modelo Skip-gram, donde cada palabra es representada como una bolsa de caracteres o $n-gramas$. Y a cada uno de los n-gramas, se le asignará un vector, que luego, sumados, representarán cada palabra. Dos características muy importantes son, (I) el método es muy rápido, permitiendo entrenar grandes conjuntos de datos; y (II) permite calcular la representación de palabras que no aparecen en el conjunto de entrenamiento.

Repasando el modelo Skip-gram, introducido en (Mikolov et al., 2013), dado un vocabulario V, donde una palabra es identificada por su índice $w \in \{1, ..., V\}$, el objetivo es aprender la representación vectorial de cada palabra w. Dado un documento grande representado como una secuencia de palabras $w_1, ..., w_t$, el objetivo del modelo Skip-gram es que las representaciones de las palabras sean entrenadas para *predecir bien* otras palabras que aparecen en el contexto, es decir, minimizar la siguiente probabilidad logarítmica:

$$\sum_{t=1}^{T} \sum_{c \in C_t} \log p(w_c | w_t),$$

[29] Sitio de FastText `https://fasttext.cc`. Último acceso: Febrero 2021.

[30] Parte de la lingüística que estudia las reglas que rigen la flexión, la composición y la derivación de las palabras. Por ejemplo, las distintas conjugaciones de los verbos en español.

donde el contexto C_t es el conjunto de índices de palabras que están alrededor de w_t.

3.4.5.1. Modelo sub-palabra

Usando una representación en forma de vector para cada palabra, se ignora la estructura interna de las mismas. El modelo sub-palabra, propone una función de scoring s, con el objetivo de tener en cuenta esta información. Cada palabra w es representada como una bolsa de n-gramas. Se agregan los símbolos "<" y ">" para delimitar el principio y el final de cada palabra. También, se incluye la palabra en sí misma como otro n-grama. Por ejemplo, la palabra "where" y con un $n = 3$ (longitud de la sub-palabras, en este caso serían tri-gramas), obtenemos:

`'<wh', 'whe', 'her', 'ere', 're>' y '<where>'.`

Cabe aclarar que los símbolos "<" y ">" también son considerados como caracteres a la hora de construir los n-gramas. Por ejemplo, "$< wh$" es un trigrama con un delimitador de inicio de palabra.

Dado un diccionario de n-gramas de tamaño G y una palabra w, el conjunto de n-gramas que aparecen en w es $G_w \in \{1, ..., G\}$. Se asocia un vector z_g a cada n-grama g. Se representa una palabra como la suma de todas las representaciones vectoriales de sus n-gramas. Entonces, la función scoring, es:

$$s(w, c) = \sum_{g \in G_w} z_g^T v_c.$$

Este modelo permite compartir las representaciones entre las palabras y, también, aprender una representación confiable de palabras extrañas o poco frecuentes.

FastText se puede ver como una extensión del modelo Word2Vec, pero que trata cada palabra como una composición de n-gramas. Mientras Word2Vec entrena vectores utilizando palabras completas (y el contexto de las mismas), FastText utiliza sus n-gramas (y también la palabra completa) para realizarlo. Este enfoque, agrega costo computacional en el momento del entrenamiento, pero la ventaja es que los vectores resultantes tendrán información sub-palabra, lo cual han demostrado lograr mayor precisión en diferentes medidas de rendimiento.

FastText se puede ver como una extensión del modelo Word2Vec, pero que trata cada palabra como una composición de n-gramas. Mientras Word2Vec entrena vectores utilizando palabras completas (y el contexto de las mismas), FastText utiliza sus n-gramas (y también la palabra completa) para realizarlo. Este enfoque, agrega costo computacional en el momento del entrenamiento, pero la ventaja es que los vectores resultantes tendrán información sub-palabra, lo cual han demostrado lograr mayor precisión en diferentes medidas de rendimiento.

3.4.6. Semantic Distance

La *distancia semántica* usada en este trabajo está basada en redes semánticas y estadísticas de corpus (Li et al., 2006). El método en cuestión está basado en el cálculo de similaridad entre textos de distancia corta, que tiene en cuenta la información semántica y la información del orden de las palabras implicadas en las frases involucradas. La información semántica de las frases es obtenida desde una base de datos léxica y *estadísticas de corpus*[31] . El uso de una base de datos léxica posibilita modelar el sentido común humano y las estadísticas de corpus permite adaptar el método a diferentes dominios.

Uno de los puntos dominantes por el cual este método es particularmente efectivo para calcular similaridad entre frases cortas se basa en que tradicionalmente los métodos basados en palabras comunes funcionan bien sólo para textos largos, para los cuales existe una alta posibilidad de que una palabra se encuentre en los documentos en comparación. En cambio, para textos cortos, la coocurrencia de palabras entre ellos es muy rara o nula. Esto se debe a la flexibilidad de los lenguajes de expresar mismos significados usando diferentes frases en términos de estructura y palabras.

3.4.6.1. El método

El procedimiento de cálculo de similaridad entre dos frases, como se muestra en la Figura 4, utiliza un conjunto de palabras en común usando todas las palabras distintas en el par de frases (en lugar de usar un conjunto fijo como vocabulario). De cada una de las frases, se deriva un vector semántico sin procesar, utilizando la base de datos léxica. Además, un vector de orden de palabras es generado a partir de cada frase utilizando la base de datos. Como cada palabra en

[31] El estudio de los aspectos estadísticos de los textos se ha centrado tradicionalmente en el análisis de la frecuencia de los elementos y fenómenos que se encuentran en ellos, especialmente en lo referido al componente léxico.

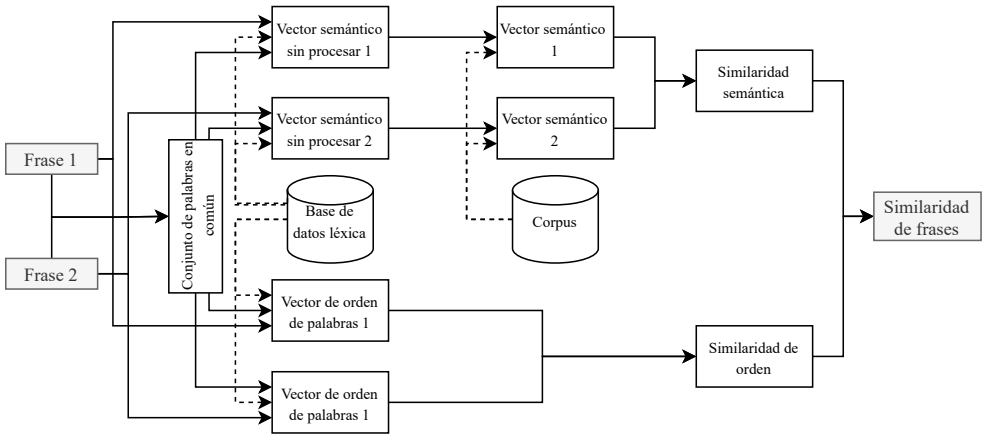

Figura 4: Diagrama de cálculo de similaridad semántica.

una frase, contribuye de forma diferente al significado de la misma, la importancia de cada una de ellas es ponderada utilizando información de contenido derivado del corpus. Combinando la información de corpus con los vectores semánticos sin procesar, se obtienen nuevos vectores semánticos que servirán para calcular la *similaridad semántica*. Una *similaridad de orden* es calculada usando los dos vectores de orden previamente generados. Finalmente, la similaridad entre frases es calculada combinando la similaridad semántica y la similaridad de orden.

3.4.6.2. Similaridad semántica entre palabras

Las bases de conocimiento tienden a representar el sentido común humano con una estructura jerárquica para un dominio específico o para un lenguaje en general. Existen varios proyectos lingüísticos disponibles en la web, tales como WordNet (Miller, 1995), Spatial Date Transfer Standard de la USGS[32] , y Gene Ontology[33] . Esta estructura jerárquica se puede modelar como una taxonomía de conceptos.

Dadas dos palabras w_1 y w_2, es necesario encontrar la similaridad $S(w_1, w_2)$ entre ellas. Esto es posible analizando la base de datos léxica (en este caso WordNet si se cuenta con palabras en el idioma inglés) de la siguiente forma: Las palabras están organizadas como conjuntos de sinónimos (llamados *synsets*), con relaciones semánticas a otros synsets. Un método directo para calcular la similaridad es encontrar el camino más corto que conecta dos palabras dentro de la jerarquía. Por ejemplo, si tomamos la estructura detallada en la Figura 5, la cual muestra un esquema de base de datos semántica en forma jerárquica que

[32] United States Geological Survey. https://www.usgs.gov/. Último acceso: Febrero 2021.
[33] Gene Ontology. http://geneontology.org/. Último acceso: Febrero 2021.

tiene como raíz al synset "entidad", el camino más corto para conectar "niño" y "niña" sería "niño-persona masculina-humano-persona femenina-niña", y tiene una longitud de 4. El synset "persona, humano", en este caso, es llamado *subsumidor mínimo de palabras*, tal como se menciona en la sección 3.4.1.5, ya que sirve de anclaje entre los dos conceptos para los cuales se quiere calcular la distancia semántica. También, se puede ver que la distancia entre "niño" y "profesor" es 6, mientras la distancia entre "niño" y "animal" es 4, lo cual es contra intuitivo e indica que el proceso es mejorable agregando más información de estas estructuras jerárquicas. Es evidente que los synsets pertenecientes a capas altas de la estructura tienen conceptos semánticos más generales y menos similaridad entre ellos, y en capas inferiores, sucede lo contrario, conceptos más particulares y similares. Lo anterior sugiere tomar en cuenta la profundidad en jerarquía.

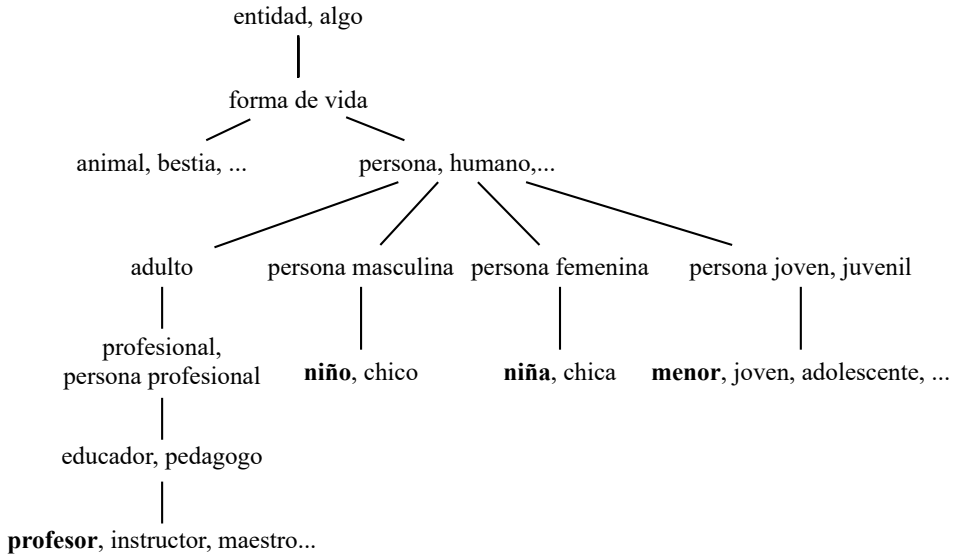

Figura 5: Ejemplo de base de datos semántica de forma jerárquica.

Para resumir, la similaridad entre palabras es determinada, no solo por el camino más corto entre ellas, sino también por la profundidad jerárquica, es decir:

$$S(w_1, w_2) = f(l, h),$$

donde l es el camino más corto entre w_1 y w_2, y h es la profundidad del subsumer de las mismas en la jerarquía de la red semántica.

3.4.6.3. Similaridad semántica entre frases

Contrariamente a métodos clásicos que usan vocabularios con miles de palabras precompiladas, este método semántico usa únicamente vectores semánticos

formados por las frases en comparación. Un conjunto de palabras comunes T es formado por la unión de las dos frases en comparación. Luego, un vector semántico léxico es derivado de este conjunto. Cada entrada del vector corresponde a una palabra del conjunto de palabras comunes, lo cual indica que la dimensión del vector es igual al número de palabras del conjunto. El valor de una entrada del vector semántico es calculado de la siguiente forma:

- **Caso 1**. Si w_i aparece en la frase, s_i es 1.
- **Caso 2**. Si w_i no está contenida en T_1, se calcula una similaridad semántica entre w_1 y cada palabra en T_1 utilizando el método de la sección anterior.

Luego, es necesario ponderar cada una de las palabras basadas en su contenido de información (Ribadas et al., 2005). Cada celda es ponderada por la información de contenido $I(w_i)$ y $I(\widetilde{w}_i)$. Finalmente, el valor de la entrada del vector semántico es:

$$s_i = \check{s} \cdot I(w_i) \cdot I(\widetilde{w}_i),$$

donde w_i es una palabra en el conjunto de palabras comunes, y $I(\widetilde{w}_i)$ es su palabra asociada en la frase. Entonces, la similaridad semántica entre dos frases es definida como el coeficiente del coseno entre los dos vectores:

$$S_s = \frac{s_1.s_2}{||s_1||.||s_2||}.$$

3.4.6.4. Similaridad de orden entre frases

Consideremos dos frases T_1 y T_2, por ejemplo:

- **T1**: A quick brown dog jumps over the lazy fox.
- **T2**. A quick brown fox jumps over the lazy dog.

Estas frases tienen exactamente las mismas palabras, pero dos de ellas ("fox" y "dog"), están ubicadas en orden inverso. Como las dos frases contienen las mismas palabras, un método basado en Bag of Words, dará como resultado que las frases son idénticas, pero para un intérprete humano es claro que las frases sólo son similares en hasta cierto punto. Por lo cual, un método computacional de similaridad entre frases debe tener en cuenta el orden de las palabras. En el ejemplo, el conjunto de palabras comunes es:

```
T = {A quick brown dog jumps over the lazy fox}.
```

Para lograr esto, este método asigna un índice único por cada palabra en T_1 y T_2, en orden, armando un arreglo de palabras. Un vector de orden r es formado para T_1 y T_2, respectivamente, basado en el conjunto de palabras comunes. Tomando T_1 como ejemplo, por cada palabra w_i en T, se intenta encontrar la palabra más similar en T_1 de la siguiente forma:

1. Si la misma palabra está presente en T_1, la entrada por la palabra en r_1 será el índice de la misma en T_1. Caso contrario, se intenta buscar la palabra más similar $\widetilde{w}_i$ semánticamente.

2. Si la similaridad entre w_i y $\widetilde{w}_i$ es mayor al umbral, la entrada de w_i en r_1 es completada con el índice de $\widetilde{w}_i$ en T_1.

3. Si las dos búsquedas anteriores fallan, la entrada de w_i en r_1 es 0.

Aplicando este procedimiento a las frases de ejemplo, tenemos:

$$r_1 = \{\ 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ \},$$

$$r_2 = \{\ 1\ 2\ 3\ 9\ 5\ 6\ 7\ 8\ 4\ \}.$$

Se propone entonces una medida de similaridad de orden entre frases de la siguiente manera:

$$S_r = 1 - \frac{\|r_1 - r_2\|}{\|r_1 + r_2\|},$$

esto significa que la similaridad de orden de palabras es determinada por la diferencia normalizada del orden de palabras.

3.4.6.5. Similaridad total entre frases

La similaridad semántica y la información sintáctica (en términos del orden de las palabras) convergen en el significado de la oración. Entonces, la similaridad total entre frases es definida como la combinación de la similaridad semántica y la similaridad del orden de palabras:

$$S(T_1, T_2) = \delta S_s + (1 - \delta) S_r,$$

$$S(T_1, T_2) = \delta \frac{s_1 . s_2}{\|s_1\|\ \|s_2\|} + (1 - \delta) \frac{\|r_1 - r_2\|}{\|r_1 + r_2\|},$$

donde $0 \leq \delta \leq 1$ decide la contribución relativa de cada una de las medidas de similaridad. Como el orden de palabras cumple un rol subordinado, se recomienda que δ tenga un valor mayor a 0,5.

3.5. Ensamble de Clustering

3.5.1. Clustering

El Clustering o *análisis cluster* tiene por objetivo agrupar elementos en grupos homogéneos en función de las similitudes o similaridades entre ellos. Normalmente se agrupan las observaciones, pero el Clustering puede también aplicarse para agrupar variables. Estos métodos se conocen también con el nombre de *métodos de clasificación automática o no supervisada*, o de reconocimiento de patrones sin supervisión (Peña, 2013). Con esta información de clasificación a mano, se puede inferir las propiedades de un objeto específico, basándose en la categoría que pertenece. Básicamente, los sistemas de clasificación son supervisados o no supervisados, dependiendo si asignan nuevos objetos de datos a uno o a un número finito de clases supervisadas discretas o categorías no supervisadas, respectivamente (Xu y Wunsch, 2008).

3.5.1.1. Definición de Cluster

Los algoritmos de clustering agrupan objetos de datos (patrones, entidades, instancias, observaciones, unidades) en un cierto número de clusters (grupos, subconjuntos o categorías).

En (Everitt y Leese, 2001) se presenta la siguiente definición: "un cluster es un conjunto de entidades que son similares, y entidades de clusters diferentes no son similares". Claramente, esta similaridad y disimilaridad debe ser definida con un criterio significativo, tales como las medidas de proximidad comentadas anteriormente.

3.5.1.2. Algoritmos de Clustering

Los algoritmos de clustering están generalmente clasificados en *clustering particional* y *clustering jerárquico*, basándose en la forma que se generan los clusters (Xu y Wunsch, 2008). Los algoritmos de clustering jerárquico encuentran clusters anidados recursivamente en un modo aglomerativo, esto es, empezando con cada uno de los puntos de datos como si fueran clusters y combinándolos con el más similar, sucesivamente, para formar de este modo una jerarquía de clusters; o bien en modo divisivo (de arriba hacia abajo –en inglés, top-down–), esto es, empezando con todos los puntos de datos en un cluster y dividiendo recursivamente cada

uno de los clusters en clusters más pequeños. Los algoritmos de clustering particional intentan encontrar todos los clusters simultáneamente como una partición de datos, y no imponen una estructura jerárquica; la base matemática es simple: se parte del concepto de *partición*, donde no existen subconjuntos solapados y la unión de los subconjuntos es exactamente el conjunto total. La entrada de un algoritmo jerárquico es una matriz de similaridad $n \times n$, donde n es el número de objetos a ser analizados por medio de clustering. Por otro lado, los algoritmos particionales puede usar tanto una matriz $n \times d$, representando n objetos en un espacio d-dimensional, o bien, como un algoritmo jerárquico, una matriz $n \times n$ (Jain, 2010).

En este trabajo, nos centraremos en algoritmos particionales, utilizando medidas de proximidad de texto para determinar cuáles objetos (cadenas de texto) pertenecerán al mismo cluster y cuáles no. Algunos ejemplos de este tipo de algoritmo son el ampliamente utilizado *k-medias* (MacQueen et al., 1967), y también el *Particionamiento Alrededor de Medoids (PAM)* o *k-medoids*. Este último se utilizará en este trabajo, por lo cual es descrito a continuación.

3.5.1.3. Particionamiento Alrededor de Medoides (PAM)

Particionamiento Alrededor de Medoides (PAM) es un algoritmo de Clustering particional, cuyo funcionamiento es similar al del k-medias, pero en lugar de tomar un conjunto de datos aleatorio como centroides y luego, iterativamente, calcularlo como medias de los clusters, se utilizan puntos, llamados *medoides*, "representativos" de un cluster, y tienen la particularidad de estar más céntricamente ubicados en el mismo (RDUSSEEUN, 1987). A diferencia de los centroides del algoritmo k-medias, que son vectores que pueden tomar valores arbitrarios en el espacio d-dimensional de elementos de entrada, los medoides son puntos que representan a un subconjunto de los mismos elementos. En este caso no se trabaja con otros datos fuera del conjunto de datos de entrada.

PAM tiene dos etapas, denominadas BUILD (construcción, etapa de formación de los clusters) y SWAP (intercambio, etapa de refinamiento de la partición obtenida). Nos centraremos en la etapa BUILD, ya que será la utilizada en este trabajo:

1. Establecer el medoide inicial m_1 como el vector de características del conjunto de datos de entrada que minimiza la distancia a todos los demás

datos en X.

$$m_1 = arg \min_{\forall x_i} \sum_{j \neq i} d(x_i, x_j),$$

donde $d(x_i, x_j)$ es una medida de distancia genérica.

2. Considerar un vector previamente no seleccionado x_i.

3. Considerar un vector previamente no seleccionado x_j y calcular la diferencia entre su distancia al medoide previamente seleccionado m_1, y su distancia al vector x_i.

4. Calcular la Contribución $C(x_i, x_j)$ del vector x_j a la selección del vector x_i, como:

$$C(x_i, x_j) = max(d(x_j, m_1) - d(x_j, x_i), 0).$$

5. Calcular la Ganancia total obtenida a partir de la selección del vector x_i como:

$$G(x_i) = \sum_j C(x_j, x_i).$$

6. Establecer el siguiente medoide como el vector previamente no seleccionado que maximiza la ganancia total, esto es:

$$m_l = arg \max_{\forall i} G(x_i).$$

7. Repetir los pasos 2 al 6 hasta llegar a obtener k medoides.

El método PAM puede aceptar una matriz de distancias como entrada, lo cual lo hace adecuado para utilizar las salidas de los métodos anteriormente mencionados. Como desventaja, se puede mencionar que su complejidad temporal es cuadrática, lo que se traduce como poca eficiencia en grandes conjuntos de datos, sin embargo, se han desarrollado mejoras sobre PAM para estos problemas de desempeño (Kaufman y Rousseeuw, 2009).

3.5.2. Ensamble de clustering

Distintos algoritmos de clustering producen distintos grupos de objetos, en forma de particiones de datos. Incluso el mismo algoritmo con distintos parámetros también produce distintos grupos de datos de salida (Xu y Wunsch, 2008). Además, las combinaciones de diferentes representaciones de datos que no pueden ser localizadas en un espacio dimensional, pueden generar particiones de datos

completamente diferentes. El Ensamble de Clustering, propone un método para extraer clusters consistentes dadas particiones variadas de entrada.

Dado un conjunto de datos de n objetos o patrones y d dimensiones, se aplican distintas técnicas de clustering a los mismos. Luego, usando Clustering de Acumulación de Evidencias (EAC), cada partición es vista como una evidencia independiente de organización de datos, las cuales se combinan, generando una nueva matriz de similaridad $n \times n$ entre los n patrones. Esta matriz poseerá una similaridad para cada par de elementos que será tanto más grande cuanto más veces dichos elementos participen en el mismo cluster para las sucesivas evidencias encontradas. La partición de datos final entre los n patrones es obtenida aplicando un algoritmo de clustering en la matriz de similaridad (Fred y Jain, 2005).

Debido a la existencia de muchos algoritmos de clustering, y a la posibilidad de poder variar los mismos con distintos parámetros de entrada, es difícil encontrar un algoritmo o variación del mismo que produzca los resultados esperados, funcione con distintas formas de clusters y sea adecuado para distintos conjuntos de datos. Por otra parte, esta variabilidad puede ser aprovechada para encontrar una estructura inter-patrón que represente a todas las técnicas tenidas en cuenta como entrada. Además, este método muestra que la combinación de distintos algoritmos puede resultar en la identificación de clusters subyacentes con formas, tamaños y densidades arbitrarias.

3.5.2.1. Definición formal

Dado $X = \{x_1, x_2, ..., x_n\}$ un conjunto de n objetos, y dada $\chi = \{x_1, x_2, ..., x_n\}$ como la representación de esos patrones; x_i puede ser definida por ejemplo sobre un espacio d-dimensional $x_i \in \mathbb{R}^2$, en el caso que adopte una representación vectorial, o $X = x_{i1}, x_{i2}, ..., x_{im_i}$ en el caso que adopte una representación de cadena de texto, donde m_i es la longitud del mismo.

Un algoritmo de clustering toma como entrada y organiza los n patrones en k clusters, respondiendo a algunas medidas de similaridad entre los patrones, formando una partición de datos P. Diferentes algoritmos de clustering producirán, en general, distintas particiones para el mismo conjunto de datos, tanto como en términos de número de patrones en un mismo cluster o en la cantidad de clusters. También es posible producir distintos resultados con el mismo algoritmo

de clustering, ya sea, variando los parámetros de entrada o explorando distintas representaciones en los patrones (Fred y Jain, 2005).

Considerando N particiones de datos X, y $\mathbb{P}$ la representación de las N particiones, definimos Ensamble de Clustering como:

$$\mathbb{P} = \{\mathrm{P}^1, \mathrm{P}^2, ..., \mathrm{P}^\mathrm{N}\},$$

$$P^1 = \{C_1^1, C_2^1, ..., C_{k1}^1\},$$

.

.

$$P^N = \{C_1^N, C_2^N, ..., C_{kN}^N\},$$

donde C_j^i es el cluster número j en la partición de datos P_i, la cual tiene k_i clusters, y n_j^i es la cardinalidad C_j^i.

El problema es encontrar la partición de datos "óptima", P^*, usando la información disponible en N particiones de datos $\mathbb{P} = \{\mathrm{P}^1, \mathrm{P}^2, ..., \mathrm{P}^\mathrm{N}\}$. Definimos k^*, como el número de clusters en P^*. Idealmente, P^* debe satisfacer las siguientes propiedades:

1. Consistencia con el Ensamble de Clustering $\mathbb{P}$.
2. Robustez con pequeñas variaciones de $\mathbb{P}$.
3. Bondad de ajuste con información de verdad de base (verdaderos clusters o patrones), si está disponible.

3.5.2.2. Acumulación de Evidencias

La idea del Clustering de Acumulación de Evidencias es combinar los resultados de múltiples ejecuciones de clustering dentro de una misma partición de datos viendo cada uno de esos resultados como una evidencia independiente de la organización de los mismos (Fred y Jain, 2005). Para realizar esto, es necesario seguir los siguientes pasos:

Producir ensambles de Clustering Los ensambles de Clustering generalmente son producidos desde dos enfoques: 1. la elección de la representación de datos y 2. la elección de los algoritmos de clustering o los parámetros de los mismos. Ya que la representación de datos de este trabajo es fija, se va a poner énfasis en el segundo enfoque. En el segundo enfoque, es posible generar ensambles de Clustering (I) aplicando diferentes algoritmos de Clustering, (II) usando el mismo algoritmo pero con diferentes parámetros o inicializaciones y (III) explorando diferentes medidas de similaridad entre los patrones, dado un algoritmo de Clustering.

Desde una perspectiva computacional, los resultados de algoritmos de Clustering son independientes, lo cual permite que sean ensamblados de forma distribuida. Además, los mismos pueden ser reusados, facilitando así un análisis de datos eficiente.

Combinar evidencias Para poder trabajar con diferentes particiones con diferentes cantidades de clusters, se propone un nuevo mecanismo para combinar resultados que deriva en una nueva medida de similaridad entre patrones. Se supone de antemano, que los patrones que pertenecen a un cluster *natural* son muy probables de ser colocados en el mismo cluster, incluso a lo largo en diferentes particiones de datos. Tomado las co-ocurrencia de pares de patrones en el mismo cluster, las N particiones de datos para n patrones, son mapeadas en una *matriz de co-asociación* $n \times n$:

$$C(i,j) = \frac{n_{ij}}{N},$$

donde n_{ij} es el número de veces que el par de patrones (i,j) es asignado al mismo cluster entre las N particiones de datos.

Entonces, el mecanismo de acumulación de evidencias mapea las particiones en el ensamble de Clustering dentro de una nueva medida de similaridad entre patrones (representada por cada elemento de la matriz de co-asociación), realizando, intrínsecamente, una transformación desde el conjunto original a la nueva representación de datos.

Capítulo 4. **Problema de investigación y propuesta**

4.1. Hipótesis de trabajo

A partir del relevamiento del estado del arte se establece la hipótesis de que los algoritmos de cálculo de similaridad de texto en sitios de CQA, con el fin de participar del proceso inherente a la aplicación de sistemas de recomendación con gran volumen de datos, pueden ser mejorados en cuanto a medidas de rendimiento y de desempeño si se aplica un método de ensamble de clustering mediante una arquitectura Big Data apropiada.

Por tal motivo, y como respuesta a la hipótesis planteada, se presenta el desarrollo de un nuevo método de cálculo de similaridad de texto basado en una arquitectura Big Data. Este método aprovecha las características de adimensionalidad y variabilidad de datos propias del ensamble de clustering. El método se aplica a un gran conjunto de datos reales con el fin de verificar la eficiencia y eficacia del procedimiento. Asimismo, se realiza un análisis comparativo del método presentado con los algoritmos para cálculo de similaridad de texto del estado del arte.

4.2. Metodología de investigación

Este trabajo comienza con una búsqueda de material científico relacionado a los siguientes temas: RS en general, RS no personalizados basados en contenido de texto, aplicación de RS en sitios de CQA, algoritmos de comparación de texto del estado del arte y su aplicación a grandes volúmenes de datos mediante métodos de ensamble de clustering y, también, una evaluación de arquitecturas de software adecuadas para un enfoque Big Data e infraestructuras acordes. Esta búsqueda se realizó mediante sitios o librerías digitales, tales como Google

Scholar[34] , IEEExplore Digital Library[35] , ScIELO[36] , Harvard Library[37] o el portal del CAICYT-CONICET[38] , entre otros.

Una vez definida la hipótesis correctamente y el plan de trabajo, se inició el desarrollo de un software de código abierto partiendo del proyecto "text comparison"[39] perteneciente al repositorio Git del departamento de Ingeniería en Sistemas de Información de la UTN FRRo. Se importaron las piezas de software del código del proyecto del estado del arte recientemente mencionado para usarlas mediante un enfoque Big Data, con nuevas herramientas basadas en Cloud Computing, Hadoop y una arquitectura de software completamente nueva que optimice este tipo de desarrollo. Una vez iniciado desarrollo del proyecto, se realizó una evaluación de las distintas opciones de herramientas y entornos que fueron utilizados. Esto incluye:

- Lenguajes de programación y librerías inherentes al mismo.
- Almacenes de datos, frameworks y proyectos de terceros para ser incorporados en la arquitectura Big Data.
- Arquitecturas de software, patrones, modelos y buenas prácticas.
- Infraestructura: local, distribuida en una red de computadoras físicas, o distribuida y virtualizada en la nube.

Paralelamente al desarrollo, se identificó y documentó la nueva solución de acuerdo con los requerimientos de la Maestría en Ingeniería en Sistemas de Información de la UTN FRRo, a fin de obtener un trabajo de investigación de tesis de maestría de excelencia, y acorde con los parámetros que caracterizan a la institución.

Por último, una vez finalizado el desarrollo, se construyó un registro con los indicadores resultantes, se validó la propuesta, se realizó el análisis comparativo con el estado del arte, se explicitaron los resultados obtenidos y se elaboraron las conclusiones, a fin de abrir y/o profundizar en nuevas líneas de investigación.

[34] Google Scholar: `https://scholar.google.com.ar`. Último acceso Febrero 2021.
[35] IEEExplore Digital Library: `http://ieeexplore.ieee.org`. Último acceso Febrero 2021.
[36] SciELO: `http://www.scielo.org`. Último acceso Febrero 2021.
[37] Harvard library: `https://library.harvard.edu`. Último acceso Febrero 2021.
[38] Centro Argentino de Información Científica y Tecnológica del CONICET: `http://www.caicyt-conicet.gov.ar/sitio`. Último acceso Agosto Febrero 2021.
[39] Repositorio GitHub: `https://github.com/Departamento-Sistemas-UTNFRRO/text_comparison`.

4.3. Método propuesto

Como alternativa para mejorar la medidas de comparación entre datos de texto en sitios de CQA bajo una arquitectura Big Data se propone en este trabajo el método EQuAL (*Ensemble method for community Question Answering sites based on cLustering*), basado en un método de ensamble de clustering de acumulación de evidencias, que puede ser utilizado para la generación de matrices de similaridad que sirven como entrada para un RS en un sitio de CQA. Este método está basado en una arquitectura Big Data distribuida y tiene en cuenta diversas distancias de texto, combinadas mediante un método de ensamble de clustering.

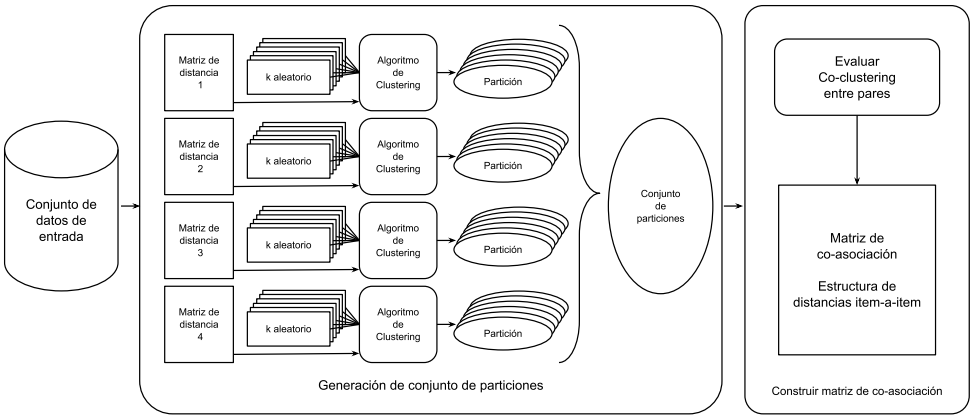

Figura 6: Método EQuAL para la generación de matrices de co-asociación desde el conjunto de datos original.

El desarrollo para este método está basado en dos pasos: (I) la generación de conjunto de particiones y (II) la construcción de matriz de co-asociación. La Figura 6 muestra un ejemplo del método, en el cual la primera etapa está compuesta de 4 matrices de distancias procesadas a través de un algoritmo de clustering, formando un conjunto de particiones, para luego, en la segunda etapa, ser ensambladas para formar la matriz de co-asociación. Veremos este proceso en detalle a continuación.

4.3.1. Generación de conjunto de particiones

El primer paso es la generación de un conjunto de particiones. El procedimiento comienza aplicando los distintos algoritmos de medidas de similaridad de texto del estado del arte al conjunto de datos de entrada. Por cada algoritmo, este procedimiento tiene como resultado una matriz de similaridad. Cada una

de las matrices de similaridad es el resultado de la combinación de todas las preguntas (individuales) de un muestreo original.

Por cada matriz de similaridad, se aplican c corridas de algoritmos de clustering, cada uno con un número k de elementos seleccionados al azar[40] . Esta combinación de n matrices y c ejecuciones del proceso de clustering, resulta en $N = n \times c$ salidas del algoritmo de clustering elegido como resultado. Cada salida consiste en una partición[41] P donde cada elemento consiste en la asignación de una pregunta individual a un cluster específico.

Con el fin de resumir la estructura de todas las particiones generadas por los algoritmos de clustering, se unen las particiones anteriormente obtenidas, dando lugar a un conjunto de particiones $\mathbb{P}$ de la siguiente manera:

$$\mathbb{P} = \{\mathrm{P}^1, \mathrm{P}^2, ..., \mathrm{P}^\mathrm{N}\},$$

donde N es el número de particiones que conforman el conjunto que será la entrada del procedimiento de construcción de la matriz de co-asociación.

4.3.2. Construcción de la matriz de co-asociación

El segundo paso es construir una matriz de co-asociación a partir del conjunto de particiones $\mathbb{P}$. Para tal fin, se aplica un algoritmo de ensamble de clustering de acumulación de evidencias, que combina cada una de estas particiones, dando como salida una matriz de co-asociación. Cada valor i, j perteneciente a la matriz de co-asociación consiste en la proporción de veces que los elementos i, j caen juntos en el mismo grupo de la salida de clustering, a lo largo de las $N = n \times c$ particiones.

La matriz de co-asociación, que es una representación integrada de las relaciones subyacentes entre los datos originales, es la entrada propuesta como nueva medida de distancia para datos de texto en sitios CQA para construir RS. Esta medida tiene la característica de ser adimensional, ya que consiste en una proporción respecto del total de datos, y además considera toda la variabilidad propia de los algoritmos de clustering utilizados para su construcción, por lo cual toma en cuenta la estructura de distancia ítem-ítem que es necesaria como

[40] Recordar de la Sección 3.5.1 que el número k es un parámetro de entrada del algoritmo de clustering que representa el número de clusters que se generarán.

[41] Recordar de la Sección 3.5.1.2 el concepto de partición en el contexto de clustering.

entrada para un RS basado en contenido incorporando varios aspectos de las distancias entre elementos de texto, en lugar de usar solo una simple medida basada individualmente en aspectos de cada una de las medidas de distancia.

El armado de matrices, la combinación de las mismas y la aplicación de estrategias para su procesamiento, implica un aumento significativo del volumen de datos y requiere una capacidad de cálculo intensiva. Una arquitectura Big Data que realice el procesamiento distribuido de los mismos es fundamental para este proceso. Además del volumen de datos con el cual se trabajará, se varían distintos parámetros, tales como la medida de similaridad y valores de umbral involucrados en procesos de clustering, con el fin de obtener resultados confiables. Esta característica redunda en múltiples ejecuciones de toda la solución. En pos de lograr una solución eficiente, cabe destacar que se implementaron experimentos basados en una infraestructura MapReduce aplicados con frameworks basados en Hadoop y cluster computing, lo cual provee la ventaja de procesar grandes cantidades de datos en instancias dinámicamente escalables, logrando cumplir con los preceptos de velocidad, variedad y veracidad del Big Data.

4.4. Arquitectura de procesamiento de datos

El procesamiento de datos se realizó con una arquitectura que permite el procesamiento distribuido en un cluster de computadoras, posibilitando que las operaciones de cómputo intensivo (ya sea cálculo de distancia o ensambles de clustering) se realicen en distintos nodos al mismo tiempo. El conjunto de datos de entrada es convertido a una colección distribuida y dividida en P particiones de datos lógicos (no confundir con las N particiones de datos de salida del método de Ensamble de Clustering). Cada una de esas particiones es asignada a un nodo del cluster, que realiza el cálculo de distancia con un algoritmo determinado. La Figura 7 muestra los componentes y procesos involucrados en la arquitectura de procesamiento de datos propuesta: el conjunto de datos original es procesado de forma distribuida generando las matrices de similaridad correspondientes a cada una de las técnicas, luego un algoritmo de clustering PAM es aplicado a cada una de ellas, para finalmente ensamblar todas las particiones en forma distribuida y así obtener la matriz de co-asociación.

Encontrar el número óptimo de particiones de datos lógicos no es una tarea fácil. Apache Spark, por ejemplo, asigna automáticamente un número de

particiones teniendo en cuenta la infraestructura del cluster y el tamaño de los archivos a procesar. Teniendo en cuenta un tamaño de archivo fijo, un cluster con 24 ejecutores podría asignar 24 particiones de datos, las cuales se procesarán en paralelo (la cantidad de ejecutores depende de la infraestructura del cluster Spark). Esto es algo relevante cuando hablamos de paralelismo en software; las tareas realizadas en distintos nodos/núcleos del cluster utilizan un *paralelismo basado en datos*, es decir que la ejecución simultánea se basa en ejecutar la misma función en todos los nodos e hilos al mismo tiempo, con distintas particiones de datos. A diferencia de lo anterior, en las tareas habituales de desarrollo de software se utiliza el denominado *paralelismo de tareas*, el cual consiste en la ejecución de distintas funciones (o una concatenación de funciones) en distintos hilos de ejecución, generalmente, con distintos conjuntos (enteros) de datos en cada uno. En este trabajo utilizaremos las dos estrategias, dependiendo cual sea más conveniente según la naturaleza de la tarea y los conjuntos de datos involucrados.

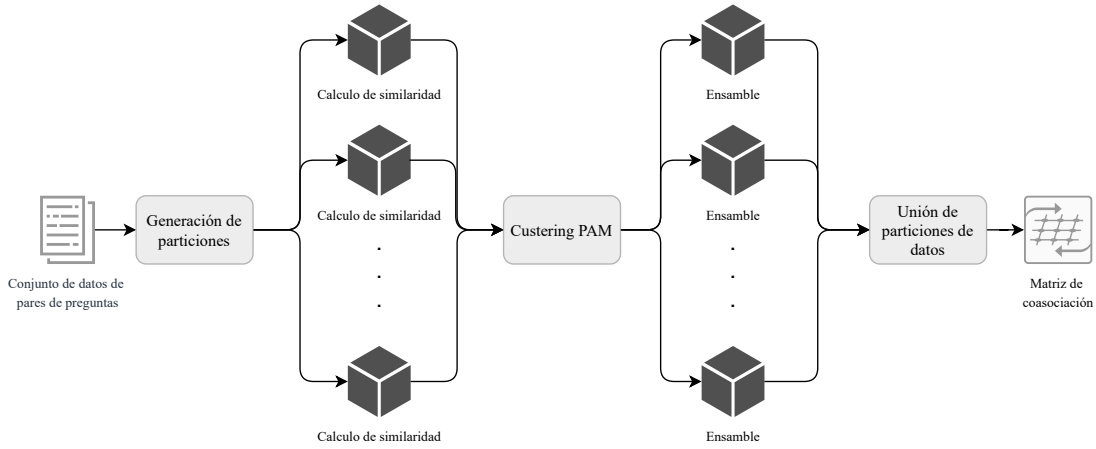

Figura 7: Infraestructura de la solución para generar una matriz de co-asociación para un RS.

Como se mencionó anteriormente, la generación de las matrices de distancia se realiza de forma paralela, particionando el conjunto de datos de entrada y luego juntando los resultados para construir cada una de las matrices. Una vez que las matrices de similaridad son generadas en la memoria del nodo maestro se procede a realizar el algoritmo de clustering en el mismo, paralelizando las distintas ejecuciones con el objetivo de optimizar los recursos de hardware y reducir el tiempo de ejecución. Las razones por la cual se eligió esta estrategia (en lugar de paralelizar las ejecuciones en todo el cluster de computadoras), son las siguientes:

- No es posible dividir la matriz de similaridad y aplicar el algoritmo de clustering a lo largo de todos los ejecutores, debido a la naturaleza del cálculo de Clustering. Por el contrario, Apache Spark provee la funcionalidad de "replicar" un conjunto de datos y enviar tareas a un ejecutor en particular (en un nodo en particular), para luego recuperar el resultado. Por lo cual, se utilizaría paralelismo de tareas en cada ejecutor, lo que provocaría un consumo excesivo de memoria en el momento de realizar experimentos en forma local. Sin embargo, es una mejora que podría ser implementada a futuro.

- Evitar *data-shuffling* excesivo entre nodos de datos. El data-shuffling es una operación que simplemente re-organiza y re-distribuye datos entre las particiones de datos apropiadas (Zhang et al., 2012). Esto es un problema cuando se realiza de forma excesiva en un cluster de computadoras, provocando demasiada transferencia de datos entre cada uno de los nodos. Por ejemplo, antes de aplicar una función personalizada, puede ser necesario realizar un ordenamiento local en cada partición, re-particionar los datos en cada una de las computadoras para luego redistribuir las mismas dependiendo, por ejemplo, en una clave primaria. Dicho esto, queda en evidencia que este procedimiento conlleva un procesamiento de red y disco rígido (entrada/salida) que deriva en el incremento de tiempo de ejecución comparado con un ambiente que use una sola computadora y el intercambio de datos entre particiones se realice en la memoria local.

El algoritmo de clustering realizado en forma centralizada produce como resultado archivos que son persistidos (guardados) en el almacenamiento local, que posibilitan aplicar un paralelismo basado en datos en el momento que el ensamble de clustering es aplicado. El formato de salida de cada una de las ejecuciones del algoritmo de clustering, posibilita que sea muy simple aplicar una función personalizada de ensamble de clustering distribuyendo los datos entre todas las particiones del cluster de computadoras y obteniendo una así matriz de co-asociación.

4.4.1. Escalabilidad horizontal

Es sabido que una computadora con una mejor configuración puede realizar más tareas que una con una configuración más débil, pero eso tiene una limitante:

El costo de producir una computadora con buen rendimiento es alto. Por lo cual, la arquitectura elegida utiliza un cluster de computadoras, que trabajan en conjunto, para lograr un gran desempeño.

Este tipo de sistemas escalan de una forma casi lineal con el número de servidores utilizados, y esto es posible debido al uso de particionamiento de datos (Pokorny, 2013). La distribución de datos horizontal posibilita dividir los cálculos computacionales en diferentes tareas concurrentes. Esto no es posible de realizar fácilmente sin un algoritmo y una arquitectura que se adapten y sean pensados desde su concepción para este tipo de funcionalidad, tales como Map-Reduce y Hadoop.

4.4.2. Escalabilidad y complejidad temporal

En la etapa de cálculo de similaridad para los distintos algoritmos tenidos en cuenta para el ensamble de clustering, siendo n el número de pares de preguntas de una muestra, y $p = 2n$ el número de preguntas individuales, se calcula la matriz triangular con entrada p, tenemos la siguiente cantidad de cálculos:

$$\frac{p(p-1)}{2} = \frac{2n(2n-1)}{2} = 2n^2 - n.$$

Es decir, si el número de pares es n, se realizarán $2n^2 - n$ cálculos por cada algoritmo de similaridad ejecutado. Además, si, por ejemplo, utilizamos θ algoritmos de similaridad el número de cálculos de similaridad es aproximadamente $\theta(2n^2 - n)$. Esto significa que al hacer cálculos de similaridad par-a-par entre elementos, la *complejidad temporal* aumenta cuadráticamente, es decir $O(n^2)$[42] [43] . Siendo estrictos, $\theta(2n^2 - n)$. no sería apropiado, ya que cada uno de los diferentes algoritmos de similaridad tiene distinta complejidad interna y el tiempo varía entre cada uno de ellos, es por eso que la métrica de complejidad opta por eliminar constantes y multiplicadores a favor de la simplicidad.

Al terminar el cálculo de similaridad, la arquitectura prevista procede a realizar un algoritmo de clustering con los resultados. Otra vez, siendo n el número de ítems (preguntas individuales), k el número de clusters elegido e i el número

[42] La notación Big-O es el lenguaje y métrica que describe la eficiencia de un algoritmo. Simboliza una cota superior sobre el tiempo, basándose en una función en la cual la variable independiente puede cambiar dependiendo de la naturaleza del algoritmo (por ejemplo, tamaño de la estructura de datos o cantidad de iteraciones). (Cormen et al., 2009)

[43] La notación Big-O tiende a eliminar constantes y términos no dominantes, ya que solo se centra en indicar cómo el algoritmo escala.

de iteraciones, para un tiempo t para calcular la distancia entre dos preguntas, tenemos una complejidad de $O(nkit)$, suponiendo que todas las iteraciones son realizadas (peor escenario) y dejando de lado la etapa de optimización de medoides para el algoritmo PAM. Por otro lado, la arquitectura presupone que el cálculo de distancias se ha realizado en una etapa anterior y las mismas son almacenadas en una estructura de datos en forma de *tabla hash*[44] en la cual es posible obtener la distancias de dos elementos haciendo una búsqueda indexada mediante una función hash pre-calculada, por lo cual la complejidad temporal promedio en un escenario ideal será de $O(1)$ (constante), y la complejidad final de la etapa de clustering de $O(nki)$, y para N ejecuciones, obtendremos $O(Nnki)$.

Por último, la etapa de ensamble de clustering utiliza la combinación de las n preguntas individuales en las N ejecuciones del algoritmo de clustering, obteniendo $x = N \times n$ registros. Cada uno de esos registros son agrupados mediante una función de agregación *group by*, una función que tiene una complejidad conocida de $O(x \log(x))$. El conjunto de datos generado es combinado con sí mismo para obtener las intersecciones, es decir $O(x^2)$. Dado que el término x^2 escala más rápido que el término $x \log(x)$ es posible simplificar la complejidad temporal a solo $O(x^2)$, o también $O(N^2 n^2)$.

De forma simplificada, la complejidad total que ofrece el método propuesto es de $O(n^2 + Nnki + N^2 n^2)$, lo cual es una función cuadrática que depende de la cantidad de preguntas individuales n. Lo anterior deja en evidencia la importancia de utilizar sistemas distribuidos y paralelismo basado en datos, ya que si bien la cantidad de cálculos total no cambia, sería posible hacer que la cantidad de cálculos que cada ejecutor realice solo sea un subconjunto de la cantidad total. En un contexto ideal y como modelo simplificado, por ejemplo, si se cuenta con un cluster de computadoras de 8 ejecutores (que realiza solo una tarea cada uno) y cada uno de ellos procesa $n/8$ ($n_{nuevo} = n/número\_de\_ejecutores$) preguntas individuales, se realizarán $(n/8)^2 = n^2/64$ cálculos de similaridad en cada uno de ellos, es decir, la cantidad de cálculos total también se reduce de forma cuadrática. El ejemplo de la Figura 8, es el modelo simplificado anteriormente mencionado, esquematizado con fines puramente descriptivos, que supone paralelismo de datos perfecto, cada uno de los ejecutores con la misma configuración, realizando una sola tarea, y sin tener en cuenta tiempos de entrada/salida y re-

[44] Una tabla hash es una estructura de datos que asocia claves con valores. La operación principal que soporta de manera eficiente es la búsqueda mediante una función hash.

particionamiento de datos. Se muestra la curva temporal de 4 clusters distintos de computadoras, con 1, 2, 4 y 8 ejecutores.

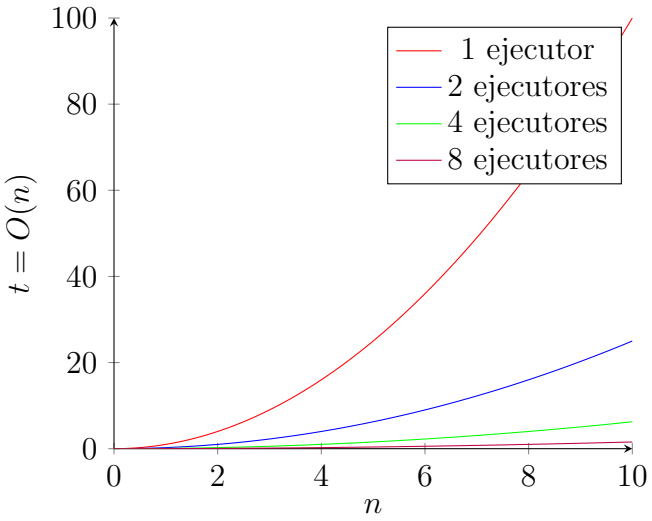

Figura 8: Simplificación de funciones de complejidad temporal según la cantidad de ejecutores del cluster de computadoras.

La unidad básica de Apache Spark es llamada tarea, y la cantidad de tareas depende del número de particiones del conjunto de datos de entrada. Cada una de las tareas se encuentra en un hilo de ejecución de una JVM[45] , dedicado exclusivamente a un *ejecutor*. Uno o más ejecutores son desplegados en los distintos *nodos* del cluster de computadoras (Janardhanan y Samuel, 2020). Es decir, que el paralelismo obtenido por la infraestructura Spark depende de los tres niveles: tareas, ejecutores y nodos; así como también de la naturaleza del algoritmo que se ejecuta en la misma. Con el fin de dar un ejemplo aplicable al cálculo de similaridad, se configura un cluster de computadoras de 4 ejecutores. Si cada uno de los ejecutores asigna 1-14 núcleos de CPU de forma dinámica, el número total de núcleos variará entre 4 y 56. Esto significa, que habrá de 4 a 56 tareas ejecutándose paralelamente en la infraestructura definida.

4.5. Ejemplo de implementación en un sistema de recomendación de tiempo real

Como punto de partida a esta sección, se aclara que este trabajo de tesis no cubre la implementación del RS propiamente dicho. No obstante, tal como

[45] Del inglés *Java Virtual Machine*, es una máquina virtual que permite ejecutar programas en código intermedio Java.

se plantea en la Sección **??**, persigue como objetivos la construcción de una arquitectura Big Data para ser aplicada a un gran volumen de datos reales de entrada de texto, y el desarrollo de un nuevo método sobre esta arquitectura, basado en ensambles de clustering, para implementar la matriz de similaridad de texto asociada al RS para sitios de CQA. Para tal fin, se propone en esta sección un ejemplo de RS en tiempo real, para brindar un contexto de aplicación real del presente trabajo. El ejemplo será desarrollado teniendo en cuenta tres casos de uso diferentes: (I) un proceso fuera de línea ejecutado periódicamente que actualizará la base de datos, caso de uso central de este trabajo y en el cual se sustenta toda la arquitectura propuesta, y que también puede pensarse como una parte integrante de la implementación propuesta en este apartado; (II) el punto de vista del usuario que consulta una pregunta en el sitio; y (III) el caso (opcional) en que el usuario agrega una nueva pregunta al sitio.

A continuación, entonces, veremos el desarrollo de un ejemplo acerca de cómo funcionarían en la práctica los tres casos de uso más frecuentes de aplicación que incorporan las ideas presentadas en esta Tesis.

El proceso completo del sistema de recomendación De acuerdo a lo mencionado en el apartado anterior, la propuesta ejemplo de un sistema de recomendación completo se puede analizar teniendo en cuenta tres casos de uso, el primero de los cuales se basa directamente en la propuesta de esta Tesis.

El primer caso de uso consiste en un proceso que se ejecuta fuera de línea. El mismo es el encargado de actualizar las relaciones de similaridad entre todas las preguntas existentes en el sitio, utilizando el método EQuAL propuesto en esta tesis para perseguir tal fin, sustentándose en la arquitectura Big data planteada. Además, este proceso es el responsable de entrenar los modelos de recomendación con cierta frecuencia. Este caso de uso utiliza las ideas presentadas en esta Tesis. El segundo caso de uso consiste en la consulta de una pregunta en el sitio de CQA. Este caso se nutre del caso de uso anterior, y utiliza sus resultados para obtener una lista de preguntas similares a la consultada. Por otro lado, si bien el procesamiento fuera de línea tiene en cuenta todas las preguntas existentes, no posee la capacidad de recomendar preguntas similares a una pregunta recién agregada al sistema en tiempo real (hasta que el mismo sea ejecutado nuevamente). Por lo anterior, el tercer caso de uso, que es opcional, se trata de una actualización en la que un usuario agrega una nueva pregunta al sitio, y la im-

plementación tecnológica correspondiente a partir de este caso para recomendar preguntas similares en tiempo real.

Entonces, se dará una propuesta de implementación de un RS completo, para dar contexto y tener una idea de funcionamiento en conjunto del mismo. En el mismo se incluye, como parte componente del primer caso de uso, el método propuesto en esta Tesis y su arquitectura subyacente.

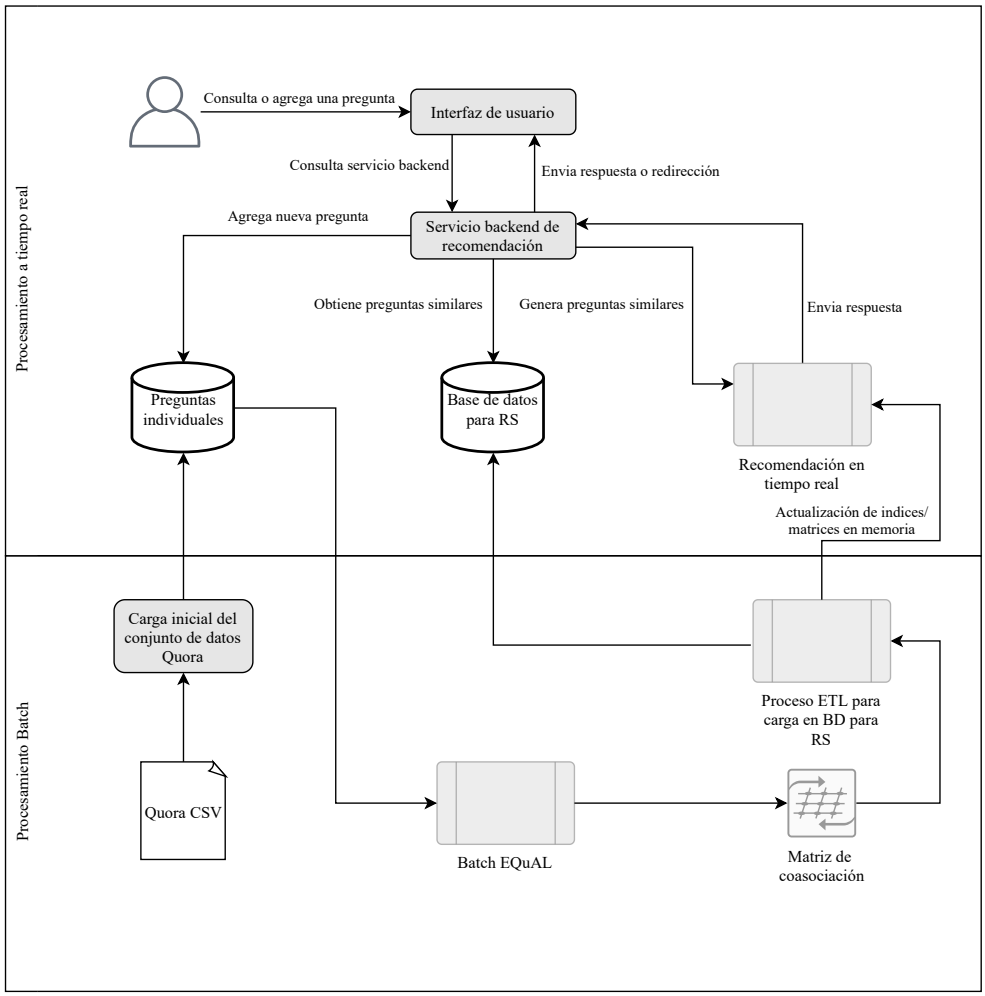

Figura 9: Arquitectura de un sistema de recomendación a tiempo real utilizando el método EQuAL.

4.5.1. Arquitectura general

La arquitectura del RS embebido en un sitio de CQA colaborativo, consiste básicamente en un microservicio dedicado a recomendación, un artefacto de software que realizará la función de recomendación a tiempo real de forma distribuida, una base de datos altamente escalable NoSQL, y un proceso fuera de línea que generará una actualización de la misma de forma periódica.

La Figura 9 muestra cómo interactúa la totalidad de componentes relativos al RS, divididos fundamentalmente en procesamiento a tiempo real y fuera de línea; y además, muestra todas las interacciones posibles entre los componentes. En los siguientes apartados, se explicará cada uno de los tres casos de uso dentro de este sistema.

4.5.2. Procesamiento fuera de línea

La Figura 10 muestra el procesamiento fuera de línea para actualización y carga inicial de la base de datos que contendrá la matriz de co-asociación en un formato adecuado para consulta. Este proceso se ejecutará periódicamente, teniendo como entrada al conjunto de datos (en este ejemplo, el de Quora) en una base de datos. Para nuestro caso, esta base de datos sería inicializada con el conjunto de datos utilizado en este trabajo de tesis en forma de archivos de texto. A continuación se detallará cada uno de los casos.

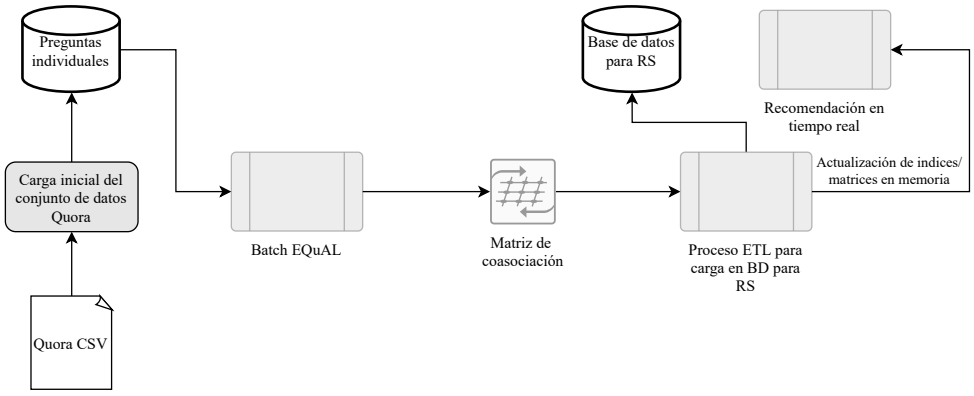

Figura 10: Arquitectura de un sistema de recomendación a tiempo real utilizando el método EQuAL.

Como punto de partida, se carga inicialmente una base de datos con el conjunto de datos Quora original. Esta base de datos podría ser relacional, o de otro tipo, pero debería tener dos características principales: rápida inserción de registros y debe permitir obtener todos los registros de una tabla mediante una sola consulta. La inserción rápida es importante en el caso del agregado de una nueva pregunta en tiempo real. Por otro lado, la capacidad de seleccionar todos los registros de una tabla tiene importancia para el proceso fuera de línea que utilizará el método EQuAL para la actualización periódica del RS.

De forma periódica, el proceso fuera de línea (en el cual se basa este trabajo) obtendrá todas las preguntas de la base de datos de preguntas individuales para

crear la matriz de co-asociación. La periodicidad podrá variar dependiendo de la configuración y la arquitectura en la cual se ejecuta. Mientras más rápido sea este proceso, podría ser programado con más frecuencia. Podría ejecutarse, por ejemplo, diariamente, en momentos de bajo tráfico en el sitio en cuestión. Por otro lado, este proceso será el encargado de entrenar, de forma ad-hoc, los modelos que así lo requieran, y generar nuevos corpus que se nutrirán de nuevas palabras provenientes de preguntas nuevas agregadas por los usuarios una vez que el sistema esté en funcionamiento.

Tabla 1: Matriz de co-asociación salida del proceso EQuAL.

question_id_1	**question_id_2**	**question_1**	**question_2**	**similarity**
1	2	question_desc_1	question_desc_2	0.34
1	3	question_desc_1	question_desc_3	0.67
1	4	question_desc_1	question_desc_4	0.92

Adicionalmente, un proceso ETL[46] , que tendrá como origen la matriz de co-asociación, cargará la base de datos que guardará la información de similaridad entre preguntas y es consultada por el servicio de recomendación. Las características que debe tener esta base de datos serán descritas en el apartado siguiente. El proceso ETL podrá ser lanzado mediante un evento que indique que la nueva matriz de co-asociación ha sido generada, o bien podría ser una extensión del proceso EQuAL. Un ejemplo de la entrada de este procesamiento puede ser el que se muestra en la Tabla 1, el cual muestra la matriz de co-asociación: las columnas *question_id_*1 y *question_id_*2 corresponden a los identificadores de preguntas, las columnas *question_*1 y *question_*2 corresponden al contenido de cada una de las preguntas, y la columna *similarity* indica cual es la similaridad entre las dos preguntas de cada una de las filas. Por ejemplo, la primera fila indica una similaridad de 0,34 entre las preguntas 1 y 2. Luego del proceso ETL, los registros de la base de datos para el RS tendrán la estructura que se muestra en la Tabla 2, que posee solo tres columnas, el ID de pregunta (*question_id*), su descripción (*question*), y la preguntas similares con su similaridad. Cada una de las preguntas similares está compuesta por un ID, la descripción y la similaridad con la pregunta correspondiente a la fila en cuestión (*similar_questions*). Esto permitirá consultar la pregunta mediante su ID de manera muy eficiente. Por otro lado, la carga de la tabla también será muy eficiente, ya que la cantidad

[46] Siglas para Extract, Transform and Load «extraer, transformar y cargar».

de registros se reduce a la cantidad de preguntas individuales, en lugar de la cantidad de registros de la matriz de co-asociación. La columna "similar_questions" será en formato JSON, lo cual permite acceder a datos con una estructura definida, y serializar y deserializar los mismos de una forma estándar.

Además, este proceso ETL podría ser utilizado para la actualización de índices o matrices en memoria utilizadas para un RS a tiempo real.

Tabla 2: Ejemplo de un registro de la base de datos de similaridad entre preguntas.

question_id	question	similar_questions

1	question_desc_1	<pre>[
		  {
		  "id": 4,
		  "question": "question_desc_4",
		  "similarity": "0.92"
		  },
		  {
		  "id": 3,
		  "question": "question_desc_3",
		  "similarity": "0.67"
		  },
		  {
		  "id": 2,
		  "question": "question_desc_2",
		  "similarity": "0.34"
		  }
		]</pre>

4.5.3. Consulta de una pregunta existente

En este apartado, se supone que la base de datos posee toda la información necesaria y está disponible; y además, que existe una aplicación web que consulta un servicio backend dedicado al sistema de recomendación, mediante una API REST[47] . Este último consultará la base de datos para devolver las preguntas similares a un ID de pregunta dado.

La Figura 11 muestra el flujo de la consulta de una pregunta existente. En el momento que un usuario consulta una pregunta en el sitio, el servicio frontend enviará una solicitud GET al servicio backend, utilizando el ID de la pregunta en cuestión. El servicio backend buscará la pregunta en la base de datos utilizando el ID de pregunta y retornando todas las preguntas similares. Las preguntas similares correspondientes pueden estar ordenadas por el valor de similaridad

[47] Representational State Transfer. Es una arquitectura y conjuntos de convenciones basada en servicios web para intercambios de información.

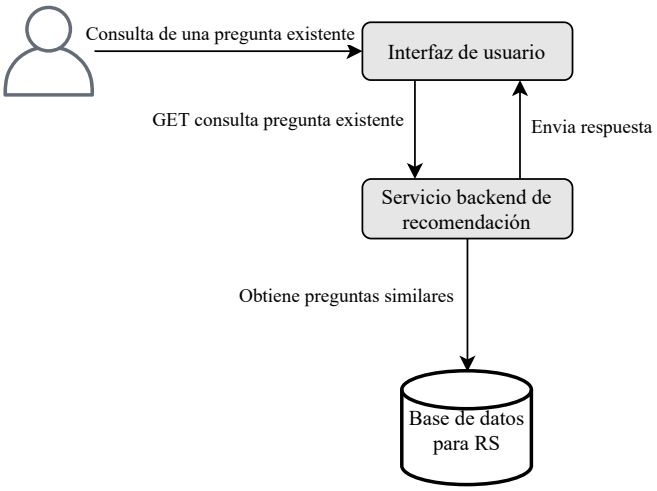

Figura 11: Flujo de consulta de una pregunta existente.

en forma decreciente. Esta operación debería ser ejecutada de manera rápida y eficiente, debido a que el servicio backend es dedicado al RS y la base de datos posee las siguientes características:

- La información de similaridad entre preguntas estará almacenada en una tabla con una clave primaria simple (PK), formada solo por el ID de pregunta. Esta estructura genera un índice por ID de pregunta, lo cual posibilita una búsqueda rápida.

- La tabla anteriormente mencionada podría estar particionada por la PK. Esto significa que se generan particiones de datos a lo largo de los diferentes nodos en el esquema de base de datos, basado en una función hash consistente y generada por el ID de pregunta. Esto es especial para búsquedas rápidas por PK, y además, facilita la escalabilidad horizontal y replicación.

- La información de preguntas similares para un ID dado, estará en formato JSON, haciendo que el servicio backend tenga un trabajo casi nulo para disponibilizar las preguntas similares al usuario final.

- Se recomienda una base de datos NoSQL ya que ellas vienen con un número de buenas prácticas arquitectónicas que afectan el rendimiento, tales como distribución de datos automática, baja latencia, consultas asíncronas, replicación de datos y alta disponibilidad.

4.5.4. Agregar una nueva pregunta

Esta funcionalidad es opcional y depende de los requerimientos del sitio y las funciones de recomendación a desarrollar, ya que recomendar ítems instantáneamente para un ítem recién agregado a un RS es un gran desafío para los arquitectos de software. Cabe aclarar que esta funcionalidad es opcional ya que, si los requerimientos del sistema no necesitan de la misma, se podría esperar a la ejecución del proceso fuera de línea, de tal forma que este asigne preguntas similares a la pregunta recién agregada. El propósito principal es realizar el cálculo en tiempo real de los ítems recomendados para el ítem recién agregado al sistema, hacerlos disponibles al usuario final, y que esto no signifique una sobrecarga del mismo. La recomendación en tiempo real será provisoria hasta que esta pregunta sea tenida en cuenta por el proceso fuera de línea y se encuentre disponible en la base de datos del RS, por lo cual esta funcionalidad puede ser utilizada como respaldo en el caso de que una pregunta no sea encontrada en dicha base de datos. La arquitectura propuesta a continuación, que se detalla en la Figura 12, propone el cálculo de similaridad a tiempo real, utilizando métodos derivados del método EQuAL que sean apropiados y tengan buen desempeño. Esto permitirá que no solamente los usuarios del sitio puedan tener acceso a las preguntas de manera inmediata, sino que estas también sean accesibles al mismo usuario que agregó la pregunta, evitando así, el tiempo de espera para que su pregunta sea respondida.

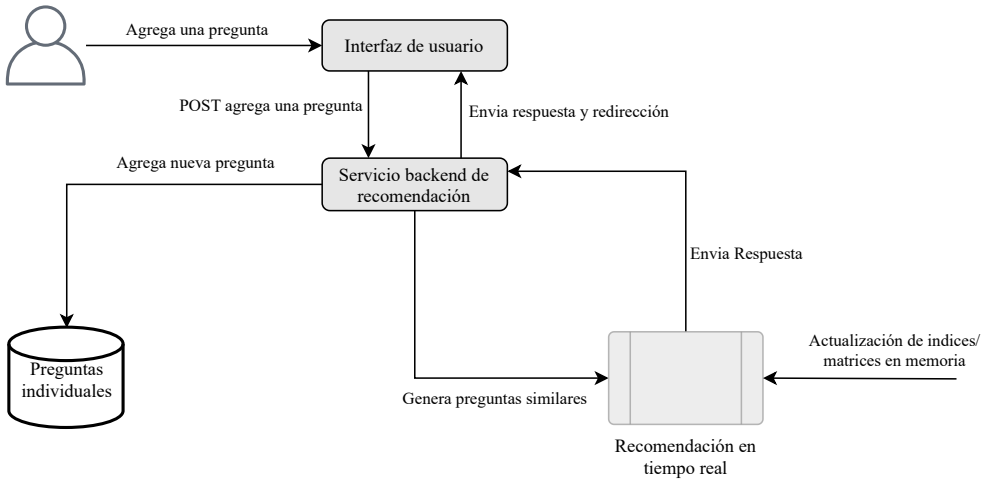

Figura 12: Flujo en el RS a tiempo real cuando se agrega una nueva pregunta al sistema.

Cuando se agrega una nueva pregunta, la información es enviada al servicio backend mediante una solicitud POST, el cual es enviado al módulo que realiza

el cálculo de recomendación a tiempo real. Este último retorna las preguntas similares para que estén disponibles al usuario. Luego de efectuar la pregunta, el usuario puede ser redireccionado a una URL que contenga la pregunta recién agregada y las preguntas similares. Por otro lado, y de manera asíncrona, el servicio backend guarda la pregunta recién agregada a la base de datos de preguntas individuales, para que sea tenida en cuenta en la próxima ejecución del proceso fuera de línea.

El módulo de recomendación en tiempo real podría realizar el cálculo de recomendación con distintos enfoques. A continuación, se mencionan propuestas, que quedan fuera del alcance de esta tesis, para implementar arquitecturas de buen desempeño utilizando enfoque de Big Data.

Cálculo de similaridad contra todas las preguntas del sitio En este enfoque, de ser posible, este módulo debería mantener la matriz de co-asociación en memoria, para poder realizar el cálculo a demanda. La pregunta recién ingresada debe compararse con todas las preguntas del sitio. Al mismo tiempo, se quiere tener control del tamaño de la matriz. Para tal fin, es posible mantener solo una fracción de la misma con los datos más importantes. Existen diversas técnicas para llevarlo a cabo, por ejemplo implementar un algoritmo de *Caché de Reemplazo Adaptativo* (ARC[48]) que descarta los objetos que no fueron usados recientemente y los objetos con menor frecuencia de uso. Por otro lado, las preguntas, como se describió anteriormente, deben tener un identificador único. Por lo cual, cada registro en el caché puede ser asignado a una pregunta. Con identificadores en el rango $0..N-1$ si N es la cantidad total de registros en el caché.

En cuanto al cálculo de similaridad, los modelos que lo requieran deben estar pre-entrenados y cargados en memoria. Por ejemplo, podría ser posible que una nueva pregunta realice una comparación con todas las preguntas en caché (fuerza bruta) utilizando cada uno de los algoritmos de similaridad. El proceso de ensamble de clustering puede ser simplificado a la media de los valores calculados por los algoritmos de similaridad subyacentes, para luego retornar las k preguntas más similares a la recién agregada.

[48] Por sus siglas en inglés *Adaptive Replacement Cache*.

Búsqueda de vecinos cercanos con representaciones vectoriales Este enfoque utiliza representaciones vectoriales de cada una de las preguntas individuales. Para tal fin, es posible que el método EQuAL que se ejecuta periódicamente genere una representación vectorial de cada una de las preguntas, utilizando uno de los métodos (o varios de ellos combinados) que usan este tipo de representación como paso intermedio del cálculo de similaridad. Con este listado de preguntas en forma de vector, es posible cargar un índice en memoria que permita la búsqueda de un conjunto de vectores inmediatamente.

Mediante una nueva pregunta agregada al sistema y su representación vectorial, es posible buscar sus k vecinos más cercanos. Este método de clasificación es llamado *KNN* (del inglés *k-Nearest-Neighbors*). Para un registro t a ser clasificado, sus k vecinos más cercanos son obtenidos en la forma de vecinos de t (Guo et al., 2003). Este método es utilizado por varias librerías reconocidas por la industria, tales como Annoy [49] , las cuales poseen un muy buen desempeño y realizan un uso muy eficiente de memoria.

Modelos de scoring Es posible utilizar el proceso ETL como parte del procesamiento fuera de línea para cargar un motor de búsqueda que contenga índices basados en scoring. Ciertas herramientas, tales como ElasticSearch[50] , tienen la capacidad de devolver documentos similares a uno consultado basándose en la generación de índices. El listado de preguntas puede ser tomado como entrada para la generación del mismo. Cada uno de los documentos es indexado. Los términos presentes en cada una de las preguntas (documentos) son descompuestos e indexados de forma invertida. Cada término es indexado, y contendrá, por ejemplo, en qué posición dentro de cada documento se encuentra. En el momento de una nueva consulta, se devuelven todos los documentos que coinciden con la misma. Con el fin de dar relevancia a los mismos, cada uno de los documentos tiene asignado un score que es calculado con el método TF-IDF (tal como se explicó anteriormente), es decir que un término que se encuentra varias veces en el documento consultado pero es contenido por pocos documentos en el índice, tendrá más relevancia que un término que aparece con más frecuencia en distintos documentos. Entonces, la relevancia de cada uno de los documen-

[49] Spotify Annoy GitHub Repository: `https://github.com/spotify/annoy`. Último acceso: Junio 2021.

[50] Sitio web ElasticSearch: `https://www.elastic.co/`. Último acceso: Junio 2021.

tos retornados, depende (en parte) del peso de cada término en el documento consultado.

4.5.5. Condiciones institucionales para el desarrollo de la tesis. Infraestructura y equipamiento

El presente trabajo se lleva a cabo en el marco del Proyecto PID UTN: Minería de Datos aplicado a problemáticas de Big Data de la Universidad Tecnológica Nacional, Facultad Regional Rosario[51] .

El tesista cursó la Maestría en Ingeniería en Sistemas de Información en dicha Facultad y contó con el equipamiento e instalaciones del Departamento en Ingeniería en Sistemas de Información. Con el objetivo de realizar el desarrollo tecnológico de este trabajo de tesis, el tesista utilizó equipos propios así como también los equipos de la universidad, cuando fue necesario. Los conjuntos de datos para la experimentación y validación de la solución propuesta están disponibles libremente en Internet, así como también el software y herramientas necesarias.

[51] Código del PID: SIUTNRO0005006. Bajo la dirección del Ing. Eduardo Amar.

Capítulo 5. **Experimentos**

Los experimentos descritos en este capítulo se encuentran en los siguientes recursos:

1. Repositorio GitHub `https://github.com/Departamento-Sistemas-UTNFRRO/big_data_text_similarity`, en la cuenta oficial de la UTN FRRo, ya que servirá de aporte para el departamento de investigación. El código está basado en Python y es utilizado para la generación y validación de matrices de co-asociación utilizando el método EQuAL en modo fuera de línea.

2. Repositorio GitHub `https://github.com/fedetesone/anova_equal`, perteneciente a la cuenta del tesista. El código está basado en R y es utilizado para la validación de rendimiento del método EQuAL.

3. Los datos generados por los experimentos, que sirven como entrada para el análisis de validación realizado a continuación, están disponibles en Google Drive[52] .

5.1. Estado del arte

El proyecto en el cual se realizaron los experimentos que este trabajo tiene como principal estado del arte, es un proyecto basado en código Python que se puede encontrar en el siguiente repositorio GitHub `https://github.com/Departamento-Sistemas-UTNFRRO/text_comparison`.

Este proyecto tiene como características principales la utilización de los 5 algoritmos de similaridad mencionados anteriormente (y evaluados a continuación) y la ejecución de cada uno de ellos en el marco de un patrón master-worker en un solo microprocesador. Este patrón es utilizado para el procesamiento paralelo,

[52] Datos resultados de los experimentos en `https://drive.google.com/drive/folders/1oY8NP1k7y0Yw2ZoNUMq22SV716tgL6wi?usp=sharing`

en el cual una tarea es enviada a cada uno de los "workers" para ser procesada. En este caso en particular, la cantidad de "workers" es fija y especificada como un parámetro en tiempo de ejecución.

5.1.1. Medidas de rendimiento y error

Utilizando cada par de preguntas del conjunto de datos original, se calculó la matriz de confusión para cada uno de los algoritmos de similaridad[53] . Como se puede ver en la Tabla 3.

Tabla 3: Matrices de confusión para los cinco algoritmos de medidas de similaridad.

			Predicho		**Exactitud**	**Error**
			0	**1**		
TF	**Real**	**0**	0.4355	0.1953	0.6776	0.3224
		1	0.1271	0.2421		
TF/IDF	**Real**	**0**	0.4477	0.1831	0.6685	0.3315
		1	0.1484	0.2208		
Word2Vec	**Real**	**0**	0.4343	0.1965	0.6788	0.3212
		1	0.1247	0.2445		
FastText	**Real**	**0**	0.5033	0.1275	0.6725	0.3275
		1	0.2	0.1692		
Semantic Distance	**Real**	**0**	0.4877	0.1431	**0.6797**	**0.3203**
		1	0.1772	0.192		

Los resultados porcentuales de las matrices de confusión se muestran en cada intersección de filas y columnas. Los resultados reales se muestran en las filas, teniendo un valor 0 cuando las preguntas son distintas y un valor 1 cuando las preguntas son iguales. Los resultados predichos se muestran en las columnas.

La exactitud obtenida a partir de las matrices de confusió —que se calcula como la suma de los resultados acertados entre los valores predichos y reales— en todos los casos excede el 66 %, alcanzando un máximo de 67,97 % para Semantic Distance. Por otro lado, con respecto al error promedio, se llega a un valor máximo de 33,15 % en el caso de TF/IDF.

Observando las matrices de confusión, se presenta un desbalance en los dos valores tomados para calcular la exactitud, siendo que los resultados obtenidos

[53] El método de validación será explicado con más detalle en la sección 5.6.2.

para la clase “0” considerablemente mejores que para la clase “1”. Por ejemplo, el desbalance más grande se encuentra en la matriz de confusión de FastText, el cual es 0,5033–0,1692 = 0,334. Este tipo de desbalance puede ser originado por la distribución del conjunto de datos de entrada (36,9 % pares de preguntas son clase 1 y el 63,1 % restante es clase 0), el cual pretende ser corregido con un método de muestreo que será explicado más adelante.

5.2. Preprocesamiento del conjunto de datos

La calidad de un conjunto de datos del mundo real depende del número de errores que contenga. Los errores de entrada de datos, tanto simples como complejos, son muy frecuentes, más allá de las validaciones que se les hayan realizado a los mismos (Maletic y Marcus, 2000). Además de los errores de datos, que es necesario eliminar, también es crucial preprocesar los mismos con el fin de generar un conjunto de datos homogéneo y así obtener mejores resultados.

Para el conjunto de datos de preguntas del sitio web Quora, se realizaron los siguientes trabajos de preprocesamiento:

1. Convertir el texto en minúscula; esto habilita a que las comparaciones de texto entre preguntas con algoritmos que son sensibles al cambio entre letras mayúscula y minúscula, sean efectivas.

2. Eliminar fórmulas; las cuales están encerradas entre etiquetas [math][/math] y [code][/code] ya que su análisis es muy complejo y contraproducente en términos de rendimiento.

3. Reemplazar números por letras; posibilita poner en el mismo plano preguntas que contienen números y otras que no lo hacen. Además, al utilizar palabras del inglés es posible que las mismas se encuentren en taxonomías para comparaciones semánticas.

4. Eliminar caracteres especiales, ya que los datos deben ser uniformes. Un signo de exclamación o de pregunta no cambiaría la semántica de la pregunta (desde la perspectiva del análisis de similaridad) y agregaría ruido al momento de procesarlas.

Para resumir, cada una de las preguntas será parámetro de una función que aplica las técnicas anteriormente mencionadas, tal como la que se muestra en el

Fragmento de codigo 1. El fragmento de código muestra un ejemplo de función de limpieza aplicada en código Python. La misma es definida en la línea 1, con un parámetro "text" de entrada. Este parámetro es enviado a las distintas funciones que realizan cada uno de los trabajos de preprocesamiento: transformación a minúsculas, eliminación de fórmulas, reemplazo de números, y eliminación de caracteres especiales, en las líneas 2, 3, 4, y 5 respectivamente. Luego, en la línea 7 se retorna el resultado con todas las transformaciones aplicadas.

```
1 def clean_text(text):
2         text = text.lower()
3         text = remove_formulas(text)
4         text = replace_numbers(text)
5         text = remove_special_delimiters(text)
6
7         return text
```

Fragmento de código 1: Ejemplo de función de limpieza.

Ejemplos de la salida de la función de limpieza, pueden ser:

```
i: How do I find the zeros of the polynomial function
[math]f(x)=\dfrac{1}{2}x^{3}-3x[/math]?
o: how do i find the zeros of the polynomial function

i: Would you switch from Canon 6D to Leica D-LUX 109?
o: would you switch from canon six d to leica d-lux one zero nine
```

En el primer ejemplo, es posible ver cómo se elimina la fórmula polinómica, mientras que en el segundo, todos los números fueron transformados a letras, con espacios entre ellos. En ambos, se puede ver que el signo de interrogación se elimina y se transforman todas las palabras a minúscula.

5.3. Muestreo del conjunto de datos

Se generaron muestras del conjunto de datos original en forma de subconjuntos del mismo. Cada uno de los subconjuntos, fue generado de forma pseudo-aleatoria utilizando la función sample del paquete random, de la librería Python. De tal forma, se generaron listas de distintos tamaños de elementos únicos seleccionados del total de la población de pares de preguntas.

Cada uno de los subconjuntos de muestreo tuvo un criterio de aceptación: la proporción de pares de preguntas iguales (con indicador 1) debería ser parecida a la proporción de preguntas distintas (con indicador 0); por lo cual, un subconjunto de muestreo es aceptado cuando la proporción de preguntas iguales está entre el 35 % y 65 % del total de preguntas del subconjunto. Esto garantiza que cada uno de los subconjuntos sea estadísticamente significativo y posea una variabilidad de datos tal que derive en resultados confiables.

5.4. Generación de particiones

5.4.1. Cálculo de similaridades

Se generaron particiones desde los subconjuntos de muestreos generados desde el conjunto de datos original. Teniendo en cuenta que se utilizó un algoritmo de clustering para desarrollar el método de este trabajo, fue necesario descomponer la estructura de los archivos de entrada —el cual contiene un identificador por cada par de preguntas— en preguntas individuales, con el fin de poder identificarlas unívocamente y utilizarlas como los objetos de entrada del análisis de clustering. La estructura de los subconjuntos de muestreo se clarifica en la Tabla 4. Cada una de las filas contiene un identificador secuencial *sequence_id*, el identificador de par de preguntas proveniente del conjunto de datos original *question_pair_id*, y las dos preguntas correspondientes *question_1* y *question_2*.

Tabla 4: Ejemplo de la estructura de los subconjuntos de muestreo.

question_pair_id	**question_1**	**question_2**
123004	question_0	question_2
98776	question_1	question_3

Luego, el conjunto el subconjunto de preguntas preparado para el cálculo de distancia fue generado de la siguiente forma:

1. Se crea una matriz con la unión de las columnas *question_1* y *question_2*, generando una fila por cada pregunta individual y un número secuencial que las identifica, como se muestra en la Tabla 5.

2. Se realiza una combinación de cada una de las filas contra la matriz en sí misma (eliminando resultados repetidos).

Tabla 5: Ejemplo de la estructura del conjunto de preguntas individuales de la muestra en curso.

sequence_id	**question**
0	question_0
1	question_1
2	question_2
3	question_3

Tabla 6: Combinación de todas las preguntas individuales de una muestra.

sequence_id_1	**question_id_1**	**sequence_id_2**	**question_id_2**
0	question_0	1	question_1
0	question_0	2	question_2
0	question_0	3	question_3
1	question_1	2	question_2
1	question_1	3	question_3
2	question_2	3	question_3

La Tabla 6 da un ejemplo de la combinación de todas las preguntas individuales de la muestra, utilizando el identificador de secuencia y el contenido de cada una de ellas. Esta estructura sirve de entrada para el cálculo de similaridad para cada una de las técnicas previamente mencionadas. Consecuentemente, es posible realizar un cálculo de similaridad entre las dos preguntas que pertenecen a una misma fila, y agregar esta información a la estructura de datos anterior. La Tabla 7 posee la misma estructura que la Tabla 6, con el agregado del valor de similaridad entre el par de preguntas de cada una de las filas, en la columna *similarity*.

Tabla 7: Ejemplo de la estructura de matriz de similaridad en formato de tabla.

sequence_id_1	**question_id_1**	**sequence_id_2**	**question_id_2**	**similarity**
0	question_0	1	question_1	similarity_01
0	question_0	2	question_2	similarity_02
0	question_0	3	question_3	similarity_03
1	question_1	2	question_2	similarity_12
1	question_1	3	question_3	similarity_13
2	question_2	3	question_3	similarity_23

Si se piensa el subconjunto de datos de la Tabla 7 en forma de matriz, en lugar de estructura de tabla, al calcular la distancia de cada una de las filas, se estaría formando una *matriz triangular superior*, en la cual cada uno de los

elementos es la distancia entre un par de preguntas:

$$\begin{bmatrix} 0 & similarity\_01 & similarity\_02 & similarity\_03 \\ 0 & 0 & similarity\_12 & similarity\_13 \\ 0 & 0 & 0 & similarity\_23 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

La estructura de tabla, en lugar de una estructura matricial como la anterior, es utilizada por cuestiones de tecnología. Al utilizarse Python, Apache Spark y el sistema de archivos, es posible generar archivos delimitados por comas (archivos CSV) utilizando el soporte nativo que poseen las tecnologías anteriormente mencionadas. Adicionalmente, esto posibilitó crear particiones de datos de manera sencilla y procesarlas de forma distribuida.

5.4.2. Clustering y etiquetado

Una vez que las similaridades son calculadas en forma distribuida los resultados se almacenan y son colectados para realizar un etiquetado a partir de un análisis de clustering.

El algoritmo elegido fue Clustering de Partición Alrededor de Medoides (PAM) como forma de "etiquetar" cada una de las preguntas como pertenecientes a un determinado cluster, por cada una de las ejecuciones. El motivo de esta elección es porque los elementos no pueden representarse de forma directa en un espacio dimensional, dado que se comparan elementos de texto. Por esta razón, se usa un método que pueda recibir una matriz de similaridad como entrada, y los medoides son elementos reales que se encuentran en el conjunto de datos original. Se implementó de la forma que se describe a continuación.

5.4.2.1. Entrada y configuración inicial

Como entrada fue utilizada la matriz de similaridad resultado del paso anterior, que sirve para obtener las distancias precalculadas entre cada uno de los elementos que participan en el proceso de clustering. Además, en otro arreglo en memoria, se utilizó el conjunto de preguntas individuales con un identificador secuencial, con el fin de facilitar el algoritmo, tal como se muestra en la Tabla 5.

Por cada una de las corridas de clustering (parámetro de configuración) se proporciona un k inicial, que representa al número de clusters (o medoides) que

define el algoritmo, y en los cuales las preguntas son distribuidas (proceso de etiquetado); y además, son identificadas con un identificador único universal (UUID[54] por sus siglas en inglés) con el fin de poder recuperar cada uno de los resultados cuando se realice el ensamble de Clustering de cada una de estas ejecuciones.

5.4.2.2. Proceso de clustering

Como se mencionó anteriormente, el algoritmo PAM tiene dos etapas, denominadas BUILD y SWAP. Por motivos de rendimiento y simplicidad, los experimentos fueron realizados solo utilizando la etapa BUILD para construir cada uno de los clusters. El algoritmo realiza una cantidad de iteraciones máxima (parámetro configurable), y por cada una de ellas se realiza el proceso descrito a continuación.

Generación de etiquetas La generación de etiquetas significa asignar un medoide a una pregunta en particular, con el objetivo del armado de clusters. Como cada medoide es exclusivamente representativo de un cluster, la asignación de un medoide a una pregunta es equivalente a la asignación de la pregunta al cluster correspondiente.

1. Se obtiene su similaridad con cada uno de los medoides. Para esto, se busca cada par de preguntas en una de las matrices de similaridad generadas por el paso anterior.

2. Se asigna la pregunta al medoide (cluster) en la cual su similaridad es máxima.

3. Se obtiene la suma de las similaridades totales por cada medoide, y se almacena en un arreglo que será utilizado en el siguiente paso.

4. Se generan pares (ID pregunta, ID pregunta medoide) que representarán las etiquetas utilizadas por el algoritmo de ensamble.

Actualización de medoides Se busca actualizar los medoides con el fin de evaluar si, luego del cambio, se consiguen mejores resultados. Esto es, por cada uno de los clusters, se obtiene la similaridad de las preguntas todas contra todas, de la siguiente manera:

[54] Identificador único universal `https://es.wikipedia.org/wiki/Identificador_%C3%BAnico_universal`

1. Se toma una pregunta i de la lista, y se compara la similaridad con todas las preguntas restantes del cluster.

2. Se suman todas las similaridades calculadas.

3. Si esta suma es mayor a la suma de las similaridades que obtuvo el medoide actual en el paso anterior, la pregunta i pasa a ser el nuevo medoide.

4. Se recalculan las etiquetas ($id\_pregunta, id\_pregunta\_medoide$)

La *generación de etiquetas* y la *actualización de medoides* se realiza de forma iterativa hasta que (I) los medoides convergen o; (II) se llega al límite máximo de iteraciones. La convergencia de medoides significa que los medoides calculados por la iteración actual, son exactamente los mismos que la iteración anterior, lo que indica que el resultado es óptimo (dentro de los parámetros y las capacidades del algoritmo). Por otro lado, en caso de que no se haya conseguido la convergencia, el conjunto de clusters generado en la última iteración es el que se toma como válido.

5.4.2.3. Estructura de los resultados

Los resultados son almacenados en un archivo CSV que posee la estructura de la Tabla 8. La primera columna es un UUID que identifica una ejecución en particular del algoritmo de clustering, la segunda columna es un identificador de pregunta individual, y la tercera columna es un identificador de pregunta real que actúa como medoide de un cluster.

Tabla 8: Ejemplo de la estructura del resultado de la ejecución del algoritmo de clustering.

run_uuid	question_id	assigned_medoid
63815467136575428551131593057064980770	336	856
63815467136575428551131593057064980770	342	856
63815467136575428551131593057064980770	26	358
63815467136575428551131593057064980770	1364	437

Cada uno de los archivos almacena un UUID único (run_uuid), es cual es idéntico en cada una de las filas, para facilitar el algoritmo de ensamble. Además, cada archivo contiene todas las preguntas individuales de la muestra en cuestión ($question\_id$), y el cluster a la cual pertenecen, el cual es representado por el ID de la pregunta que fue tomada como medoide para ese cluster ($assigned\_medoid$).

Se generan tantos archivos por la cantidad de ejecuciones configuradas para cada una de las técnicas de similaridad del estado del arte. Por ejemplo, si utilizamos Word2Vec, TF, TFIDF, FastText y Semántica (5 técnicas) y se configuraron 100 ejecuciones por cada una de ellas, se obtendrán 500 archivos de etiquetas; los cuales serán la única entrada del algoritmo de ensamble de clustering.

5.5. Ensamble de Clustering

El proceso de ensamble comienza obteniendo todos los archivos de etiquetas generados por el paso anterior. El resultado es una colección Spark, con la misma estructura que los archivos físicos, pero residente en memoria y en todos los nodos del cluster Hadoop, con el fin de poder realizar el procesamiento de una forma eficiente y escalable.

El conjunto de etiquetas es agrupado por ID de pregunta, y se aplica una función de grupo que obtendrá una lista de tuplas o parejas $(run\_uuid, cluster\_id)$. Con el fin de clarificar esta idea, se va a dar un ejemplo: Si se realizan 3 ejecuciones del algoritmo de clustering, cada una de ellas es representada como un archivo en formato CSV y procesada de manera distribuida. Cada uno de esos archivos generados para cada ejecución, contiene información del UUID de ejecución y la asignación de cada una de las preguntas individuales a un cluster resultado. En el caso de PAM, cada identificador de cluster es un identificador real de una pregunta individual (o medoide desde la perspectiva algorítmica). En caso de que se realicen 3 ejecuciones, a modo de ejemplo, los archivos de etiqueta generados pueden representarse como se muestra en las Tablas 9, 10 y 11, con identificadores de ejecución run_uuid_1, run_uuid_2 y run_uuid_3, respectivamente.

Tabla 9: Ejemplo de asignación de clusters a preguntas individuales para la ejecución 1.

run_uuid	**question_id**	**cluster_id**
run_uuid_1	1	1
run_uuid_1	2	1
run_uuid_1	3	1
run_uuid_1	4	4

Considerando run_uuid_1, se puede observar que las preguntas 1, 2 y 3 fueron asignadas al cluster representado por la pregunta 1. En cambio, la pregunta 4 fue asignada a un cluster distinto, representado por ella misma. La intuición para

Tabla 10: Ejemplo de asignación de clusters a preguntas individuales para la ejecución 2.

run_uuid	**question_id**	**cluster_id**
run_uuid_2	1	1
run_uuid_2	2	2
run_uuid_2	3	1
run_uuid_2	4	2

Tabla 11: Ejemplo de asignación de clusters a preguntas individuales para la ejecución 3.

run_uuid	**question_id**	**cluster_id**
run_uuid_3	1	3
run_uuid_3	2	2
run_uuid_3	3	3
run_uuid_3	4	2

esta ejecución en particular, es que las preguntas 1, 2 y 3 son "más similares" entre sí, que la pregunta 4.

En el siguiente paso, el conjunto de datos es agrupado por *question_id*, y se aplica una función de grupo que genera tuplas $(run\_uuid, cluster\_id)$. Para el ejemplo anterior resultaría tal como se muestra en la Tabla 12. La columna *tuples* contiene un arreglo de tuplas que indica, por cada pregunta individual, el conjunto de ocurrencias ejecución-cluster que fueron generadas a lo largo del experimento.

Tabla 12: Conjunto de datos de asignación de clusters agrupados por pregunta individual para generación de tuplas $(run\_uuid, cluster\_id)$.

question_id	**tuples**
1	[(run_uuid_1,1),(run_uuid_2,1),(run_uuid_3,3)]
2	[(run_uuid_1,1),(run_uuid_2,2),(run_uuid_3,2)]
3	[(run_uuid_1,1),(run_uuid_2,1),(run_uuid_3,3)]
4	[(run_uuid_1,4),(run_uuid_2,2),(run_uuid_3,2)]

Hasta ahora, el conjunto de datos representado por la Tabla 12 muestra, por cada una de las preguntas, a qué cluster fue asignada por cada ejecución del algoritmo. Siguiendo con el ejemplo, la pregunta 1 y la pregunta 2 fueron asignadas al cluster con identificador 1 para run_uuid_1, ya que las dos preguntas poseen la tupla $(run\_uuid\_1, 1)$. Este formato de representación facilita el cálculo de la cantidad de veces que dos preguntas fueron asignadas al mismo cluster, para una misma ejecución. Para obtener el resultado de este cálculo, es necesario realizar

una intersección de arreglos para cada una de las combinaciones de preguntas. Para este ejemplo, la intersección de arreglos se muestra en la Tabla 13.

Tabla 13: Conjunto de datos intermedio que indica cuando dos preguntas coinciden en el mismo cluster/ejecución mediante un arreglo de tuplas.

question_id_1	**question_id_2**	**tuples**
1	2	[(run_uuid_1,1)]
1	3	[(run_uuid_1,1),(run_uuid_2,1),(run_uuid_3,3)]
1	4	[]
2	3	[(run_uuid_1,1)]
2	4	[(run_uuid_2,2)]
3	4	[]

La longitud de cada una de las listas indica la cantidad de veces que una pregunta coincide con otra pregunta en el mismo cluster para una misma ejecución. Por ejemplo, las preguntas 1 y 3 resultaron en el mismo cluster para todas las ejecuciones, lo que indicaría una gran similaridad entre ellas. Por el contrario, las preguntas 1 y 4 no presentan ninguna intersección, lo cual es interpretado como una similaridad muy baja entre las preguntas en cuestión.

La cantidad de veces que una pregunta coincide con otra, dividido la cantidad total de ejecuciones, nos indica, en un rango normalizado $[0, 1]$, cuán similares son entre cada una de ellas, de manera adimensional. Aplicando este concepto, calculamos:

```
1 len(set(tuples_1).intersection(set(tuples_2))) / total_runs
```

Este es un cociente donde la expresión `len(set(tuples_1).intersection(set(tuples_2)))` en el numerador representa la cantidad de veces que dos preguntas coincidieron en el mismo cluster para una misma ejecución, y la expresión `total_runs` en el denominador representa la cantidad total de ejecuciones, respondiendo a la fórmula de Ensamble de Clustering de Acumulacion de Evidencias:

$$C(i, j) = \frac{n_{ij}}{N}.$$

donde n_{ij} y N corresponden a las expresiones del numerador y denominador arriba mencionadas, respectivamente.

Considerando el ejemplo en cuestión, el conjunto de datos resultado de aplicar la fórmula anterior se muestra en la Tabla 14, que representa a la matriz de co-asociación de salida del proceso de ensamble de clustering. Esta matriz se

Tabla 14: Ejemplo de matriz de co-asociación salida del proceso de ensamble de clustering.

question_id_1	**question_id_2**	**similarity**
1	2	0.3333
1	3	1.0
1	4	0
2	3	0.3333
2	4	0.3333
3	4	0

representa con tres columnas: las dos primeras identifican a las preguntas y la tercera es la información de similaridad entre ellas.

Particularmente, la matriz de co-asociación mostrada en la Tabla 14, indica que las preguntas 1 y 3 coincidieron en el 100 % de las ejecuciones (valor 1,0, segunda fila), mientras que las preguntas 1 y 2 solo en un tercio de las mismas (valor 0,3333, primera fila), y las preguntas 1 y 4 nunca coincidieron (valor 0, tercera fila). Estos indicadores son tomados entonces como la similaridad resultante del método propuesto.

5.6. Método de validación

5.6.1. Generación de conjuntos estadísticamente significativos

Para generar resultados estadísticamente significativos se ejecutó el proceso completo de modo iterativo, variando dos parámetros principales: (I) el tamaño de la muestra y (II) el número de clusters k. Como experimentos para este trabajo se realizaron ejecuciones con conjuntos de datos aleatorios de 100, 500, 1000, 1500 y 2000 pares de preguntas (200, 1000, 2000, 3000 y 4000 preguntas individuales, respectivamente: en total 5 diferentes tamaños de muestra). Para cada tamaño de muestra, se realizaron 10 muestras aleatorias manteniendo un k fijo. Por ejemplo, para un número de clusters $k = 15$ y para un tamaño de muestra 100, se realizaron 10 ejecuciones con un conjunto de datos de entrada aleatorio para cada ejecución. Dando un total de 5 (distintos tamaños de muestra) $\times$ 10 (cantidad de ejecuciones por tamaño de muestra) $=$ 50 matrices de co-asociación resultado, para cada valor de k dado.

5.6.1.1. Elección de los tamaños de muestra

En cuanto a la elección del tamaño de muestra en el apartado anterior, se tomó en cuenta que los conjuntos de datos sean lo suficientemente grandes como para generar resultados estadísticamente significativos, pero con un tamaño apropiado para la ejecución eficiente de los experimentos de forma local, en favor no aumentar innecesariamente la complejidad para la depuración de los mismos. Cuando se aumenta el tamaño de la muestra en forma lineal, la cantidad de cómputos por cada uno de los algoritmos de similaridad aumentan de forma cuadrática. Como se mencionó anteriormente, siendo n el número de pares de preguntas de una muestra, se realizarán $2n^2 - n$ cálculos. Además, si, por ejemplo, utilizamos 5 algoritmos de similaridad el número de cálculos de similaridad es de $5(2n^2 - n)$, con el agregado de que al algoritmos de clustering PAM y el ensamble de clustering también aumentan su complejidad de una forma considerable, dependiendo de nuestro número de clusters k.

5.6.1.2. Elección del número de clusters

La elección del número de clusters k en los experimentos realizados sigue una lógica que busca una distribución homogénea de los mismos a lo largo de todos los tamaños de muestras, es decir, se pretende no variar el número de clusters entre ellas, para una justa comparación. Como regla general (*rule of thumb* o *regla del pulgar*) se considera como número "óptimo" de clusters un valor de alrededor de $\sqrt{n/2}$ (Kodinariya y Makwana, 2013), siendo n el tamaño de la muestra. Para el número de muestras elegido (200, 1000, 2000, 3000 y 4000 preguntas individuales), los valores siguiendo esta regla serían $k = 10$, $k = 22$, $k = 31$, $k = 38$, $k = 44$.

Los valores generados por la regla del pulgar se encuentran en un rango $[10, 44]$, por lo cual, finalmente se optó por elegir los valores de k con una separación uniforme de los mismos, para facilitar su interpretación y visualización, respetando ese rango de cobertura. Los experimentos para este trabajo se realizaron con los valores $k = 5, 10, 15, 20, 25, 30, 35, 40, 45, 50$.

La validación de los valores de k elegidos para realizar los experimentos fueron validados en conjunto mediante el rendimiento de la matriz de co-asociación. Partiendo de la base del *método del codo* para la evaluación del rendimiento de un algoritmo de clustering, el cual evalúa el porcentaje de variabilidad (suma de

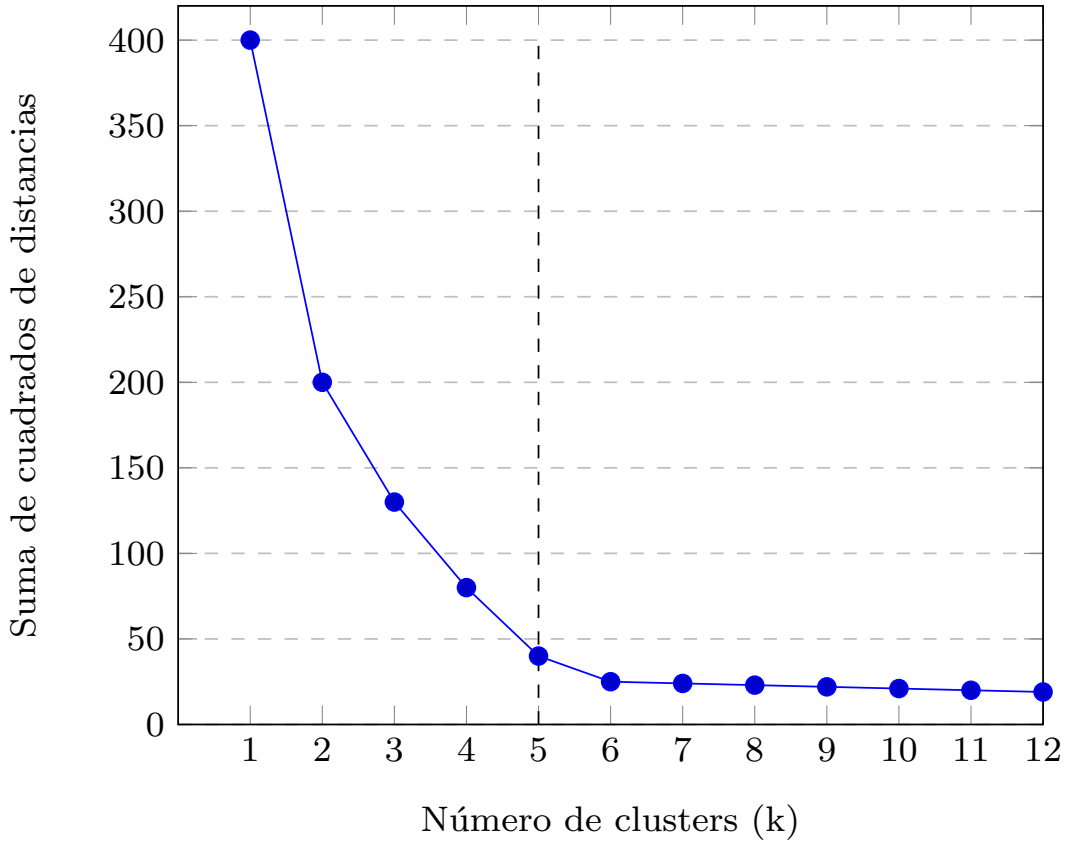

Figura 13: Ejemplo de gráfico para método del codo. Valor óptimo $k = 5$.

cuadrados de distancias) explicada en función del número de clusters, mediante la idea de encontrar el número mínimo de clusters por el cual agregando un cluster adicional no modelaría mejor los datos. El porcentaje de variabilidad explicado por los clusters es graficado contra el número de clusters. Los primeros clusters agregarán una información considerable al modelo, pero en determinado punto la ganancia marginal caerá dramáticamente, dando un ángulo en el gráfico (Bholowalia y Kumar, 2014). La Figura 13 muestra ejemplo en el cual en un un gráfico bidimensional se compara el número de clusters (en abscisas) y la suma de cuadrados de distancias (en ordenadas) como medida de variabilidad de los clusters. En este gráfico de ejemplo se puede ver como luego de $k = 5$, la suma de cuadrados de distancias varía muy poco, ya que los valores esbozan una curva con valores aproximadamente constantes a partir de este punto.

Con el fin de dar una aplicación más integral al método del codo, en lugar de calcularlo por cada una de las técnicas de clustering aplicadas, el mismo se aplicó utilizando la matriz de co-asociación generada a partir del ensamble. Para cuantificar y evaluar el rendimiento, se utilizaron matrices de confusión que indican el error entre los resultados obtenidos y la clasificación proveniente

del conjunto de datos original, por cada uno de los valores de k anteriormente mencionados, como veremos a continuación.

5.6.2. Estructura de las matrices de confusión

Una matriz de confusión es una matriz que muestra clasificaciones predichas y su comparación con las clases reales. Una matriz de confusión puede ser de tamaño $L \times L$ donde L es el número de diferentes valores de etiqueta o clase (Provost y Kohavi, 1998). En este trabajo, las clases son: 0 (las preguntas comparadas no son iguales) y 1 (las preguntas son iguales). Por lo cual, la matriz de confusión que se deriva, se muestra en la Tabla 15. Los valores "a" y "d" representan el porcentaje de veces que los valores predichos fueron iguales a los reales, es decir, los casos en que las preguntas fueron predichas, respectivamente, iguales o distintas y realmente lo eran. Por otro lado, "c" es la proporción de preguntas predichas distintas, que en realidad son iguales; y "b" la proporción de preguntas predichas iguales, pero son realmente distintas. El objetivo de un algoritmo con buen desempeño entonces, es maximizar "a" y "d" y, por lo tanto, minimizar "b" y "c".

Tabla 15: Matriz de confusión para validación de resultados.

		Predicho	
		0	**1**
Real	**0**	a	b
	1	c	d

Los indicadores de desempeño seleccionados para ser evaluados en los experimentos realizados con fines comparativos, son los siguientes:

- **Exactitud:** $(a + d)/(a + b + c + d)$.
- **Error:** $(b + c)/(a + b + c + d)$.

En estos indicadores se cumple la condición $a + b + c + d = 1$.

5.6.2.1. Preparación de los datos

Para evaluar el rendimiento del algoritmo de ensamble, se toma la matriz de co-asociación generada como resultado del proceso total y la muestra de pares de preguntas que se utilizó como entrada para ese proceso. Por ejemplo, se considera

la muestra de preguntas de la Tabla 16, la cual muestra dos pares de preguntas (123004 y 98776 - 4 preguntas en total) con su identificador de par y un indicador en la columna *equal*, que tiene dos valores posibles: 1 (preguntas iguales) y 0 (preguntas distintas). Por otro lado, en la Tabla 17, se muestra la matriz de co-asociación generada, por el método EQuAL, a partir de la Tabla 16, teniendo en cuenta todas las combinaciones tomadas de a 2 de las 4 preguntas originales, para las cuales se agrega la similaridad entre pares, en la columna similarity. Por último, se filtran solo los pares de preguntas de la matriz de co-asociación que se encuentran en la Tabla 16, ya que son las únicas con la cual su similaridad puede compararse con fines de validación, es decir, el conjunto de datos mostrado en la Tabla 18. Lo que nos deja con un conjunto de pares de preguntas que es posible comparar en su totalidad con la muestra original.

Tabla 16: Muestras de pares de preguntas que se utilizó como entrada del método EQuAL.

sequence_id	**question_pair_id**	**question_1**	**question_2**	**equal**
0	123004	question_10	question_20	1
1	98776	question_11	question_21	0

Tabla 17: Matriz de co-asociación generada a partir de la muestra de la Tabla 16.

question_id_1	**question_id_2**	**question_1**	**question_2**	**similarity**
question_10	question_11	contenido	contenido	0.857
question_10	question_20	contenido	contenido	0.210
question_10	question_21	contenido	contenido	0.126
question_11	question_20	contenido	contenido	0.006
question_11	question_21	contenido	contenido	0.368
question_20	question_21	contenido	contenido	0.146

Tabla 18: Filtrado de la Tabla 17 con los pares de preguntas que se encuentran en la Tabla 16.

question_id_1	**question_id_2**	**question_1**	**question_2**	**similarity**
question_10	question_11	contenido	contenido	0.857
question_11	question_21	contenido	contenido	0.368

Ya se realizaron los cálculos de similaridad, los algoritmos de clustering y la matriz de co-asociación. En la siguiente sección, se procederá a interpretar los resultados obtenidos.

5.6.2.2. Construcción y elección del umbral correcto

La problemática que se intenta resolver teniendo en cuenta todas las similaridades obtenidas a partir del ensamble de clustering es ¿Cuándo consideramos a esas preguntas iguales y cuándo no? El criterio tomado es que cuando la similaridad S entre un par de preguntas (q_1, q_2) es igual o superior a cierto umbral t se considera que son iguales (valor 1) y distintas si sucede lo contrario (valor 0). De esta forma:

$$f(x) = \begin{cases} 1 & si \quad S(q_1, q_2) \geq t \\ 0 & si \quad S(q_1, q_2) < t \end{cases} .$$

La elección del mejor umbral, se realiza eligiendo valores en el intervalo $(0, 1)$ y evaluando cual de ellos conlleva a un mejor rendimiento, es decir, que los valores calculados a partir del umbral coincidan, en una mayor medida, con el valor real proveniente de la muestra de datos. Por ejemplo, tomando valores potenciales de umbral con intervalos 0,05 se forma un arreglo como $[0{,}05, 0{,}1, 0{,}15, ..., 0{,}90, 0{,}95]$ y se itera sobre cada uno de ellos. Por cada uno de los valores en el arreglo:

1. Se consideran en cada iteración todos los pares de preguntas tomados para realizar la comparación, provenientes de la matriz de co-asociación.

2. Por cada uno de los valores de similaridad, se asigna 1 si son mayores o iguales al umbral, 0 si pasa lo contrario.

3. Si los valores asignados en el paso anterior coinciden con el valor real, se asigna un valor *true* (verdadero), si no coinciden, se asigna *false* (falso).

4. Se calcula la proporción de pares de preguntas asignadas con *true*, es decir, que el valor real coincide con el predicho, y se obtiene la *exactitud* del método.

El valor de umbral que arroje la mayor exactitud, será el utilizado para evaluar el desempeño del método al construir la matriz de confusión.

Volviendo al ejemplo anterior, un umbral elegido de 0,65 aplicado a la Tabla 18, arrojaría el resultado de la Tabla 19 (el único par de preguntas que presenta una similaridad mayor al umbral es $(question\_10, question\_11)$, por lo cual se le asigna el valor 1). La Tabla 19, muestra un resultado idéntico al conjunto de entrada, es decir, a la Tabla 16. Lo anterior significa que los pares predichos son iguales a los pares reales. En otro caso, si el umbral que tiene mejor rendimiento fuese 0,9, el resultado obtenido luego del cómputo hubiese sido el de la Tabla 20,

el cual expone una diferencia entre el valor real y el predicho del par de preguntas ($question\_10, question\_11$). Lo anterior denota el mayor rendimiento del umbral 0,65 contra el umbral 0,9 y su importancia en la elección del valor correcto.

Tabla 19: Asignación binaria de los resultados de similaridad obtenidos en la Tabla 18, teniendo en cuenta un umbral de 0,65.

question_id_1	**question_id_2**	**question_1**	**question_2**	**equal**
question_10	question_11	contenido	contenido	1
question_11	question_21	contenido	contenido	0

Tabla 20: Asignación binaria de los resultados de similaridad obtenidos en la Tabla 18, teniendo en cuenta un umbral de 0,9.

question_id_1	**question_id_2**	**question_1**	**question_2**	**equal**
question_10	question_11	contenido	contenido	0
question_11	question_21	contenido	contenido	0

En conclusión, el rendimiento del algoritmo se medirá comparando la variable de clase (1 o 0) del conjunto de datos de entrada, con la variable de clase construida desde la comparación de las similaridades resultados del método y un umbral apropiado. Cuantos más pares de preguntas coincidan, mejor será el rendimiento del algoritmo, el cual se podrá visualizar mediante matrices de confusión.

5.6.2.3. Construcción de las matrices de confusión

Con el fin de poder ilustrar cómo se construyen las matrices de confusión a partir de la comparación de los conjuntos de datos, se utilizará el siguiente ejemplo. Supongamos que el conjunto de datos de entrada es el que se muestra en la Tabla 21, Y el resultado obtenido luego de la elección del mejor umbral, es la Tabla 22. Por lo cual, comparando los valores de la columna "equal", obtendremos el resultado de la Tabla 23. Resumiendo los resultados, la matriz de confusión final se muestra en la Tabla 24. La exactitud obtenida a partir de este conjunto de datos y la ejecución hipotética del método es 0,75 (y por lo tanto, el error es 0,25).

Tabla 21: Ejemplo de conjunto de datos de entrada (reales) para validación.

sequence_id	**question_pair_id**	**question_1**	**question_2**	**equal**
0	123004	question_10	question_20	1
1	98776	question_11	question_21	1
2	14422	question_12	question_22	1
3	12321	question_13	question_23	1
4	999	question_14	question_24	0
5	7448	question_15	question_25	0
6	69553	question_16	question_26	0
7	2447	question_17	question_27	1

Tabla 22: Ejemplo de conjunto de datos de predichos por el método EQuAL.

question_id_1	**question_id_2**	**question_1**	**question_2**	**equal**
question_id_10	question_id_20	question_10	question_20	1
question_id_11	question_id_21	question_11	question_21	1
question_id_12	question_id_22	question_12	question_22	0
question_id_13	question_id_23	question_13	question_23	1
question_id_14	question_id_24	question_14	question_24	1
question_id_15	question_id_25	question_15	question_25	0
question_id_16	question_id_26	question_16	question_26	0
question_id_17	question_id_27	question_17	question_27	1

Tabla 23: Resultado de comparación de las tablas 21 y 22 para validación y construcción de matrices de confusión.

sequence_id	**real**	**predicho**	**resultado**
0	1	1	true
1	1	1	true
2	1	0	false
3	1	1	true
4	0	1	false
5	0	0	true
6	0	0	true
7	1	1	true

Tabla 24: Matriz de confusión obtenida a partir de la comparación de las tablas 21 y 22.

		Predicho	
		0	**1**
Real	**0**	0.25	0.125
	1	0.125	0.5

Capítulo 6. **Resultados**

En esta sección, se presentan los resultados obtenidos a partir de los experimentos realizados. Los mismos se presentan en distintas tablas que se pueden organizar en dos grandes grupos: i) Análisis del método propuesto y; ii) Análisis del método propuesto y algoritmos del estado del arte. Las matrices de confusión calculadas para generar las matrices de confusión promedio utilizadas en este apartado, se encuentran en la Tabla 40, ubicada en el anexo de este trabajo.

6.1. Análisis del método propuesto

A continuación se muestran distintas ejecuciones del método EQuAL, con el fin de generar resultados estadísticamente significativos, como se explicó en el apartado 5.6. Se presentan tablas correspondientes a un tamaño de muestra en particular: 100, 500, 1000, 1500 y 2000 pares de preguntas; y en cada una de ellas, las filas corresponden a un número de clusters k distinto: 5, 10, 15, 20, 25, 30, 35, 40, 45 y 50. Además, en todas las ejecuciones se realizaron 100 repeticiones del método de clustering, es decir, teniendo en cuenta que se ensamblaron 5 algoritmos del estado del arte, para cada una de las ejecuciones de un valor k en particular se realizaron un total de 500 ejecuciones del método de clustering. Como salida de evaluación del rendimiento del método EQuAL, se utilizan matrices de confusión. En las Tablas de 25 a 29, se muestran, por cada fila, una matriz de confusión con los valores promedio de cada una de las 10 ejecuciones para ese tamaño de muestra y número de clusters k. Como indicador de rendimiento final, se suman las columnas "% 0/0" (porcentaje de preguntas distintas predichas correctamente) y "% 1/1" (porcentaje de preguntas iguales predichas incorrectamente) para obtener la exactitud, y el error es derivado de $1 - exactitud$ o sumando "% 0/1" (porcentaje de preguntas distintas predichas incorrectamente) y "% 1/0" (porcentaje de preguntas iguales predichas incorrec-

tamente). Para finalizar, se resalta, en verde, el resultado con error más pequeño correspondiente a un valor k para cada tamaño de muestra.

Tabla 25: Matrices de confusión promedio del método EQuAL. 100 muestras de 100 pares de preguntas cada una.

Número de Clusters (k)	**% 0/0**	**% 0/1**	**% 1/0**	**% 1/1**	**Exactitud**	**Error**
5	0.475	0.119	0.203	0.203	0.678	0.322
10	0.491	0.103	0.215	0.191	0.682	0.318
15	0.444	0.15	0.164	0.242	0.686	0.314
20	0.449	0.145	0.173	0.233	0.682	0.318
25	0.435	0.159	0.15	0.256	0.691	0.309
30	0.435	0.159	0.145	0.261	0.696	0.304
35	0.444	0.15	0.157	0.249	0.693	0.307
40	0.408	0.186	0.123	0.283	0.691	0.309
45	0.459	0.135	0.176	0.23	0.689	0.311
50	0.463	0.131	0.177	0.229	0.692	0.308

Tabla 26: Matrices de confusión promedio del método EQuAL. 100 muestras de 500 pares de preguntas cada una.

Número de Clusters (k)	**% 0/0**	**% 0/1**	**% 1/0**	**% 1/1**	**Exactitud**	**Error**
5	0.4562	0.1458	0.1908	0.2072	0.6634	0.3366
10	0.4468	0.1596	0.1658	0.2278	0.6746	0.3254
15	0.4356	0.1708	0.1542	0.2394	0.675	0.325
20	0.4316	0.1748	0.1444	0.2492	0.6808	0.3192
25	0.4306	0.1758	0.1468	0.2468	0.6774	0.3226
30	0.4322	0.1742	0.1476	0.246	0.6782	0.3218
35	0.4334	0.173	0.1458	0.2478	0.6812	0.3188
40	0.4272	0.1792	0.1378	0.2558	0.683	0.317
45	0.439	0.1674	0.1488	0.2448	0.6838	0.3162
50	0.4378	0.1686	0.1454	0.2482	0.686	0.314

En forma de resumen comparativo, se van a analizar los errores cruzando cada uno de los tamaños de muestra con el número de clusters k utilizado, tal como se muestra en la Tabla 30. La media mínima de error fue obtenida para el tamaño de muestra 100, con 0,312 y para el valor $k = 50$, con 0,32286. El valor promedio de error más bajo se ubicó en el tamaño de muestra 100 para $k = 30$, el cual fue de 0,304. Además, con el fin de tener documentada la variabilidad del método, se calculó la varianza promedio por tamaño de muestra que resultó tener un valor mínimo de 0,000124864 y la varianza promedio por valor k con

Tabla 27: Matrices de confusión promedio del método EQuAL. 100 muestras de 1000 pares de preguntas cada una.

Número de Clusters (k)	**% 0/0**	**% 0/1**	**% 1/0**	**% 1/1**	**Exactitud**	**Error**
5	0.4488	0.1561	0.1902	0.2049	0.6537	0.3463
10	0.4482	0.1559	0.1847	0.2112	0.6594	0.3406
15	0.4502	0.1539	0.188	0.2079	0.6581	0.3419
20	0.4655	0.1386	0.2007	0.1952	0.6607	0.3393
25	0.462	0.1421	0.1957	0.2002	0.6622	0.3378
30	0.461	0.1431	0.1933	0.2026	0.6636	0.3364
35	0.4608	0.1433	0.1933	0.2026	0.6634	0.3366
40	0.466	0.1381	0.2016	0.1943	0.6603	0.3397
45	0.4445	0.1596	0.1765	0.2194	0.6639	0.3361
50	0.4521	0.152	0.1804	0.2155	0.6676	0.3324

Tabla 28: Matrices de confusión promedio del método EQuAL. 100 muestras de 1500 pares de preguntas cada una.

Número de Clusters (k)	**% 0/0**	**% 0/1**	**% 1/0**	**% 1/1**	**Exactitud**	**Error**
5	0.4297	0.1773	0.1713	0.2217	0.6514	0.3486
10	0.4709	0.136	0.2035	0.1896	0.6605	0.3395
15	0.4293	0.1777	0.1633	0.2297	0.659	0.341
20	0.4269	0.18	0.1624	0.2307	0.6576	0.3424
25	0.4383	0.1687	0.171	0.222	0.6603	0.3397
30	0.4509	0.156	0.1824	0.2107	0.6616	0.3384
35	0.4599	0.1471	0.1893	0.2037	0.6636	0.3364
40	0.4439	0.1631	0.1753	0.2177	0.6616	0.3384
45	0.4485	0.1585	0.1765	0.2165	0.665	0.335
50	0.451	0.156	0.1733	0.2197	0.6707	0.3293

un valor mínimo de 0,000416228. Las Figuras 14 y 15 muestran el análisis de los resultados de la Tabla 30 desde dos perspectivas distintas: los errores para los valores de k a lo largo de los distintos tamaños de muestra, y los errores para los tamaños de muestra a lo largo de los distintos valores de k, con el fin de poder observar cómo se comporta el método EQuAL en cada perspectiva, teniendo en cuenta variabilidad y valor absoluto de los errores promedio.

En la Figura 14 se puede visualizar que los valores de error no tienen una marcada proporción con el tamaño de muestra, por el contrario, se mantienen relativamente estables a lo largo de ella, pero arrojando valores más bajos en tamaños de muestra chicos (líneas en colores rojo y azul), lo que indicaría que la variabilidad agregada por el método de ensamble provoca buenos resultados. Si

Tabla 29: Matrices de confusión promedio del método EQuAL. 100 muestras de 2000 pares de preguntas cada una.

Número de Clusters (k)	% 0/0	% 0/1	% 1/0	% 1/1	Exactitud	Error
5	0.4409	0.1668	0.1739	0.2184	0.6593	0.3407
10	0.4496	0.1581	0.1808	0.2115	0.6611	0.3389
15	0.446	0.1617	0.1737	0.2186	0.6646	0.3354
20	0.4476	0.1601	0.1774	0.2149	0.6625	0.3375
25	0.4637	0.144	0.1914	0.2009	0.6646	0.3354
30	0.431	0.1767	0.1586	0.2337	0.6647	0.3353
35	0.4451	0.1626	0.1689	0.2234	0.6685	0.3315
40	0.4569	0.1508	0.1788	0.2135	0.6704	0.3296
45	0.428	0.1797	0.152	0.2403	0.6683	0.3317
50	0.449	0.1587	0.1719	0.2204	0.6694	0.3306

Tabla 30: Error en tamaños de muestra vs. número de clusters k, con media y varianza.

k / Tam. muestra	100	500	1000	1500	2000	Media	Varianza
5	0.322	0.3366	0.3463	0.3486	0.3407	0.33884	0.0004429
10	0.318	0.3254	0.3406	0.3395	0.3389	0.33248	0.0004162
15	0.314	0.325	0.3419	0.341	0.3354	0.33146	0.0005621
20	0.318	0.3192	0.3393	0.3424	0.3375	0.33128	0.0005489
25	0.309	0.3226	0.3378	0.3397	0.3354	0.3289	0.0006738
30	0.304	0.3218	0.3364	0.3384	0.3353	0.32718	0.0008430
35	0.307	0.3188	0.3366	0.3364	0.3315	0.32606	0.0006635
40	0.309	0.317	0.3397	0.3384	0.3296	0.32674	0.0007216
45	0.311	0.3162	0.3361	0.335	0.3317	0.326	0.000536
50	0.308	0.314	0.3324	0.3293	0.3306	0.32286	0.0004917
Media	0.312	0.32166	0.33871	0.33887	0.33466		
Varianza	0.0003	0.0003736	0.0001299	0.0002258	0.0001248		

tomamos las medias por tamaño de muestra, el error promedio mínimo es 0,312 para el tamaño de muestra 100 y el máximo es 0,33887 para el tamaño de muestra 1500, lo cual es una diferencia de solo 0,02687 en unidades normalizadas (es decir, en el rango $[0, 1]$). Por otro lado, es posible ver fácilmente que los mejores resultados (errores más bajos) fueron arrojados cuando el número de cluster aumenta (esta correspondencia será confirmada en el análisis de la Figura 15). Por ejemplo, la curva de $k = 50$ posee el menor error en casi todos los tamaños de muestra.

En la Figura 15 se puede observar una clara tendencia decreciente a medida que se aumenta el número de clusters, para todos los tamaños de muestra. Esto también queda en evidencia cuando se calculan las medias en cada valor de k, alcanzando una media mínima de error de 0,32286 ubicada cuando $k = 50$ y una media máxima de 0,33884 cuando $k = 5$.

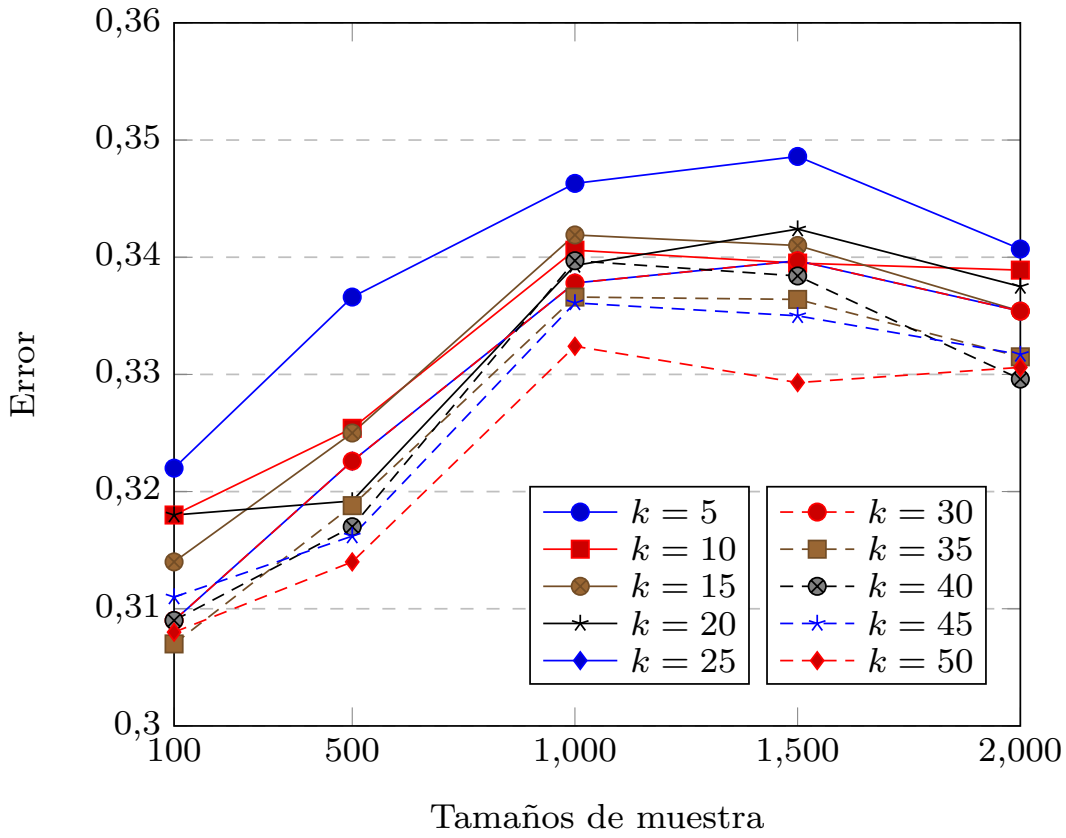

Figura 14: Errores de los valores de k en los distintos tamaños de muestra.

Ambos gráficos sugieren que el método se comporta mejor con valores altos de k, es decir, con más clusters en los cuales las preguntas pueden ser ubicadas.

6.2. Análisis del método propuesto y algoritmos del estado del arte

Con el fin de realizar una comparación de los resultados del método EQuAL con los algoritmos del estado del arte, se ejecutaron experimentos con la misma metodología: 10 diferentes ejecuciones con una muestra aleatoria para cada una de ellas. En las Tablas 31 a 35, se muestran los resultados de estos algoritmos y el mejor resultado del método EQuAL para el correspondiente tamaño de muestra, en forma de matrices de confusión para cada una de las técnicas. Para ellos, se generan 10 muestras aleatorias, y ellas servirán como conjunto de datos origen para cada uno de los métodos (Term Frequency, Term Frequency/Inverse Document Frequency, Word2Vec, FastText y Semantic Distance). Esto posibilita que el cálculo de similaridad y las medidas de desempeño de cada uno de los métodos se generen con exactamente el mismo conjunto de datos, y los resultados

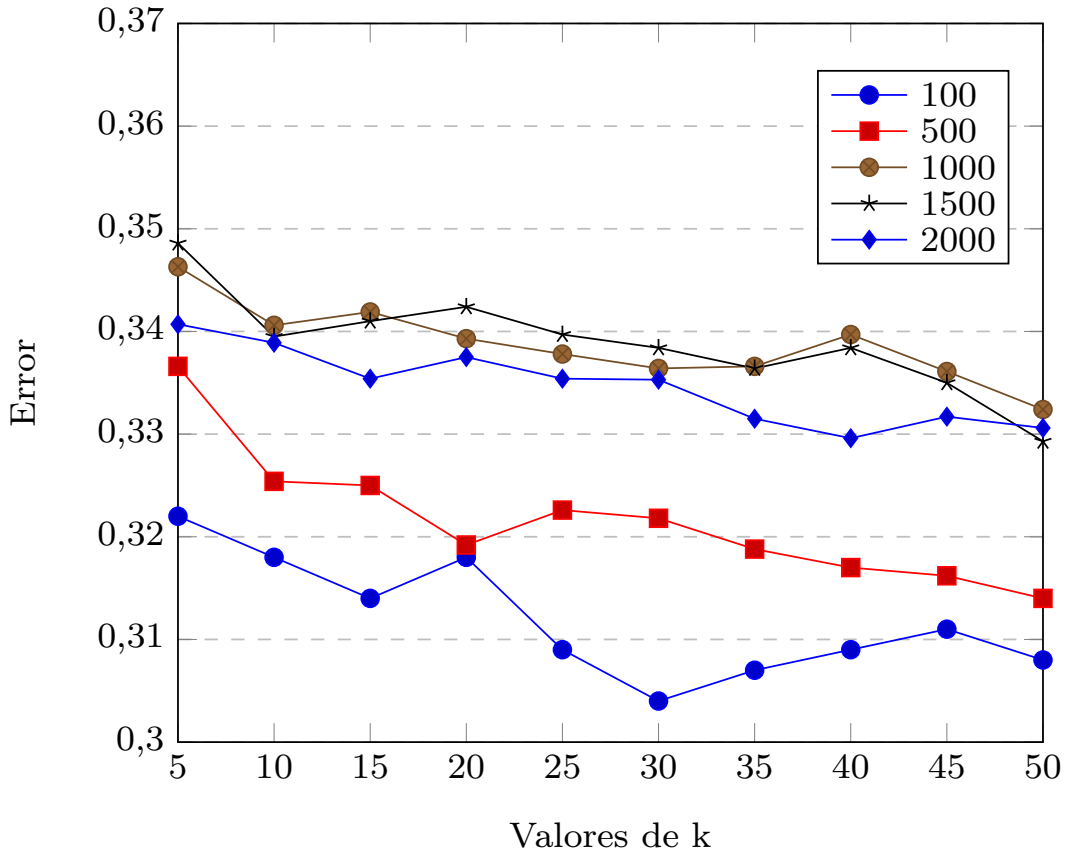

Figura 15: Errores de los valores de k en los distintos tamaños de muestra.

solamente dependan del cálculo de similaridad (en el caso del método EQuAL también dependen del algoritmo de clustering). Se resalta con color verde el mejor resultado (error más pequeño) por cada una de ellas.

Tabla 31: EQuAL vs. técnicas del estado del arte. Tamaño de muestra 100 pares de preguntas y 10 ejecuciones en cada una de las técnicas.

	k	% 0/0	% 0/1	% 1/0	% 1/1	Exactitud	Error
bow	—	0.429	0.165	0.131	0.275	0.704	0.296
ft	—	0.413	0.181	0.153	0.253	0.666	0.334
w2v	—	0.396	0.198	0.119	0.287	0.683	0.317
gtfidf	—	0.48	0.114	0.197	0.209	0.689	0.311
sem	—	0.473	0.121	0.18	0.226	0.699	0.301
EQuAL	30	0.463	0.131	0.177	0.229	0.696	0.304

En forma de resumen, en la Tabla 36, se pueden ver todos los errores por cada uno de los métodos, más la media y la varianza de cada uno de ellos. Esta tabla muestra una media de error a través de las distintas muestras aleatorias para el método EQuAL y las ejecuciones de los algoritmos del estado del arte. El método EQuAL, en este caso, muestra muy buenos resultados para experimentos

Tabla 32: EQuAL vs. técnicas del estado del arte. Tamaño de muestra 500 pares de preguntas y 10 ejecuciones en cada una de las técnicas.

	k	% 0/0	% 0/1	% 1/0	% 1/1	Exactitud	Error
bow	—	0.3948	0.2072	0.105	0.293	0.6878	0.3122
ft	—	0.4724	0.1296	0.1986	0.1994	0.6718	0.3282
w2v	—	0.3758	0.2262	0.0984	0.2996	0.6754	0.3246
gtfidf	—	0.4346	0.1674	0.1544	0.2436	0.6782	0.3218
sem	—	0.4648	0.1372	0.1804	0.2176	0.6824	0.3176
EQuAL	50	0.4378	0.1686	0.1454	0.2482	0.686	0.314

Tabla 33: EQuAL vs. técnicas del estado del arte. Tamaño de muestra 1000 pares de preguntas y 10 ejecuciones en cada una de las técnicas.

	k	% 0/0	% 0/1	% 1/0	% 1/1	Exactitud	Error
bow	—	0.386	0.2189	0.096	0.2991	0.6851	0.3149
ft	—	0.4322	0.1727	0.1548	0.2403	0.6725	0.3275
w2v	—	0.4134	0.1915	0.1254	0.2697	0.6831	0.3169
gtfidf	—	0.4161	0.1888	0.1364	0.2587	0.6748	0.3252
sem	—	0.4639	0.141	0.1752	0.2199	0.6838	0.3162
EQuAL	50	0.4521	0.152	0.1804	0.2155	0.6676	0.3324

Tabla 34: EQuAL vs. técnicas del estado del arte. Tamaño de muestra 1500 pares de preguntas y 10 ejecuciones en cada una de las técnicas.

	k	% 0/0	% 0/1	% 1/0	% 1/1	Exactitud	Error
bow	—	0.4070667	0.198	0.1222667	0.2726667	0.6797333	0.3202667
ft	—	0.4736666	0.1314	0.2006667	0.1942667	0.6679333	0.3320667
w2v	—	0.4266	0.1784667	0.1446667	0.2502667	0.6768667	0.3231337
gtfidf	—	0.4217333	0.1833333	0.1488667	0.2460667	0.6678	0.3322
sem	—	0.463	0.1420667	0.1917333	0.2032	0.6662	0.3338
EQuAL	50	0.451	0.156	0.1733	0.2197	0.6707	0.3293

Tabla 35: EQuAL vs. técnicas del estado del arte. Tamaño de muestra 2000 pares de preguntas y 10 ejecuciones en cada una de las técnicas.

	k	% 0/0	% 0/1	% 1/0	% 1/1	Exactitud	Error
bow	—	0.402	0.20455	0.1102	0.28325	0.68525	0.31475
ft	—	0.46865	0.1379	0.19655	0.1969	0.66555	0.33445
w2v	—	0.40985	0.1967	0.12665	0.2668	0.67665	0.32335
gtfidf	—	0.43255	0.174	0.15625	0.2372	0.66975	0.33025
sem	—	0.4887	0.11785	0.20705	0.1864	0.6751	0.3249
EQuAL	50	0.449	0.1587	0.1719	0.2204	0.6694	0.3306

realizados con tamaños de muestra más pequeños, superando a todos los métodos con excepción de bow y Semantic Distance para el tamaño de muestra 100, y solo siendo superado por bow en los experimentos con tamaño de muestra de 500 pares de preguntas. Por otro lado, se mantiene en un buen rango en los tamaños de muestra más grandes.

Tabla 36: Error en los algoritmos del estado del arte vs. método EQuAL por tamaño de muestra, media y varianza.

	100	**500**	**1000**	**1500**	**2000**	**Media**	**Varianza**
bow	0.296	0.3122	0.3149	0.3202667	0.31475	0.3116233	0.0003396
ft	0.334	0.3282	0.3275	0.3320667	0.33445	0.3312433	0.0000418
w2v	0.317	0.3246	0.3169	0.3231333	0.32335	0.3209967	0.0000558
gtfidf	0.311	0.3218	0.3252	0.3322	0.33025	0.32409	0.0002815
sem	0.301	0.3176	0.3162	0.3338	0.3249	0.3187	0.0005872
EQuAL	0.308	0.314	0.3324	0.3293	0.3306	0.32286	0.0004917

El método propuesto en el presente trabajo posee un indicador de medias de error total a lo largo de todas las muestras de 0,32286. Adicionalmente, analizando las varianzas de cada uno de los métodos, se observa que el método EQuAL se encuentra en el rango esperado (media 0,000491712) y es muy similar a la varianza de los demás métodos. Estos indicadores se pueden visualizar en la Figura 16: los valores de error promedio (en ordenadas) a lo largo de los tamaños de muestra estudiados en este trabajo (en abscisas). En esta Figura el método EQuAL se destaca en línea azul con trazo grueso.

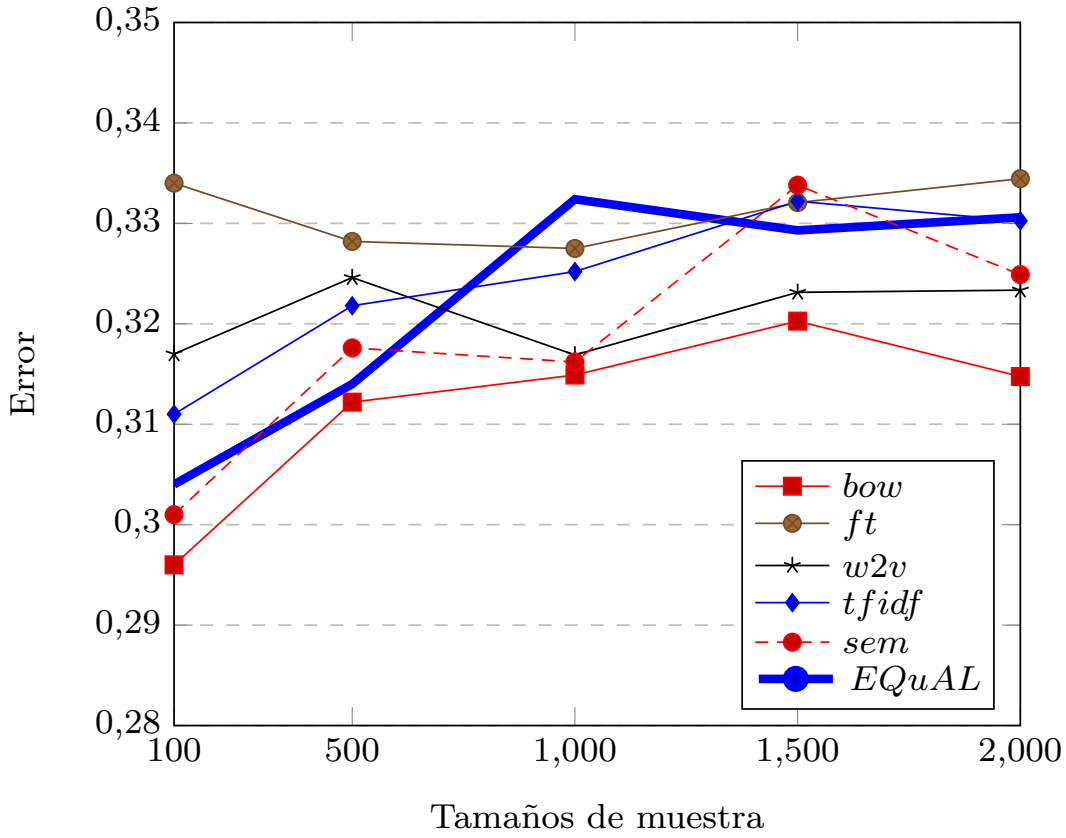

Figura 16: Errores de los tamaños de muestra para el método EQuAL y los algoritmos del estado del arte.

6.3. Otras observaciones de interés

6.3.1. Análisis de varianza del método propuesto

Con el fin de analizar si el método EQuAL es apto para su aplicación en RS y poner en perspectiva sus resultados en comparación con los algoritmos del estado del arte, se realiza un *análisis de varianza* con la aplicación del *método de Tukey*.

El análisis de varianza prueba la significancia estadística de la diferencia de medias (o tendencia central) entre los diferentes métodos en análisis (Tabachnick y Fidell, 2007). En el caso de estudio, considerando la esperanza del error como variable en análisis, deseamos probar que el método EQuAL es apto, eficiente y eficaz para ser utilizado en un sistema de recomendación, en comparación con métodos ampliamente utilizados y reconocidos por la comunidad. En este apartado se usa el análisis de varianza como una herramienta para tal fin. Si los resultados son los esperados, podemos llegar a la conclusión que el método EQuAL puede ser utilizado como medio para generar matrices de similaridad aptas para un RS eficiente y eficaz, mediante pruebas estadísticamente significativas. Se realizó un análisis de varianza[55] con los datos[56] utilizados para generar las tablas resumen del apartado "6.2 Análisis del método propuesto y algoritmos del estado del arte".

Los algoritmos del estado del arte y el método EQuAL basado en el ensamble de los mismos generaron conjuntos de datos que son utilizados como entrada para el análisis. Se denomina μ_0 a la esperanza de los errores del método EQuAL y se denominan $\mu_i, i = 1, ..., 5$ a las esperanzas de los errores de los métodos, TF, TF-IDF, FastText, Word2Vec y Semantic Distance, respectivamente. Se plantean las siguientes hipótesis:

- H_0: $\mu_0 - \mu_i = 0, i = 1, ..., 5.$
- H_1: $\mu_0 - \mu_i \neq 0, i = 1, ..., 5.$

Por lo mencionado, se plantean estas pruebas de hipótesis para cada variable en forma separada.

[55] El código completo del análisis de varianza se encuentra en este repositorio GitHub `https://github.com/fedetesone/anova_equal`.

[56] Conjunto de datos fuente para el análisis de varianzas: `https://github.com/fedetesone/anova_equal/blob/main/resultados.csv`.

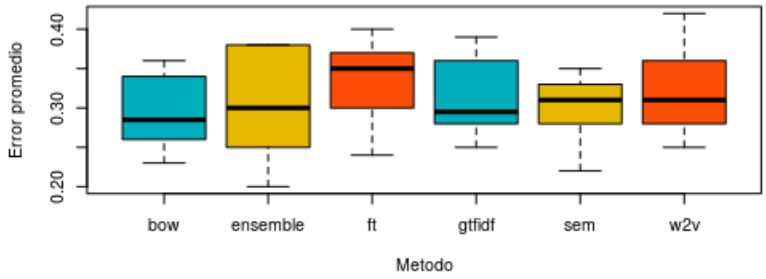

Figura 17: Gráfico de caja y bigote para las medias de error de los métodos en estudio para tamaño de muestra de 100 pares de preguntas.

El análisis de medias de error a continuación se realizará comparando el método EQuAL contra los métodos del estado del arte (uno a uno), para un tamaño de muestra determinado. Para tal fin, se utiliza el método de Tukey para crear intervalos de confianza para todas las diferencias de medias, en parejas. Por lo anterior, es necesario el énfasis especial es el método EQuAL realizando comparaciones específicas mediante pares, contra cada una de las técnicas del estado del arte (Abdi y Williams, 2010). Filtrando las comparaciones uno a uno en las cuales se encuentra el método EQuAL, es posible saber cómo se comporta. Si las medias de error no tienen diferencias significativas en términos estadísticos, esto significa que no existe una diferencia estadísticamente significativa entre el método EQuAL y los restantes métodos del estado del arte. Con lo cual se concluye que, al menos, el método presentado es apto para la generación de medidas de similaridad a partir del conjunto de datos en estudio, y su consecuente aplicación en RS.

A continuación, se muestran los métodos Tukey aplicados en R para cada uno de los tamaños de muestra tenidos en cuenta en los experimentos, utilizando un nivel de confianza de 95 %($\alpha = 0{,}05$).

6.3.1.1. Tamaño de muestra de 100 pares de preguntas

	diff	lwr	upr	p adj
ensemble−bow	0.008	−0.0589175	0.0749175	0.9992360
ft−ensemble	0.030	−0.0369175	0.0969175	0.7702455
gtfidf−ensemble	0.007	−0.0599175	0.0739175	0.9996012
sem−ensemble	−0.003	−0.0699175	0.0639175	0.9999940
w2v−ensemble	0.013	−0.0539175	0.0799175	0.9923331

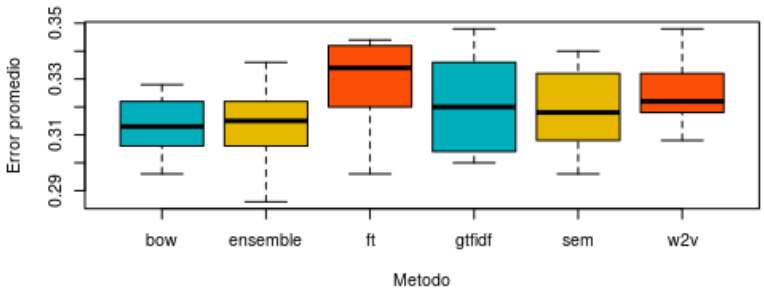

Figura 18: Gráfico de caja y bigote para las medias de error de los métodos en estudio para tamaño de muestra de 500 pares de preguntas.

El método EQuAL posee una media de error que no posee diferencias significativas a todos los métodos. En todos los casos, los intervalos de confianza incluyen al valor 0 y $p-adj > \alpha$. La Figura 17 muestra un gráfico de caja y bigote para las medias de error en estudio y tamaño de muestra 100. En la misma, se puede visualizar que el método EQuAL posee el menor valor de error para una ejecución individual en particular, representado por el bigote inferior.

6.3.1.2. Tamaño de muestra de 500 pares de preguntas

	diff	lwr	upr	p adj
ensemble–bow	0.0018	−0.016671384	0.020271384	0.9997181
ft–ensemble	0.0142	−0.004271384	0.032671384	0.2237050
gtfidf–ensemble	0.0078	−0.010671384	0.026271384	0.8113536
sem–ensemble	0.0036	−0.014871384	0.022071384	0.9922183
w2v–ensemble	0.0106	−0.007871384	0.029071384	0.5407045

El método EQuAL posee una media de error que no posee diferencias significativas a todos los métodos. En todos los casos, los intervalos de confianza incluyen al valor 0 y $p-adj > \alpha$. La Figura 18 muestra un gráfico de caja y bigote para las medias de error en estudio y tamaño de muestra 500. En la misma, se puede visualizar que el método EQuAL posee el menor valor de error para una ejecución individual en particular, representado por el bigote inferior.

6.3.1.3. Tamaño de muestra de 1000 pares de preguntas

	diff	lwr	upr	p adj
ensemble–bow	0.0175	−0.004161633	0.039161633	0.1791584
ft–ensemble	−0.0049	−0.026561633	0.016761633	0.9846755
gtfidf–ensemble	−0.0072	−0.028861633	0.014461633	0.9217004

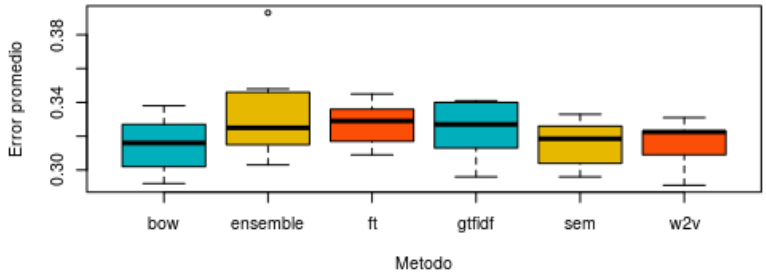

Figura 19: Gráfico de caja y bigote para las medias de error de los métodos en estudio para tamaño de muestra de 1000 pares de preguntas.

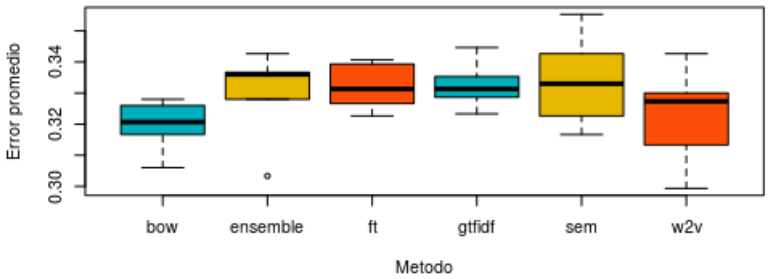

Figura 20: Gráfico de caja y bigote para las medias de error de los métodos en estudio para tamaño de muestra de 1500 pares de preguntas.

sem−ensemble	−0.0162	−0.037861633	0.005461633	0.2504112
w2v−ensemble	−0.0155	−0.037161633	0.006161633	0.2956972

El método EQuAL posee una media de error que no posee diferencias significativas a todos los métodos. En todos los casos, los intervalos de confianza incluyen al valor 0 y $p - adj > \alpha$. La Figura 19 muestra un gráfico de caja y bigote para las medias de error en estudio y tamaño de muestra 1000.

6.3.1.4. Tamaño de muestra de 1500 pares de preguntas

	diff	lwr	upr	p adj
ensemble−bow	0.0090666667	−7.427521e−03	0.025560854	0.5832109
ft−ensemble	0.0027333333	−1.376085e−02	0.019227521	0.9962642
gtfidf−ensemble	0.0028666667	−1.362752e−02	0.019360854	0.9953275
sem−ensemble	0.0044666667	−1.202752e−02	0.020960854	0.9656207
w2v−ensemble	−0.0062000000	−2.269419e−02	0.010294187	0.8729702

El método EQuAL posee una media de error que no posee diferencias significativas a todos los métodos. En todos los casos, los intervalos de confianza

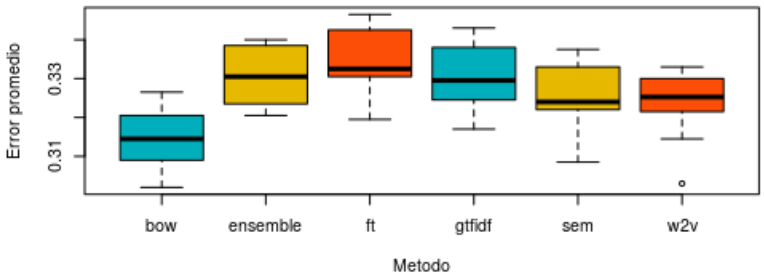

Figura 21: Gráfico de caja y bigote para las medias de error de los métodos en estudio para tamaño de muestra de 2000 pares de preguntas.

incluyen al valor 0 y $p - adj > \alpha$. La Figura 20 muestra un gráfico de caja y bigote para las medias de error en estudio y tamaño de muestra 1500.

6.3.1.5. Tamaño de muestra de 2000 pares de preguntas

	diff	lwr	upr	p adj
ensemble−bow	0.01585	0.001648	0.0300516	0.020523
ft−ensemble	0.00385	−0.010351	0.0180516	0.965462
gtfidf−ensemble	−0.00035	−0.014551	0.0138516	0.999999
sem−ensemble	−0.00570	−0.019901	0.0085016	0.839375
w2v−ensemble	−0.00725	−0.021451	0.0069516	0.657185

En este caso, el intervalo de confianza contra el método bow no incluye al cero, ya que este método tuvo muy buenos indicadores en este tamaño de muestra. No obstante, el método EQuAL tiene una diferencia significativa solo con el método bow, no habiendo diferencias significativas con los métodos restantes, ya que en esos casos, los intervalos de confianza incluyen al valor 0 y $p - adj > \alpha$. La Figura 21 muestra un gráfico de caja y bigote para las medias de error en estudio y tamaño de muestra 2000.

6.3.1.6. Resumen de resultados del análisis de varianza

En todos los tamaños de muestra, el método EQuAL no muestra diferencias significativas con ninguno de los algoritmos del estado del arte, ya que todos los intervalos de confianza contienen al valor cero, exceptuando la comparación EQuAL-bow para el tamaño de muestra de 2000 pares de preguntas. En este último caso, el método bow muestra una media de error significativamente más baja. En total, se realizaron 25 intervalos de confianza, y en solo uno el método EQuAL se obtuvo una media de error más alta, ya que el método bow fue significativamente mejor al resto.

En general, el método EQuAL tiene un buen comportamiento en cuanto a medias de error a lo largo de todos los tamaños de muestra. Esto significa que las esperanzas de error no tienen diferencias significativas con los métodos del estado del arte (los cuales son utilizados por la comunidad en el cálculo de similaridad y análisis de texto) y son, incluso, superadoras en algunos casos. Teniendo en cuenta este indicador, se puede concluir que el método EQuAL es apto para su implementación en RS.

6.3.2. Desempeño con tamaños de muestra pequeños

En el análisis de varianza de las medias de error realizado anteriormente, se puede observar que el método EQuAL se comporta muy bien en los tamaños de muestra de 100 y 500 pares de preguntas: el método propuesto fue estadísticamente igual o superior a los métodos del estado del arte. Sin embargo, para tamaños de muestra más grandes, fue superado en algunas oportunidades.

Esto demuestra que el agregado de variabilidad de datos puede ser influyente en el cálculo de similaridad. Los métodos del estado del arte comparan cada una de las preguntas de cada par entre sí, mientras que el método EQuAL realiza una comparación "todas contra todas" con el objetivo de realizar un método de clustering a continuación. Para clarificar, se puede concluir que los gráficos y estadísticas obtenidos muestran que métodos basados en ensamble de clustering, y en particular el método EQuAL, proponen una medida de similaridad adimensional que puede mejorar ciertas medidas de rendimiento en comparativa con otros algoritmos. Citando a Fred y Jain (2005), esta mejora se hace aún más evidente en tamaños de muestras chicos o conjuntos de datos complejos.

6.3.3. Dependencia de un método de ensamble con sus algoritmos subyacentes

Si bien es una medida adimensional y propone mejorar las medidas de rendimiento en ciertos aspectos, los métodos de ensamble de clustering son dependientes de sus algoritmos subyacentes y, por lo tanto, las medidas de desempeño son similares en cuanto a, por ejemplo, el valor de error obtenido. Los análisis de varianza realizados ponen en evidencia la dependencia del método equal con los algoritmos ensamblados. La media de error es estadísticamente similar en la mayoría de los casos, lo cual indica que los valores arrojados por el método

de ensamble están fuertemente relacionados con las medias subyacentes, dando la posibilidad de mejorar los indicadores de rendimiento cuando se utilicen algoritmos de similaridad de texto que tengan mejor comportamiento.

6.3.4. Influencia del conjunto de datos Quora

En las secciones anteriores se demostró que el método EQuAL es apto para la utilización en un sistema de recomendación. No obstante, no posee ventajas claras sobre los métodos del estado del arte en algunas medidas de rendimiento en particular.

El conjunto de datos Quora y la naturaleza del mismo, no permite realizar un diagrama de dispersión que posibilite ver la forma de los clusters generados. Dicho esto, y sabiendo que la técnica de Ensamble de Clustering es particularmente efectiva en la detección de clusters con formas no convencionales, no es posible identificar fácilmente cuál es la forma de los clusters en cuestión, y verificar si el método EQuAL se adapta perfectamente al conjunto de datos.

Por otro lado, el conjunto de datos compara preguntas una a una, que generalmente tienen una o más palabras en común. Esto provoca que los métodos como TF o TF-IDF que cuantifican la cantidad de palabras en común, se comporten relativamente bien o arrojen, al menos, un resultado distinto de cero que supera a los métodos "más inteligentes". Lo contrario ocurriría, si la comparación de texto fuese entre frases cortas que utilicen sinónimos o palabras completamente distintas, donde los algoritmos basados en taxonomías y técnicas de ML tendrían clara ventaja.

El método EQuAL, no es únicamente efectivo para algoritmos subyacentes basados en análisis de texto, sino que también es posible utilizarlo para cualquier conjunto de datos y técnica que generen matrices de distancia como salida. Tal es así, que este método, sobre la arquitectura propuesta, podría funcionar de manera excepcional con conjuntos de datos que puedan ser graficados y tengan clusters identificables visualmente, con formas que sean convenientes para métodos basados en Ensamble de Clustering.

6.4. Análisis de desempeño

El objetivo del análisis de desempeño es evaluar si la arquitectura propuesta es eficiente y escalable para la aplicación del método EQuAL. El cluster de prueba

consiste en un cluster local, por lo cual el paralelismo será realizado modificando la cantidad de núcleos CPU que se le provee al cluster Hadoop local. Esta configuración puede ser extrapolada fácilmente a un cluster de computadoras con más de un nodo, y con varios ejecutores por nodo.

Las pruebas son realizadas ejecutando los experimentos de forma local y variando la configuración del contexto Spark. Ya que en este trabajo, la etapa de clustering no se realizó utilizando Apache Spark, no es detallada en esta sección. Se analizan entonces, la etapa de cálculo de similaridad, el ensamble de clustering, y la ejecución total de los experimentos. El análisis será realizado con tamaños de muestra 100, 500 y 1000 pares de preguntas. Cada una de las ejecuciones, se realizan utilizando dos técnicas de similaridad (TF y TFIDF), para luego ensamblarlas. Además, con cada una de ellas, se realizó un algoritmo de clustering con $k = 5$ y 20 ejecuciones del mismo. Las pruebas de rendimiento se realizaron con el siguiente equipo:

```
Modelo: MacBook Pro
Procesador: Intel Core i7
Velocidad del procesador: 2.6 GHz
Numero de nucleos: 6
Caché L2 (por núcleo): 256 KB
Caché L3: 12 MB
Tecnología Hyper-Threading: Habilitada
Memoria: 16 GB RAM
Almacenamiento: APPLE SSD AP0512M
```

La cantidad de núcleos de CPU asignados al contexto Spark será de 1, 2, 4, 8 y 12. La Tabla 37 muestra, para un número de núcleos CPU y un tamaño de muestra, la cantidad de cálculos realizados, el tiempo total para la etapa de cálculo de matrices de similaridad, y una estimación de cálculos/segundo siendo la cantidad de cálculos total de $2n^2 - n$ por cada una de las matrices de similaridad generadas (dos en total en las pruebas de desempeño). La Tabla 38 muestra, para un número de núcleos CPU y un tamaño de muestra, la cantidad de cálculos realizados, el tiempo total para la etapa de ensamble de clustering, y una estimación de cálculos/segundo siendo la cantidad de cálculos total de N^2n^2 siendo N la cantidad de ejecuciones del algoritmo de clustering en total. Para finalizar, la Tabla 39 muestra, para un número de núcleos CPU y un tamaño de

muestra, la cantidad de cálculos realizados, y los tiempos totales de cada una de las ejecuciones. El tiempo total está compuesto un tiempo de preprocesamiento, cálculos de similaridades, algoritmos de clustering, y ensamble de clustering[57] . La cantidad estimada de cálculos total, es la sumatoria de los dos explicados anteriormente más una cantidad estimada para las ejecuciones del algoritmo de clustering $Nnki$, donde $N = 40$ ejecuciones, $k = 5$ clusters para $i = 300$ iteraciones.

Tabla 37: Cálculos aproximados y tiempo para la etapa de cálculo de matrices de similaridad para los distintos tamaños de muestra.

Cálculo de matrices de similaridad				
Núcleo CPU	Tamaño de muestra	Cálculos	Tiempo	Calc/seg
1	100	39996	62.990	634.957
2	100	39996	46.081	867.955
4	100	39996	33.074	1209.303
8	100	39996	31.276	1278.827
12	100	39996	32.336	1236.882
1	500	999996	763.889	1309.086
2	500	999996	576.669	1734.091
4	500	999996	402.376	2485.226
8	500	999996	324.826	3078.561
12	500	999996	328.300	3045.980
1	1,000	3999996	2950.810	1355.559
2	1,000	3999996	2073.179	1929.402
4	1,000	3999996	1423.895968	2809.191184
8	1,000	3999996	1176.5926	3399.644023
12	1,000	3999996	1253.433926	3191.230042

Graficando las Tablas 37, 38 y 39 para cada uno de los tamaños de muestra de forma sumarizada, se puede ver claramente como el desempeño aumenta hasta cierto nivel de paralelismo. En las Figuras 22, 23 y 24 se grafican los datos de rendimiento para los tamaños de muestra 100, 500 y 1000 respectivamente. En cada una de ellas, las curvas de cálculo de similaridad, ensamble y total son graficadas, para mostrar la relación entre el número de núcleos de CPU (en abscisas) y el tiempo en segundos (en ordenadas). En un cluster Spark, existe una cota superior al número de nodos/ejecutores/CPU en el cual más allá del mismo, el aumento del desempeño es marginal. Este límite depende de la naturaleza de la aplicación y difiere notablemente si la misma hace uso intensivo de CPU o uso intensivo de operaciones de entrada-salida. Además, es esperable que para

[57] Detalles de los tiempos de ejecución en la tabla 41

Tabla 38: Cálculos aproximados y tiempo para la etapa de ensamble de clustering para los distintos tamaños de muestra.

Ensamble de Clustering				
Núcleo CPU	Tamaño de muestra	Cálculos	Tiempo	Calc/seg
1	100	16000000	15.978	1001359.283
2	100	16000000	10.388	1540193.812
4	100	16000000	6.872	2328158.952
8	100	16000000	6.309	2536184.216
12	100	16000000	5.698	2808037.305
1	500	400000000	40.430	9893694.968
2	500	400000000	25.684	15573735.642
4	500	400000000	21.511	18594808.023
8	500	400000000	18.017	22201723.864
12	500	400000000	15.737	25417188.196
1	1,000	1600000000	95.886	16686498.744
2	1,000	1600000000	58.053	27560883.672
4	1,000	1600000000	47.732525	33520120.71
8	1,000	1600000000	43.265526	36980944.14
12	1,000	1600000000	43.699182	36613957.67

Tabla 39: Cálculos aproximados y tiempos totales para los distintos tamaños de muestra.

Total				
Núcleo CPU	Tamaño de muestra	Cálculos	Tiempo	Calc/seg
1	100	40039996	88.554	452154.152
2	100	40039996	64.830	617610.886
4	100	40039996	47.607	841058.532
8	100	40039996	45.024	889313.092
12	100	40039996	44.428	901233.021
1	500	520999996	828.022	629210.531
2	500	520999996	624.102	834799.734
4	500	520999996	446.984	1165589.860
8	500	520999996	363.617	1432826.642
12	500	520999996	362.982	1435333.416
1	1,000	1843999996	3106.846	593527.943
2	1,000	1843999996	2199.752	838276.366
4	1,000	1843999996	1530.309932	1204984.662
8	1,000	1843999996	1282.804328	1437475.658
12	1,000	1843999996	1370.473092	1345520.76

conjuntos de datos más grandes se necesiten más tareas de ejecución, para lograr un mejor desempeño.

El tiempo total de ejecución del método EQuAL decrece exponencialmente configurando un correcto nivel de paralelismo. Por lo cual, la arquitectura

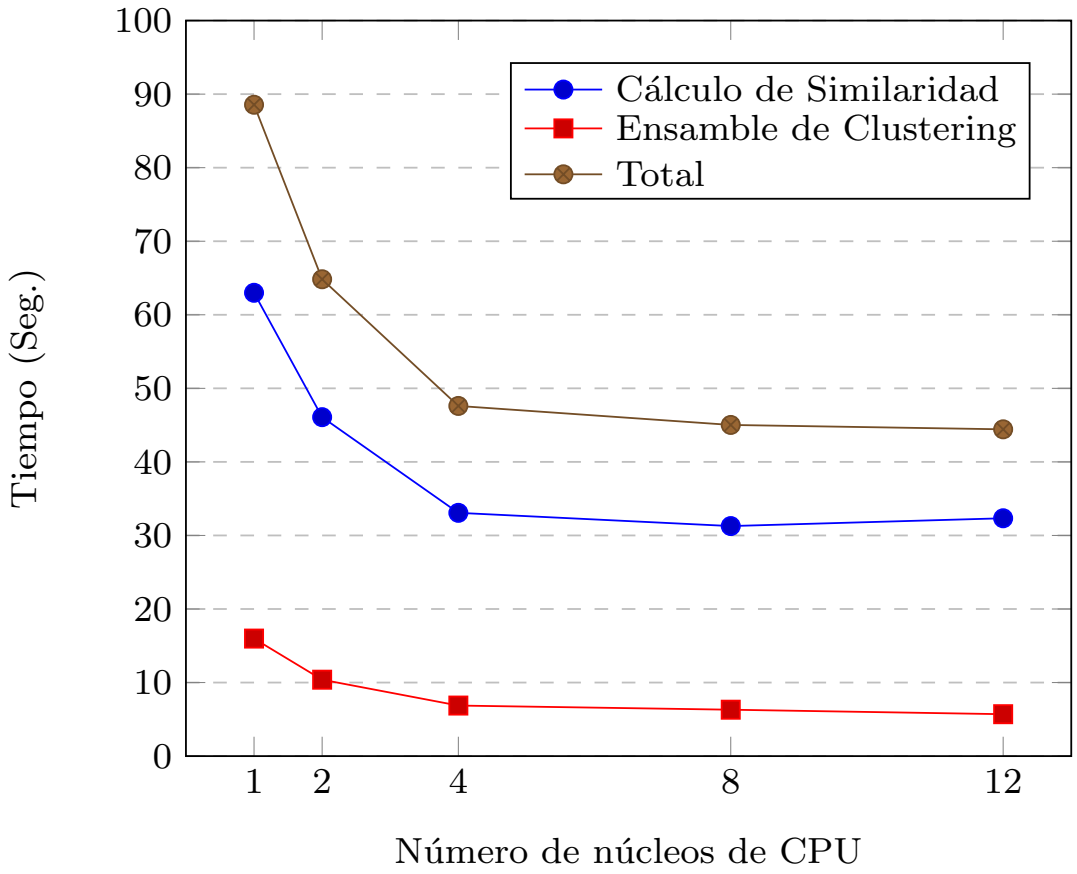

Figura 22: Tiempos en segundos para ejecuciones de tamaño de muestra de 100 pares de preguntas y distintos núcleos de CPU.

propuesta provee un método sencillo para escalar horizontalmente y poder procesar grandes cantidades de datos de forma eficiente. La configuración óptima del cluster de computadoras que soportará la arquitectura depende del tamaño del conjunto de datos de entrada.

6.5. Resumen de resultados

En el análisis del método EQuAL, mediante experimentación con distintos parámetros (tamaño de muestra y número de clusters), se obtuvo una media de error de 0,312 para el tamaño de muestra de 100 pares de preguntas y $k = 50$. Esta tendencia en los resultados demuestra dos particularidades: el método EQuAL tuvo buen rendimiento con tamaño de muestras chicas y con un alto número de clusters. Esta tendencia en tamaños de muestra chicos puede ser consecuencia del agregado de variabilidad que este método proporciona. Por otro lado, la tendencia a la baja de media de error es mucho más clara con valores de k altos, obteniendo medias de error muy buenas cuando $k > 25$ (0,309, 0,304, 0,307, 0,309,

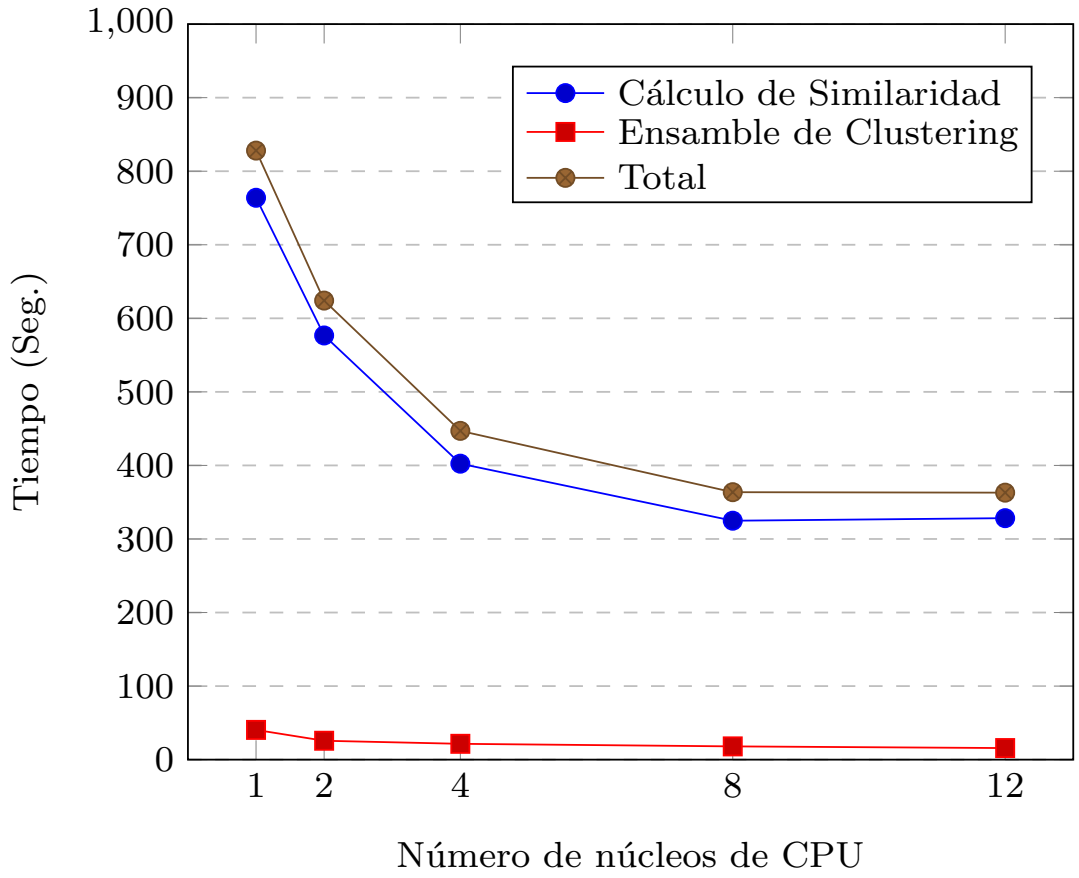

Figura 23: Tiempos en segundos para ejecuciones de tamaño de muestra de 500 pares de preguntas y distintos núcleos de CPU.

0,311, 0,308 para $k = 25, 30, 35, 40, 45, 50$). Por otro lado, comparando el método EQuAL con los algoritmos del estado del arte, se concluye que posee indicadores aptos para su aplicación en RS, en cuanto a medias de error y varianza. La media de error total del método EQuAL (teniendo en cuenta todas las ejecuciones realizadas en los experimentos) es de 0,32286, la cual no difiere demasiado de la mejor (bow con 0,31162) y supera a FastText y TF-IDF con 0,33124 y 0,32409 respectivamente.

Se realizó un Análisis de Varianza del método propuesto en contraste con los métodos del estado del arte para poder afirmar que el método EQuAL es apto para ser aplicado en RS, de forma eficiente y eficaz, de forma estadísticamente significativa. Los métodos analizados fueron bow, TF-IDF, FastText, Word2Vec y Semantic Distance y los tamaños de muestras tomados para el análisis fueron 100, 500, 1000, 1500 y 2000 pares de preguntas, completando así 25 intervalos de confianza. En la mayoría de los tamaños de muestra analizados, la media de error fue estadísticamente igual en comparación con cada uno de los métodos

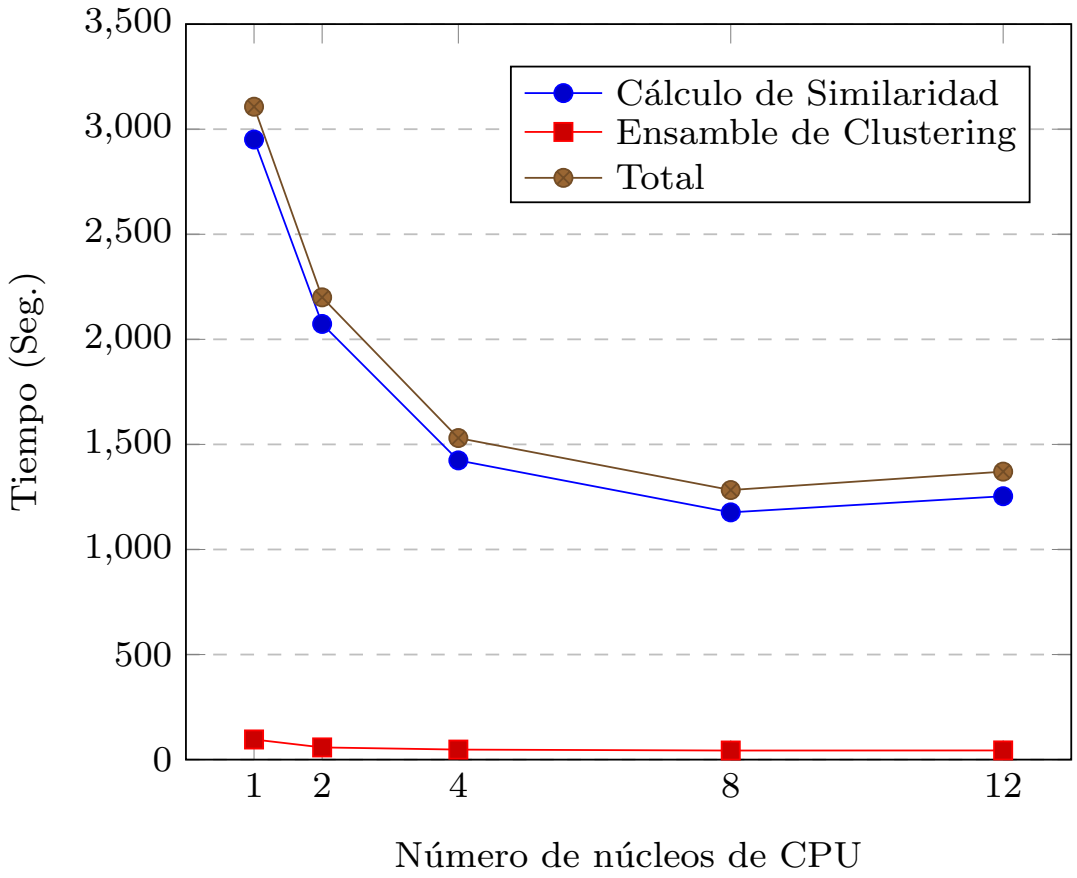

Figura 24: Tiempos en segundos para ejecuciones de tamaño de muestra de 1000 pares de preguntas y distintos núcleos de CPU.

del estado del arte, con excepción de una de la comparación EQuAL-bow para el tamaño de muestra de 2000 pares de preguntas, en el cual bow obtuvo una media de error significativamente más baja que el resto de los algoritmos.

Adicionalmente, es posible afirmar que el método propuesto depende tanto de los algoritmos subyacentes como del conjunto de datos de entrada. Por un lado, es altamente probable que arroje buenos resultados si los algoritmos subyacentes también lo hacen, y viceversa. Por otro lado, es posible adaptarlo al conjunto de datos y elegir los algoritmos subyacentes adecuados para el mismo, dando versatilidad y un resultado compuesto por características aportadas por cada uno de ellos.

Para finalizar, se desarrolló una arquitectura de software que realiza los cálculos de similaridad y procesamiento del ensamble de clustering de manera adaptable. Esto significa que es posible ejecutar cálculos de matrices de similaridad con un número variable de técnicas de comparación de análisis de texto (es posible agregar o quitar una nueva técnica fácilmente) y luego ensamblar sus resultados.

La arquitectura obtenida se ejecuta en forma distribuida y escalable horizontalmente, permitiendo incrementar el rendimiento de manera exponencial solo con un cambio de configuración de la infraestructura base (número de nodos en el cluster, memoria RAM asignada a cada ejecutor, almacenamiento, y/o tipo de servidores) y de la configuración Spark sin tener que modificar el código en absoluto.

Capítulo 7. **Conclusiones**

7.1. Contribuciones realizadas

A lo largo de este trabajo de Tesis se realizaron diversos aportes relativos a la propuesta de un nuevo método de similaridad de texto basado en ensamble de clustering, una propuesta de arquitectura de software para el procesamiento de datos y su futura aplicación en sistemas de recomendación de tiempo real. El objetivo final del mismo es que estos aportes puedan ser aplicados en un sistema de recomendación de un sitio de Community Question Answering (CQA) para agregar la capacidad de recomendar preguntas similares a una pregunta dada, y así minimizar el tiempo en que un usuario puede encontrar la respuesta que necesita.

Como primer aporte, se diseñó un método que utiliza una medida de similaridad de texto confiable y efectiva entre preguntas de un sitio de CQA. El conjunto de datos utilizado proviene de un sitio web llamado Quora. Para el mismo, se utilizó una técnica de ensamble de clustering, utilizando como base 5 técnicas de similaridad verificadas y utilizadas ampliamente por la comunidad, generando una salida con información de similaridad entre pares de preguntas. El método, llamado EQuAL, se basa en dos etapas principales: una generación de particiones y la generación de una matriz de co-asociación que contiene la información de similaridad entre las preguntas. La primera etapa realiza un muestreo del conjunto de datos original, aplica cada uno de los métodos del estado del arte (Term Frequency, Term Frequency/Inverse Document Frequency, Word2Vec, FastText y Semantic Distance) y genera una matriz de similaridad por cada uno de ellos; para luego asignar cada una de las preguntas en un cluster, utilizando una técnica de clustering PAM (Partición Alrededor de Medoids). Adicionalmente, la segunda etapa obtiene todas las etiquetas generadas por el paso anterior para generar una matriz de co-asociación mediante un algoritmo de ensamble de clustering. La matriz de co-asociación contiene todas las relaciones entre preguntas individuales

en forma de similaridad, de manera estandarizada (en un rango $[0, 1]$) y adimensional. La experimentación del método se implementó mediante código basado en Python y Apache Spark. Como resultado de los experimentos, se observa que el método EQuAL se comporta comparativamente similar a los del estado del arte y muestra muy buenos resultados para todos los tamaños de muestra en los que se realizaron experimentos de validación, manteniendo buenos indicadores como una media de error de 0,32286 y una varianza media de 0,00049. Se encontró, también, que existe una independencia del tamaño de muestra, lo que indica que la variabilidad agregada por el método de ensamble provoca buenos resultados. Lo anterior significa que los indicadores de rendimiento obtenidos son significativamente comparables con los métodos del estado del arte, e inclusive superador en algunos casos.

Por otro lado, y como medio posibilitador del primer aporte, se propuso un segundo aporte: una arquitectura de software de procesamiento distribuido basado en Big Data. La misma está basada en una arquitectura Hadoop que funciona sobre un cluster de computadoras y permite la ejecución de cálculos en forma distribuida. Se demostró, mediante los experimentos realizados para este trabajo de tesis, que la misma posibilita buena flexibilidad a la hora de escalar horizontalmente y de configurar una infraestructura adecuada. Esto significa que el paralelismo otorgado por la arquitectura propuesta habilita un desempeño que permitirá escalar la cantidad de datos de una forma sencilla y eficiente.

Si bien el método fue probado mediante un desarrollo que se ejecuta de forma fuera de línea, el mismo puede ser adaptado para su utilización en tiempo real para generar información de similaridad de forma eficiente para un ítem de recomendación nuevo. Por este motivo, se proporcionó una base teórica y práctica que puede ser implementada de forma eficiente en varios aspectos de un sistema de recomendación en tiempo real basado en similaridad de texto.

7.2. Futuras investigaciones

A partir de las contribuciones realizadas, es posible derivar nuevas investigaciones. Considerando la generación de una medida de similaridad de texto, no solo para preguntas en sitios de CQA, sino para cualquier tipo de fragmentos de texto que se desee comparar y, teniendo la posibilidad de extrapolarlo a cualquier

sitio Web, sistema o almacén de datos en general, es posible reconocer algunas líneas potenciales de investigación tales como:

- Continuar con el desarrollo para lograr un RS operativo en su totalidad utilizando el método propuesto basado en ensamble de clustering y similaridad entre ítems. Para tal fin, es necesario enfocarse en una infraestructura adecuada para recomendar ítems en tiempo real, teniendo en cuenta buenas prácticas relacionadas con la puesta en producción de un modelo de Machine Learning/Recomendación. Algunas de las opciones posibles a ser aplicadas fueron descritas en el apartado 4.5.4.

- Elaborar una arquitectura Big Data adaptable que mejore y optimice el funcionamiento de algunos aspectos. Por ejemplo, expandir su adaptabilidad para poder utilizarla para la aplicación de otros procesos de Clustering o algoritmos de Deep Learning. Tener en cuenta y planificar soluciones a grandes desafíos en el momento de desarrollo de un sistema de recomendación completo: serialización de modelos de ML cuando se procesan en forma distribuida, la aplicación de modelos de ML en tiempo real (Machine Learning Inference[58]), comparación del desempeño en términos de eficiencia con distintos algoritmos y herramientas, entre otros.

- Utilizar los resultados obtenidos en otros tipos de sitios donde se puedan aplicar RS basados en texto, tales como sitios de e-commerce, portales académicos o redes sociales. Adicionalmente, es posible el uso del método de ensamble de clustering y la arquitectura desarrollada para sistemas de recomendación que contengan ítems más complejos que texto, pero que puedan ser representados de forma vectorial o que sea posible el cálculo de distancia/similaridad entre ellos. Por ejemplo, construir una representación vectorial de restaurantes a recomendar, basándose en el nombre del mismo, su localización geográfica, tipo de cocina, menú disponible, entre otras características.

- Continuar el desarrollo para crear un framework adaptable a distintas técnicas de distancias de texto para que sea sencillo agregar/quitar cada una de ellas, y así facilitar la investigación en la combinación de las mismas mediante ensamble de clustering. Por otro lado, realizar experimentos con

[58] Este concepto es usado cuando un modelo entrenado de ML predice la salida a una entrada en tiempo real.

distintos conjuntos de datos de entrada que puedan ofrecer comportamientos diferentes al aplicarse en un proceso de Ensamble de Clustering.

- Crear y estructurar información para policy makers e instituciones de ciencia, tecnología, innovación y desarrollo con el objetivo de construir insumos para el diseño, implementación, ejecución y evaluación de políticas públicas y educativas. De esta manera, se colaborará en la mejora y optimización de todos los aspectos referidos a los procesos de búsqueda (encontrar información relevante, país donde fue publicada, naturaleza —sea institucional o personal—, fuentes —bibliotecas virtuales, repositorios digitales, bases científicas, revistas científicas open source—), y sus resultados (construir datos en tiempo real, identificar referentes en los temas, establecer contactos y poder construir prácticas colaborativas). El conjunto de estas acciones, y su análisis, ofrece un panorama para diseñar y construir acciones de manera estratégica, adecuándose a fines y objetivos específicos y coyunturales.

Anexos

Tabla 40: Anexo de matrices de confusión utilizados para construir las secciones 6.1 and 6.2.

Técnica	**Tam. muestra**	**0/0**	**0/1**	**1/0**	**1/1**	**Precisión**	**Error**
bow	100	0.3800	0.1900	0.0700	0.3600	0.7400	0.2600
bow	100	0.5400	0.0700	0.2100	0.1800	0.7200	0.2800
bow	100	0.5600	0.0300	0.3100	0.1000	0.6600	0.3400
bow	100	0.3100	0.2800	0.0800	0.3300	0.6400	0.3600
bow	100	0.3600	0.2200	0.0600	0.3600	0.7200	0.2800
bow	100	0.5700	0.0400	0.2800	0.1100	0.6800	0.3200
bow	100	0.3700	0.2300	0.1200	0.2800	0.6500	0.3500
bow	100	0.4100	0.2000	0.0500	0.3400	0.7500	0.2500
bow	100	0.3300	0.2500	0.0400	0.3800	0.7100	0.2900
bow	100	0.4600	0.1400	0.0900	0.3100	0.7700	0.2300
bow	500	0.3540	0.2460	0.0640	0.3360	0.6900	0.3100
bow	500	0.3240	0.2560	0.0600	0.3600	0.6840	0.3160
bow	500	0.3780	0.2300	0.0920	0.3000	0.6780	0.3220
bow	500	0.3980	0.2020	0.0940	0.3060	0.7040	0.2960
bow	500	0.4280	0.1780	0.1440	0.2500	0.6780	0.3220
bow	500	0.4400	0.1600	0.1460	0.2540	0.6940	0.3060
bow	500	0.4140	0.1940	0.1340	0.2580	0.6720	0.3280
bow	500	0.4240	0.1780	0.1200	0.2780	0.7020	0.2980
bow	500	0.3540	0.2560	0.0600	0.3300	0.6840	0.3160
bow	500	0.4340	0.1720	0.1360	0.2580	0.6920	0.3080
bow	1000	0.3600	0.2500	0.0590	0.3310	0.6910	0.3090
bow	1000	0.4310	0.1740	0.1280	0.2670	0.6980	0.3020
bow	1000	0.3860	0.2100	0.1070	0.2970	0.6830	0.3170
bow	1000	0.3530	0.2480	0.0670	0.3320	0.6850	0.3150
bow	1000	0.3840	0.2260	0.1060	0.2840	0.6680	0.3320
bow	1000	0.3820	0.2270	0.1000	0.2910	0.6730	0.3270
bow	1000	0.4430	0.1600	0.1320	0.2650	0.7080	0.2920
bow	1000	0.3280	0.2760	0.0620	0.3340	0.6620	0.3380
bow	1000	0.4470	0.1570	0.1380	0.2580	0.7050	0.2950
bow	1000	0.3460	0.2610	0.0610	0.3320	0.6780	0.3220
bow	1500	0.4113	0.1967	0.1253	0.2667	0.6780	0.3220
bow	1500	0.3947	0.2147	0.1007	0.2900	0.6847	0.3153
bow	1500	0.3900	0.2147	0.1047	0.2907	0.6807	0.3193
bow	1500	0.4147	0.1920	0.1353	0.2580	0.6727	0.3273
bow	1500	0.4113	0.1947	0.1220	0.2720	0.6833	0.3167
bow	1500	0.4127	0.1900	0.1353	0.2620	0.6747	0.3253

Tabla 40: Anexo de matrices de confusión utilizados para construir las secciones 6.1 and 6.2.

Técnica	**Tam. muestra**	**0/0**	**0/1**	**1/0**	**1/1**	**Precisión**	**Error**
bow	1500	0.3560	0.2453	0.0607	0.3380	0.6940	0.3060
bow	1500	0.4520	0.1573	0.1593	0.2313	0.6833	0.3167
bow	1500	0.4253	0.1833	0.1427	0.2487	0.6740	0.3260
bow	1500	0.4027	0.1913	0.1367	0.2693	0.6720	0.3280
bow	2000	0.4035	0.2045	0.1010	0.2910	0.6945	0.3055
bow	2000	0.3940	0.2130	0.1075	0.2855	0.6795	0.3205
bow	2000	0.4015	0.2045	0.0975	0.2965	0.6980	0.3020
bow	2000	0.4030	0.2050	0.1040	0.2880	0.6910	0.3090
bow	2000	0.4290	0.1805	0.1320	0.2585	0.6875	0.3125
bow	2000	0.4290	0.1795	0.1325	0.2590	0.6880	0.3120
bow	2000	0.4135	0.1960	0.1305	0.2600	0.6735	0.3265
bow	2000	0.3775	0.2265	0.0970	0.2990	0.6765	0.3235
bow	2000	0.3840	0.2205	0.0990	0.2965	0.6805	0.3195
bow	2000	0.3850	0.2155	0.1010	0.2985	0.6835	0.3165
ft	100	0.3500	0.2200	0.1000	0.3300	0.6800	0.3200
ft	100	0.5400	0.0700	0.2300	0.1600	0.7000	0.3000
ft	100	0.2400	0.3500	0.0500	0.3600	0.6000	0.4000
ft	100	0.4400	0.1500	0.2100	0.2000	0.6400	0.3600
ft	100	0.4500	0.1300	0.2400	0.1800	0.6300	0.3700
ft	100	0.4800	0.1300	0.2200	0.1700	0.6500	0.3500
ft	100	0.3000	0.3000	0.0800	0.3200	0.6200	0.3800
ft	100	0.3800	0.2300	0.0400	0.3500	0.7300	0.2700
ft	100	0.4200	0.1600	0.1900	0.2300	0.6500	0.3500
ft	100	0.5300	0.0700	0.1700	0.2300	0.7600	0.2400
ft	500	0.4920	0.1080	0.2140	0.1860	0.6780	0.3220
ft	500	0.3120	0.2680	0.0740	0.3460	0.6580	0.3420
ft	500	0.4700	0.1380	0.1980	0.1940	0.6640	0.3360
ft	500	0.4940	0.1060	0.2140	0.1860	0.6800	0.3200
ft	500	0.4960	0.1100	0.2240	0.1700	0.6660	0.3340
ft	500	0.5020	0.0980	0.1980	0.2020	0.7040	0.2960
ft	500	0.4860	0.1220	0.2220	0.1700	0.6560	0.3440
ft	500	0.4980	0.1040	0.2080	0.1900	0.6880	0.3120
ft	500	0.4840	0.1260	0.2160	0.1740	0.6580	0.3420
ft	500	0.4900	0.1160	0.2180	0.1760	0.6660	0.3340
ft	1000	0.5030	0.1070	0.2090	0.1810	0.6840	0.3160
ft	1000	0.4820	0.1230	0.2100	0.1850	0.6670	0.3330
ft	1000	0.4710	0.1250	0.2110	0.1930	0.6640	0.3360
ft	1000	0.3480	0.2530	0.0650	0.3340	0.6820	0.3180
ft	1000	0.4950	0.1150	0.2140	0.1760	0.6710	0.3290
ft	1000	0.3310	0.2780	0.0650	0.3260	0.6570	0.3430
ft	1000	0.5090	0.0940	0.2150	0.1820	0.6910	0.3090
ft	1000	0.4830	0.1210	0.1960	0.2000	0.6830	0.3170
ft	1000	0.3550	0.2490	0.0800	0.3160	0.6710	0.3290
ft	1000	0.3450	0.2620	0.0830	0.3100	0.6550	0.3450
ft	1500	0.3633	0.2447	0.0880	0.3040	0.6673	0.3327
ft	1500	0.5073	0.1020	0.2207	0.1700	0.6773	0.3227
ft	1500	0.4827	0.1220	0.2013	0.1940	0.6767	0.3233
ft	1500	0.4887	0.1180	0.2107	0.1827	0.6713	0.3287
ft	1500	0.4740	0.1320	0.2087	0.1853	0.6593	0.3407

Tabla 40: Anexo de matrices de confusión utilizados para construir las secciones 6.1 and 6.2.

Técnica	Tam. muestra	0/0	0/1	1/0	1/1	Precisión	Error
ft	1500	0.4747	0.1280	0.2127	0.1847	0.6593	0.3407
ft	1500	0.4913	0.1100	0.2200	0.1787	0.6700	0.3300
ft	1500	0.4920	0.1173	0.2187	0.1720	0.6640	0.3360
ft	1500	0.4807	0.1280	0.1987	0.1927	0.6733	0.3267
ft	1500	0.4820	0.1120	0.2273	0.1787	0.6607	0.3393
ft	2000	0.4935	0.1145	0.2050	0.1870	0.6805	0.3195
ft	2000	0.3320	0.2750	0.0675	0.3255	0.6575	0.3425
ft	2000	0.4790	0.1270	0.2035	0.1905	0.6695	0.3305
ft	2000	0.4955	0.1125	0.2170	0.1750	0.6705	0.3295
ft	2000	0.4900	0.1195	0.2110	0.1795	0.6695	0.3305
ft	2000	0.4940	0.1145	0.2165	0.1750	0.6690	0.3310
ft	2000	0.4770	0.1325	0.2015	0.1890	0.6660	0.3340
ft	2000	0.4815	0.1225	0.2140	0.1820	0.6635	0.3365
ft	2000	0.4730	0.1315	0.2125	0.1830	0.6560	0.3440
ft	2000	0.4710	0.1295	0.2170	0.1825	0.6535	0.3465
w2v	100	0.4500	0.1200	0.1600	0.2700	0.7200	0.2800
w2v	100	0.4400	0.1700	0.1300	0.2600	0.7000	0.3000
w2v	100	0.1900	0.4000	0.0200	0.3900	0.5800	0.4200
w2v	100	0.3700	0.2200	0.1600	0.2500	0.6200	0.3800
w2v	100	0.3900	0.1900	0.1200	0.3000	0.6900	0.3100
w2v	100	0.5200	0.0900	0.2200	0.1700	0.6900	0.3100
w2v	100	0.4100	0.1900	0.1700	0.2300	0.6400	0.3600
w2v	100	0.4500	0.1600	0.0900	0.3000	0.7500	0.2500
w2v	100	0.3300	0.2500	0.0600	0.3600	0.6900	0.3100
w2v	100	0.4100	0.1900	0.0600	0.3400	0.7500	0.2500
w2v	500	0.3420	0.2580	0.0600	0.3400	0.6820	0.3180
w2v	500	0.3120	0.2680	0.0660	0.3540	0.6660	0.3340
w2v	500	0.4860	0.1220	0.2100	0.1820	0.6680	0.3320
w2v	500	0.3380	0.2620	0.0600	0.3400	0.6780	0.3220
w2v	500	0.4100	0.1960	0.1260	0.2680	0.6780	0.3220
w2v	500	0.3520	0.2480	0.0660	0.3340	0.6860	0.3140
w2v	500	0.3180	0.2900	0.0580	0.3340	0.6520	0.3480
w2v	500	0.3500	0.2520	0.0560	0.3420	0.6920	0.3080
w2v	500	0.4140	0.1960	0.1320	0.2580	0.6720	0.3280
w2v	500	0.4360	0.1700	0.1500	0.2440	0.6800	0.3200
w2v	1000	0.4250	0.1850	0.1260	0.2640	0.6890	0.3110
w2v	1000	0.4230	0.1820	0.1270	0.2680	0.6910	0.3090
w2v	1000	0.4080	0.1880	0.1350	0.2690	0.6770	0.3230
w2v	1000	0.4210	0.1800	0.1280	0.2710	0.6920	0.3080
w2v	1000	0.4150	0.1950	0.1360	0.2540	0.6690	0.3310
w2v	1000	0.4130	0.1960	0.1320	0.2590	0.6720	0.3280
w2v	1000	0.3680	0.2350	0.0560	0.3410	0.7090	0.2910
w2v	1000	0.4150	0.1890	0.1340	0.2620	0.6770	0.3230
w2v	1000	0.4190	0.1850	0.1380	0.2580	0.6770	0.3230
w2v	1000	0.4270	0.1800	0.1420	0.2510	0.6780	0.3220
w2v	1500	0.4333	0.1747	0.1473	0.2447	0.6780	0.3220
w2v	1500	0.4347	0.1747	0.1247	0.2660	0.7007	0.2993
w2v	1500	0.4147	0.1900	0.1233	0.2720	0.6867	0.3133
w2v	1500	0.4127	0.1940	0.1340	0.2593	0.6720	0.3280

Tabla 40: Anexo de matrices de confusión utilizados para construir las secciones 6.1 and 6.2.

Técnica	Tam. muestra	0/0	0/1	1/0	1/1	Precisión	Error
w2v	1500	0.4140	0.1920	0.1380	0.2560	0.6700	0.3300
w2v	1500	0.4073	0.1953	0.1340	0.2633	0.6707	0.3293
w2v	1500	0.4207	0.1807	0.1460	0.2527	0.6733	0.3267
w2v	1500	0.4193	0.1900	0.1433	0.2473	0.6667	0.3333
w2v	1500	0.4227	0.1860	0.1207	0.2707	0.6933	0.3067
w2v	1500	0.4867	0.1073	0.2353	0.1707	0.6573	0.3427
w2v	2000	0.4290	0.1790	0.1355	0.2565	0.6855	0.3145
w2v	2000	0.4190	0.1880	0.1345	0.2585	0.6775	0.3225
w2v	2000	0.4215	0.1845	0.1185	0.2755	0.6970	0.3030
w2v	2000	0.4230	0.1850	0.1410	0.2510	0.6740	0.3260
w2v	2000	0.4160	0.1935	0.1350	0.2555	0.6715	0.3285
w2v	2000	0.3470	0.2615	0.0630	0.3285	0.6755	0.3245
w2v	2000	0.4170	0.1925	0.1290	0.2615	0.6785	0.3215
w2v	2000	0.4070	0.1970	0.1360	0.2600	0.6670	0.3330
w2v	2000	0.4105	0.1940	0.1360	0.2595	0.6700	0.3300
w2v	2000	0.4085	0.1920	0.1380	0.2615	0.6700	0.3300
gtfidf	100	0.4400	0.1300	0.1200	0.3100	0.7500	0.2500
gtfidf	100	0.5800	0.0300	0.2700	0.1200	0.7000	0.3000
gtfidf	100	0.5500	0.0400	0.2800	0.1300	0.6800	0.3200
gtfidf	100	0.3200	0.2700	0.1200	0.2900	0.6100	0.3900
gtfidf	100	0.3900	0.1900	0.1000	0.3200	0.7100	0.2900
gtfidf	100	0.5900	0.0200	0.3400	0.0500	0.6400	0.3600
gtfidf	100	0.4400	0.1600	0.2000	0.2000	0.6400	0.3600
gtfidf	100	0.5400	0.0700	0.2200	0.1700	0.7100	0.2900
gtfidf	100	0.4000	0.1800	0.1000	0.3200	0.7200	0.2800
gtfidf	100	0.5500	0.0500	0.2200	0.1800	0.7300	0.2700
gtfidf	500	0.4440	0.1560	0.1640	0.2360	0.6800	0.3200
gtfidf	500	0.4040	0.1760	0.1600	0.2600	0.6640	0.3360
gtfidf	500	0.4120	0.1960	0.1240	0.2680	0.6800	0.3200
gtfidf	500	0.4620	0.1380	0.1620	0.2380	0.7000	0.3000
gtfidf	500	0.5300	0.0760	0.2660	0.1280	0.6580	0.3420
gtfidf	500	0.4500	0.1500	0.1540	0.2460	0.6960	0.3040
gtfidf	500	0.4060	0.2020	0.1460	0.2460	0.6520	0.3480
gtfidf	500	0.3920	0.2100	0.1100	0.2880	0.6800	0.3200
gtfidf	500	0.3540	0.2560	0.0720	0.3180	0.6720	0.3280
gtfidf	500	0.4920	0.1140	0.1860	0.2080	0.7000	0.3000
gtfidf	1000	0.3840	0.2260	0.1030	0.2870	0.6710	0.3290
gtfidf	1000	0.4460	0.1590	0.1540	0.2410	0.6870	0.3130
gtfidf	1000	0.3520	0.2440	0.0810	0.3230	0.6750	0.3250
gtfidf	1000	0.4190	0.1820	0.1550	0.2440	0.6630	0.3370
gtfidf	1000	0.4090	0.2010	0.1210	0.2690	0.6780	0.3220
gtfidf	1000	0.4840	0.1250	0.2150	0.1760	0.6600	0.3400
gtfidf	1000	0.4310	0.1720	0.1240	0.2730	0.7040	0.2960
gtfidf	1000	0.3910	0.2130	0.1280	0.2680	0.6590	0.3410
gtfidf	1000	0.4700	0.1340	0.1740	0.2220	0.6920	0.3080
gtfidf	1000	0.3750	0.2320	0.1090	0.2840	0.6590	0.3410
gtfidf	1500	0.4213	0.1867	0.1387	0.2533	0.6747	0.3253
gtfidf	1500	0.4293	0.1800	0.1647	0.2260	0.6553	0.3447
gtfidf	1500	0.4033	0.2013	0.1333	0.2620	0.6653	0.3347

Tabla 40: Anexo de matrices de confusión utilizados para construir las secciones 6.1 and 6.2.

Técnica	Tam. muestra	0/0	0/1	1/0	1/1	Precisión	Error
gtfidf	1500	0.4020	0.2047	0.1327	0.2607	0.6627	0.3373
gtfidf	1500	0.4527	0.1533	0.1753	0.2187	0.6713	0.3287
gtfidf	1500	0.3953	0.2073	0.1280	0.2693	0.6647	0.3353
gtfidf	1500	0.3807	0.2207	0.1027	0.2960	0.6767	0.3233
gtfidf	1500	0.4460	0.1633	0.1667	0.2240	0.6700	0.3300
gtfidf	1500	0.4693	0.1393	0.1927	0.1987	0.6680	0.3320
gtfidf	1500	0.4173	0.1767	0.1540	0.2520	0.6693	0.3307
gtfidf	2000	0.4130	0.1950	0.1240	0.2680	0.6810	0.3190
gtfidf	2000	0.4045	0.2025	0.1355	0.2575	0.6620	0.3380
gtfidf	2000	0.4145	0.1915	0.1255	0.2685	0.6830	0.3170
gtfidf	2000	0.4695	0.1385	0.1860	0.2060	0.6755	0.3245
gtfidf	2000	0.4605	0.1490	0.1800	0.2105	0.6710	0.3290
gtfidf	2000	0.4370	0.1715	0.1530	0.2385	0.6755	0.3245
gtfidf	2000	0.5015	0.1080	0.2345	0.1560	0.6575	0.3425
gtfidf	2000	0.4240	0.1800	0.1550	0.2410	0.6650	0.3350
gtfidf	2000	0.4285	0.1760	0.1540	0.2415	0.6700	0.3300
gtfidf	2000	0.3725	0.2280	0.1150	0.2845	0.6570	0.3430
sem	100	0.4800	0.0900	0.1600	0.2700	0.7500	0.2500
sem	100	0.6000	0.0100	0.2900	0.1000	0.7000	0.3000
sem	100	0.2800	0.3100	0.0400	0.3700	0.6500	0.3500
sem	100	0.4800	0.1100	0.2200	0.1900	0.6700	0.3300
sem	100	0.4200	0.1600	0.1400	0.2800	0.7000	0.3000
sem	100	0.5800	0.0300	0.2900	0.1000	0.6800	0.3200
sem	100	0.4000	0.2000	0.1400	0.2600	0.6600	0.3400
sem	100	0.5800	0.0300	0.2900	0.1000	0.6800	0.3200
sem	100	0.4100	0.1700	0.1100	0.3100	0.7200	0.2800
sem	100	0.5000	0.1000	0.1200	0.2800	0.7800	0.2200
sem	500	0.4580	0.1420	0.1780	0.2220	0.6800	0.3200
sem	500	0.3180	0.2620	0.0720	0.3480	0.6660	0.3340
sem	500	0.5740	0.0340	0.2800	0.1120	0.6860	0.3140
sem	500	0.5240	0.0760	0.2320	0.1680	0.6920	0.3080
sem	500	0.4460	0.1600	0.1360	0.2580	0.7040	0.2960
sem	500	0.4420	0.1580	0.1380	0.2620	0.7040	0.2960
sem	500	0.4800	0.1280	0.1920	0.2000	0.6800	0.3200
sem	500	0.4640	0.1380	0.2020	0.1960	0.6600	0.3400
sem	500	0.5240	0.0860	0.2300	0.1600	0.6840	0.3160
sem	500	0.4180	0.1880	0.1440	0.2500	0.6680	0.3320
sem	1000	0.4840	0.1260	0.1880	0.2020	0.6860	0.3140
sem	1000	0.4450	0.1600	0.1360	0.2590	0.7040	0.2960
sem	1000	0.4260	0.1700	0.1380	0.2660	0.6920	0.3080
sem	1000	0.4420	0.1590	0.1450	0.2540	0.6960	0.3040
sem	1000	0.5390	0.0710	0.2600	0.1300	0.6690	0.3310
sem	1000	0.4780	0.1310	0.1950	0.1960	0.6740	0.3260
sem	1000	0.4950	0.1080	0.1950	0.2020	0.6970	0.3030
sem	1000	0.4640	0.1400	0.1930	0.2030	0.6670	0.3330
sem	1000	0.4230	0.1810	0.1430	0.2530	0.6760	0.3240
sem	1000	0.4430	0.1640	0.1590	0.2340	0.6770	0.3230
sem	1500	0.4447	0.1633	0.1533	0.2387	0.6833	0.3167
sem	1500	0.4327	0.1767	0.1447	0.2460	0.6787	0.3213

Tabla 40: Anexo de matrices de confusión utilizados para construir las secciones 6.1 and 6.2.

Técnica	Tam. muestra	0/0	0/1	1/0	1/1	Precisión	Error
sem	1500	0.4247	0.1800	0.1507	0.2447	0.6693	0.3307
sem	1500	0.4693	0.1373	0.1907	0.2027	0.6720	0.3280
sem	1500	0.5153	0.0907	0.2520	0.1420	0.6573	0.3427
sem	1500	0.4653	0.1373	0.1980	0.1993	0.6647	0.3353
sem	1500	0.5173	0.0840	0.2713	0.1273	0.6447	0.3553
sem	1500	0.4320	0.1773	0.1607	0.2300	0.6620	0.3380
sem	1500	0.4713	0.1373	0.1853	0.2060	0.6773	0.3227
sem	1500	0.4573	0.1367	0.2107	0.1953	0.6527	0.3473
sem	2000	0.5285	0.0795	0.2290	0.1630	0.6915	0.3085
sem	2000	0.4770	0.1300	0.2030	0.1900	0.6670	0.3330
sem	2000	0.4785	0.1275	0.1970	0.1970	0.6755	0.3245
sem	2000	0.4880	0.1200	0.2020	0.1900	0.6780	0.3220
sem	2000	0.4880	0.1215	0.2010	0.1895	0.6775	0.3225
sem	2000	0.5225	0.0860	0.2515	0.1400	0.6625	0.3375
sem	2000	0.4770	0.1325	0.1910	0.1995	0.6765	0.3235
sem	2000	0.4400	0.1640	0.1495	0.2465	0.6865	0.3135
sem	2000	0.5250	0.0795	0.2470	0.1485	0.6735	0.3265
sem	2000	0.4625	0.1380	0.1995	0.2000	0.6625	0.3375
ensemble	100	0.4200	0.1500	0.1000	0.3300	0.7500	0.2500
ensemble	100	0.5100	0.1300	0.1400	0.2200	0.7300	0.2700
ensemble	100	0.5800	0.0100	0.3700	0.0400	0.6200	0.3800
ensemble	100	0.2200	0.3700	0.0100	0.4000	0.6200	0.3800
ensemble	100	0.4400	0.1400	0.1800	0.2400	0.6800	0.3200
ensemble	100	0.5000	0.1100	0.1700	0.2200	0.7200	0.2800
ensemble	100	0.4100	0.1900	0.1900	0.2100	0.6200	0.3800
ensemble	100	0.5200	0.0900	0.1600	0.2300	0.7500	0.2500
ensemble	100	0.3100	0.2700	0.0600	0.3600	0.6700	0.3300
ensemble	100	0.4400	0.1600	0.0400	0.3600	0.8000	0.2000
ensemble	500	0.4480	0.1620	0.1540	0.2360	0.6840	0.3160
ensemble	500	0.3620	0.2440	0.0920	0.3020	0.6640	0.3360
ensemble	500	0.5240	0.0840	0.2220	0.1700	0.6940	0.3060
ensemble	500	0.4220	0.1820	0.1400	0.2560	0.6780	0.3220
ensemble	500	0.3800	0.2200	0.0660	0.3340	0.7140	0.2860
ensemble	500	0.5020	0.1060	0.2080	0.1840	0.6860	0.3140
ensemble	500	0.4560	0.1520	0.1600	0.2320	0.6880	0.3120
ensemble	500	0.3800	0.2260	0.0980	0.2960	0.6760	0.3240
ensemble	500	0.4480	0.1620	0.1560	0.2340	0.6820	0.3180
ensemble	500	0.4560	0.1480	0.1580	0.2380	0.6940	0.3060
ensemble	1000	0.5960	0.0060	0.3870	0.0110	0.6070	0.3930
ensemble	1000	0.4230	0.1820	0.1310	0.2640	0.6870	0.3130
ensemble	1000	0.3910	0.2050	0.1220	0.2820	0.6730	0.3270
ensemble	1000	0.4680	0.1330	0.1700	0.2290	0.6970	0.3030
ensemble	1000	0.4310	0.1790	0.1670	0.2230	0.6540	0.3460
ensemble	1000	0.4190	0.1900	0.1250	0.2660	0.6850	0.3150
ensemble	1000	0.3960	0.2070	0.1160	0.2810	0.6770	0.3230
ensemble	1000	0.4850	0.1190	0.2200	0.1760	0.6610	0.3390
ensemble	1000	0.4470	0.1570	0.1600	0.2360	0.6830	0.3170
ensemble	1000	0.4650	0.1420	0.2060	0.1870	0.6520	0.3480
ensemble	1500	0.4333	0.1747	0.1680	0.2240	0.6573	0.3427

Tabla 40: Anexo de matrices de confusión utilizados para construir las secciones 6.1 and 6.2.

Técnica	**Tam. muestra**	**0/0**	**0/1**	**1/0**	**1/1**	**Precisión**	**Error**
ensemble	1500	0.4647	0.1446	0.1587	0.2320	0.6967	0.3033
ensemble	1500	0.4820	0.1227	0.2140	0.1813	0.6633	0.3367
ensemble	1500	0.4867	0.1200	0.2160	0.1773	0.6640	0.3360
ensemble	1500	0.3880	0.2180	0.1100	0.2840	0.6720	0.3280
ensemble	2000	0.4280	0.1800	0.1405	0.2515	0.6795	0.3205
ensemble	2000	0.4665	0.1405	0.1980	0.1950	0.6615	0.3385
ensemble	2000	0.4700	0.1360	0.1875	0.2065	0.6765	0.3235
ensemble	2000	0.3940	0.2140	0.1165	0.2755	0.6695	0.3305
ensemble	2000	0.4865	0.1230	0.2170	0.1735	0.6600	0.3400

Tabla 41: Anexo de detalles de tiempos de ejecución para análisis de rendimiento. Clustering con $k = 5$ y dos medidas de similaridad.

Cores/ CPU	Tamaño muestra	Ejecuciones clustering	Preproces.	Cálculo de matrices de similaridad			Clustering			Ensamble de clustering			Total		
			Tiempo	Cálculos	Tiempo	Calc/seg	Cálculos	Tiempo	Calc/seg	Cálculos	Tiempo	Calc/seg	Cálculos	Tiempo	Calc/seg
1	100	40	6.8789	39996	62.99	634.96	24000000	2.71	8867174.16	16000000	15.98	1001359.28	40039996	88.55	452154.15
2	100	40	6.2008	39996	46.08	867.96	24000000	2.16	11107727.37	16000000	10.39	1540193.81	40039996	64.83	617610.89
4	100	40	5.8395	39996	33.07	1209.30	24000000	1.82	13178167.73	16000000	6.87	2328158.95	40039996	47.61	841058.53
8	100	40	5.8732	39996	31.28	1278.83	24000000	1.57	15325053.97	16000000	6.31	2536184.22	40039996	45.02	889313.09
12	100	40	5.0119	39996	32.34	1236.88	24000000	1.38	17365633.41	16000000	5.70	2808037.30	40039996	44.43	901233.02
1	500	40	8.5414	999996	763.89	1309.09	120000000	15.16	7914687.58	400000000	40.43	9893694.97	520999996	828.02	629210.53
2	500	40	6.6576	999996	576.67	1734.09	120000000	15.09	7951718.75	400000000	25.68	15573735.64	520999996	624.10	834799.73
4	500	40	6.6802	999996	402.38	2485.23	120000000	16.42	7309903.67	400000000	21.51	18594808.02	520999996	446.98	1165589.86
8	500	40	6.2924	999996	324.83	3078.56	120000000	14.48	8286097.97	400000000	18.02	22201723.86	520999996	363.62	1432826.64
12	500	40	5.9637	999996	328.30	3045.98	120000000	12.98	9244612.68	400000000	15.74	25417188.20	520999996	362.98	1435333.42
1	1000	40	9.4724	3999996	2950.81	1355.56	240000000	50.68	4735804.84	1600000000	95.89	16686498.74	1843999996	3106.85	593527.94
2	1000	40	7.4693	3999996	2073.18	1929.40	240000000	61.05	3931189.70	1600000000	58.05	27560883.67	1843999996	2199.75	838276.37
4	1000	40	7.1002	3999996	1423.90	2809.19	240000000	51.58	4652853.69	1600000000	47.73	33520120.71	1843999996	1530.31	1204984.66
8	1000	40	6.3357	3999996	1176.59	3399.64	240000000	56.61	4239498.38	1600000000	43.27	36980944.14	1843999996	1282.80	1437475.66
12	1000	40	6.5500	3999996	1253.43	3191.23	240000000	66.79	3593355.74	1600000000	43.70	36613957.67	1843999996	1370.47	1345520.76

Bibliografía

Abdi, Hervé y Williams, Lynne J (2010). «Tukey's honestly significant difference (HSD) test». *Encyclopedia of research design*, **3**, pp. 583–585.

Adomavicius, Gediminas y Tuzhilin, Alexander (2005). «Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions». *IEEE transactions on knowledge and data engineering*, **17(6)**, pp. 734–749.

Anuyah, Oghenemaro; Azpiazu, Ion Madrazo; McNeill, David y Pera, Maria Soledad (2017). «Can Readability Enhance Recommendations on Community Question Answering Sites?».

Armstrong, Jon Scott (2001). *Principles of forecasting: a handbook for researchers and practitioners*, tomo 30. Springer Science & Business Media.

Baeza-Yates, Ricardo; Ribeiro-Neto, Berthier et al. (1999). *Modern information retrieval*, tomo 463. ACM press New York.

Berners-Lee, Timothy J y Cailliau, Robert (1992). «World-wide web».

Bholowalia, Purnima y Kumar, Arvind (2014). «EBK-means: A clustering technique based on elbow method and k-means in WSN». *International Journal of Computer Applications*, **105(9)**.

Bojanowski, Piotr; Grave, Edouard; Joulin, Armand y Mikolov, Tomas (2017). «Enriching word vectors with subword information». *Transactions of the Association for Computational Linguistics*, **5**, pp. 135–146.

Bryant, Randal; Katz, Randy H y Lazowska, Edward D (2008). «Big-data computing: creating revolutionary breakthroughs in commerce, science and society».

Burke, Robin (2000). «Knowledge-based recommender systems». *Encyclopedia of library and information systems*, **69(Supplement 32)**, pp. 175–186.

Burke, Robin (2007). «Hybrid web recommender systems». En: *The adaptive web*, pp. 377–408. Springer.

Chen, Hsinchun (1995). «Machine learning for information retrieval: neural networks, symbolic learning, and genetic algorithms». *Journal of the American society for Information Science*, **46(3)**, pp. 194–216.

Chen, Pei-Yu; Chou, Yen-Chun y Kauffman, Robert J (2009). «Community-based recommender systems: Analyzing business models from a systems operator's perspective». En: *2009 42nd Hawaii International Conference on System Sciences*, pp. 1–10. IEEE.

Christopher, D Manning; Prabhakar, Raghavan y Hinrich, Schutza (2008). «Introduction to information retrieval». *An Introduction To Information Retrieval*, **151(177)**, p. 5.

Coffman, Kerry G y Odlyzko, Andrew M (1998). «The size and growth rate of the Internet». *First Monday*, **3(10)**, pp. l–25.

Condie, Tyson; Conway, Neil; Alvaro, Peter; Hellerstein, Joseph M; Elmeleegy, Khaled y Sears, Russell (2010). «MapReduce online.» En: *Nsdi*, tomo 10, p. 20.

Copeland, B Jack (2004). «Colossus: Its origins and originators». *IEEE Annals of the History of Computing*, **(4)**, pp. 38–45.

Cormen, Thomas H; Leiserson, Charles E; Rivest, Ronald L y Stein, Clifford (2009). *Introduction to algorithms*. MIT press.

Cox, Michael y Ellsworth, David (1997). «Application-controlled demand paging for out-of-core visualization». En: *Visualization'97., Proceedings*, pp. 235–244. IEEE.

De Battista, Anabella; Cristaldo, Patricia; Ramos, Lautaro; Nunez, Juan Pablo; Retamar, Soledad; Bouzenard, Daniel y Herrera, Norma Edith (2016). «Minería de datos aplicada a datos masivos». En: *XVIII Workshop de Investigadores en Ciencias de la Computación (WICC 2016, Entre Ríos, Argentina)*, .

Dean, Jeffrey y Ghemawat, Sanjay (2008). «MapReduce: simplified data processing on large clusters». *Communications of the ACM*, **51(1)**, pp. 107–113.

Devlin, Barry A. y Murphy, Paul T. (1988). «An architecture for a business and information system». *IBM systems Journal*, **27(1)**, pp. 60–80.

Eppler, Martin J y Mengis, Jeanne (2004). «The concept of information overload: A review of literature from organization science, accounting, marketing, MIS, and related disciplines». *The information society*, **20(5)**, pp. 325–344.

Everitt, Landau S., B. y Leese, M. (2001). *Cluster analysis*, tomo 4. London: Arnold.

Fred, Ana LN y Jain, Anil K (2005). «Combining multiple clusterings using evidence accumulation». *IEEE transactions on pattern analysis and machine intelligence*, **27(6)**, pp. 835–850.

Gandomi, Amir y Haider, Murtaza (2015). «Beyond the hype: Big data concepts, methods, and analytics». *International Journal of Information Management*, **35(2)**, pp. 137–144.

Goldberg, David; Nichols, David; Oki, Brian M y Terry, Douglas (1992). «Using collaborative filtering to weave an information tapestry». *Communications of the ACM*, **35(12)**, pp. 61–70.

Gomaa, Wael H y Fahmy, Aly A (2013). «A survey of text similarity approaches». *International Journal of Computer Applications*, **68(13)**, pp. 13–18.

Gonzalez, Alejandro; Flury, Carlos; Ferrari, Franco; Guereta, Guadalupe; Valoni, Mercedes; Diez, Santiago; Pera, Sole; Madrazo Azpiazu, Ion y Leale, Guillermo (2017). «Comparative Analysis on Text Distance Measures Applied to Community Question Answering Data». En: *5to Congreso Nacional de Ingeniería Informática / Sistemas de Información CONAIISI 2017, Santa Fe, Argentina*, tomo 1, pp. 99–106.

Guo, Gongde; Wang, Hui; Bell, David; Bi, Yaxin y Greer, Kieran (2003). «KNN model-based approach in classification». En: *OTM Confederated International Conferences."On the Move to Meaningful Internet Systems"*, pp. 986–996. Springer.

Harispe, Sébastien; Ranwez, Sylvie; Janaqi, Stefan y Montmain, Jacky (2015). «Semantic similarity from natural language and ontology analysis». *Synthesis Lectures on Human Language Technologies*, **8(1)**, pp. 1–254.

Hoch, Fred; Kerr, Michael; Griffith, Anne et al. (2001). «Software as a service: Strategic backgrounder». *Software & Information Industry Association (SIIA)*.

Huang, Anna (2008). «Similarity measures for text document clustering». En: *Proceedings of the sixth new zealand computer science research student conference (NZCSRSC2008), Christchurch, New Zealand*, pp. 49–56.

Jain, Anil K (2010). «Data clustering: 50 years beyond K-means». *Pattern recognition letters*, **31(8)**, pp. 651–666.

Janardhanan, PS y Samuel, Philip (2020). «Optimum Parallelism in Spark Framework on Hadoop YARN for Maximum Cluster Resource Utilization». En: *First International Conference on Sustainable Technologies for Computational Intelligence*, pp. 351–363. Springer.

Jeon, Jiwoon; Croft, W Bruce; Lee, Joon Ho y Park, Soyeon (2006). «A framework to predict the quality of answers with non-textual features». En: *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 228–235. ACM.

Joulin, Armand; Grave, Edouard; Bojanowski, Piotr; Douze, Matthijs; Jégou, Hérve y Mikolov, Tomas (2016). «Fasttext. zip: Compressing text classification models». *arXiv preprint arXiv:1612.03651*.

Kaufman, Leonard y Rousseeuw, Peter J (2009). *Finding groups in data: an introduction to cluster analysis*, tomo 344. John Wiley & Sons.

Kodinariya, Trupti M y Makwana, Prashant R (2013). «Review on determining number of Cluster in K-Means Clustering». *International Journal*, **1(6)**, pp. 90–95.

Korenius, Tuomo; Laurikkala, Jorma y Juhola, Martti (2007). «On principal component analysis, cosine and Euclidean measures in information retrieval». *Information Sciences*, **177(22)**, pp. 4893–4905.

Laney, Doug (2001). «3D data management: Controlling data volume, velocity and variety». *META group research note*, **6(70)**, p. 1.

Launchbury, John (1993). «Lazy imperative programming». En: *Workshop on State in Programming Languages, Copenhagen, Denmark, ACM*, .

Leale, Guillermo; Milone, Diego H; Bayá, Ariel E; Granitto, Pablo Miguel y Stegmayer, Georgina (2013). «A novel clustering approach for biological data using a new distance based on Gene Ontology». En: *XIV Argentine Symposium on Artificial Intelligence (ASAI)-JAIIO 42 (2013).*, .

Leskovec, Jure; Rajaraman, Anand y Ullman, Jeffrey David (2014). *Mining of massive datasets*. Cambridge university press.

Li, Baichuan y King, Irwin (2010). «Routing questions to appropriate answerers in community question answering services». En: *Proceedings of the 19th ACM international conference on Information and knowledge management*, pp. 1585–1588. ACM.

Li, Yuhua; McLean, David; Bandar, Zuhair A; O'shea, James D y Crockett, Keeley (2006). «Sentence similarity based on semantic nets and corpus statistics». *IEEE transactions on knowledge and data engineering*, **18(8)**, pp. 1138–1150.

Lilien, GL; Kotler, P y Moorthy, KS (1992). «Marketing models Prentice-Hall». *Englewood Cliffs, NJ*.

Lin, Dekang et al. (1998). «An information-theoretic definition of similarity.» En: *Icml*, tomo 98, pp. 296–304. Citeseer.

Lops, Pasquale; De Gemmis, Marco y Semeraro, Giovanni (2011). «Content-based recommender systems: State of the art and trends». En: *Recommender systems handbook*, pp. 73–105. Springer.

MacQueen, James et al. (1967). «Some methods for classification and analysis of multivariate observations». En: *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, tomo 1, pp. 281–297. Oakland, CA, USA.

Maletic, Jonathan I y Marcus, Andrian (2000). «Data Cleansing: Beyond Integrity Analysis.» En: *Iq*, pp. 200–209. Citeseer.

Manyika, James; Chui, Michael; Brown, Brad; Bughin, Jacques; Dobbs, Richard; Roxburgh, Charles y Byers, Angela H (2011). «Big data: The next frontier for innovation, competition, and productivity».

McCormick, Chris (2016). «Word2vec tutorial-the skip-gram model».

Meng, Xiangrui; Bradley, Joseph; Yavuz, Burak; Sparks, Evan; Venkataraman, Shivaram; Liu, Davies; Freeman, Jeremy; Tsai, DB; Amde, Manish; Owen, Sean et al. (2016). «Mllib: Machine learning in apache spark». *The Journal of Machine Learning Research*, **17(1)**, pp. 1235–1241.

Mikolov, Tomas; Chen, Kai; Corrado, Greg y Dean, Jeffrey (2013). «Efficient estimation of word representations in vector space». *arXiv preprint arXiv:1301.3781*.

Miller, George A (1995). «WordNet: a lexical database for English». *Communications of the ACM*, **38(11)**, pp. 39–41.

Minnaar, Alex (2015a). «Word2Vec Tutorial Part I: The Skip-Gram Model.».

Minnaar, Alex (2015b). «Word2Vec Tutorial Part II: The Continuous Bag-of-Words Model.».

Mitchell, RS; Michalski, JG y Carbonell, TM (2013). *An Artificial Intelligence Approach*. Springer.

Morin, Frederic y Bengio, Yoshua (2005). «Hierarchical probabilistic neural network language model.» En: *Aistats*, tomo 5, pp. 246–252. Citeseer.

Morris, Robert JT y Truskowski, Brian J (2003). «The evolution of storage systems». *IBM systems Journal*, **42(2)**, pp. 205–217.

Murthi, BPS y Sarkar, Sumit (2003). «The role of the management sciences in research on personalization». *Management Science*, **49(10)**, pp. 1344–1362.

Peña, Daniel (2013). *Análisis de datos multivariantes*. McGraw-Hill España.

Pokorny, Jaroslav (2013). «NoSQL databases: a step to database scalability in web environment». *International Journal of Web Information Systems*.

Powell, Michael James David (1981). *Approximation theory and methods*. Cambridge university press.

Provost, Foster y Kohavi, R (1998). «Glossary of terms». *Journal of Machine Learning*, **30(2-3)**, pp. 271–274.

Ramos, Juan et al. (2003). «Using tf-idf to determine word relevance in document queries». En: *Proceedings of the first instructional conference on machine learning*, tomo 242, pp. 133–142.

RDUSSEEUN, Leonard KAUFMAN Peter J (1987). «Clustering by means of medoids».

Resnick, Paul y Varian, Hal R (1997). «Recommender systems». *Communications of the ACM*, **40(3)**, pp. 56–58.

Resnik, Philip (1995). «Using information content to evaluate semantic similarity in a taxonomy». *arXiv preprint cmp-lg/9511007*.

Ribadas, Francisco Jose; Vilares, Manuel y Vilares, Jesus (2005). «Semantic similarity between sentences through approximate tree matching». En: *Iberian Conference on Pattern Recognition and Image Analysis*, pp. 638–646. Springer.

Ricci, Francesco; Rokach, Lior y Shapira, Bracha (2011). «Introduction to recommender systems handbook». En: *Recommender systems handbook*, pp. 1–35. Springer.

Rich, Elaine (1979). «User modeling via stereotypes». *Cognitive science*, **3(4)**, pp. 329–354.

Robertson, Stephen (2004). «Understanding inverse document frequency: on theoretical arguments for IDF». *Journal of documentation*, **60(5)**, pp. 503–520.

Salton, G y McGill, M J (1983). «Introduction to modern information retrieval». *International Student Edition*.

Salton, Gerard (1989). «Automatic text processing: The transformation, analysis, and retrieval of». *Reading: Addison-Wesley*.

Schafer, J Ben; Frankowski, Dan; Herlocker, Jon y Sen, Shilad (2007). «Collaborative filtering recommender systems». En: *The adaptive web*, pp. 291–324. Springer.

Schütze, Hinrich; Manning, Christopher D y Raghavan, Prabhakar (2008). *Introduction to information retrieval*, tomo 39. Cambridge University Press.

Shardanand, Upendra y Maes, Pattie (1995). «Social information filtering: algorithms for automating "word of mouth"». En: *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 210–217.

Singhal, Amit et al. (2001). «Modern information retrieval: A brief overview». *IEEE Data Eng. Bull.*, **24(4)**, pp. 35–43.

Sinha, Rashmi R; Swearingen, Kirsten et al. (2001). «Comparing recommendations made by online systems and friends.» En: *DELOS*, .

Sparck Jones, Karen (1972). «A statistical interpretation of term specificity and its application in retrieval». *Journal of documentation*, **28(1)**, pp. 11–21.

Strehl, A y Chosh, J (2002). «Knowledge reuse framework for combining multiple partitions». *Journal of Machine learning Research*, **33(3)**, pp. 583–617.

Tabachnick, Barbara G y Fidell, Linda S (2007). *Experimental designs using ANOVA*. Thomson/Brooks/Cole Belmont, CA.

Wang, Yuanyuan; Chan, Stephen Chi-Fai y Ngai, Grace (2012). «Applicability of demographic recommender system to tourist attractions: A case study on trip advisor». En: *2012 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, tomo 3, pp. 97–101. IEEE.

Xu, Rui y Wunsch, Don (2008). *Clustering*, tomo 10. John Wiley & Sons.

Yang, Liu; Qiu, Minghui; Gottipati, Swapna; Zhu, Feida; Jiang, Jing; Sun, Huiping y Chen, Zhong (2013). «Cqarank: jointly model topics and expertise in community question answering». En: *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pp. 99–108. ACM.

Zhang, Jiaxing; Zhou, Hucheng; Chen, Rishan; Fan, Xuepeng; Guo, Zhenyu; Lin, Haoxiang; Li, Jack Y; Lin, Wei; Zhou, Jingren y Zhou, Lidong (2012). «Optimizing data shuffling in data-parallel computation by understanding user-defined functions». En: *Presented as part of the 9th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 12)*, pp. 295–308.