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. Marzo 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 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

Resumen **3**

Agradecimientos **7**

Índice de Tablas **8**

Índice de Figuras **9**

Capitulo 1. Introducción **10**
1. Introducción . . . 10
1.1. Área temática . . . 10
1.2. Tema específico . . . 12
1.3. Objetivo general . . . 14
1.4. Objetivos específicos . . . 14

Capitulo 2. Fundamentación **15**
2. Fundamentación . . . 15
2.1. Motivación de la tesis . . . 15
2.2. Importancia científico-tecnológica . . . 17
2.3. Formación de recursos humanos . . . 18
2.4. Importancia socio-económica . . . 19

Capitulo 3. Marco teórico **20**
3. Marco teórico . . . 20
3.1. Sitios de CQA . . . 20
3.2. Sistemas de recomendación . . . 21
3.2.1. Contexto Histórico . . . 21
3.2.2. Funciones de un Sistema de Recomendación . . . 22
3.2.3. Técnicas de Recomendación . . . 22
3.2.3.1. Basados en contenido . . . 23
3.2.3.2. Filtrado Colaborativo . . . 23
3.2.3.3. Demográficos . . . 23
3.2.3.4. Basados en conocimiento . . . 24
3.2.3.5. Basados en comunidades . . . 24
3.2.3.6. Sistemas Híbridos . . . 25
3.3. Big Data . . . 25
3.3.1. Contexto Histórico . . . 25
3.3.2. Map-Reduce . . . 27
3.3.2.1. Arquitectura Hadoop . . . 28
3.3.2.2. Apache Spark . . . 29
3.4. Medidas de distancia de texto . . . 30
3.4.1. Conceptos básicos . . . 30

3.4.1.1. Information retrieval 30
3.4.1.2. Unidad de documento 31
3.4.1.3. Stopwords 31
3.4.1.4. Tokenización 31
3.4.1.5. Similaridad 32
3.4.1.6. Medidas de proximidad 34
3.4.1.7. Modelo de espacio vectorial 34
3.4.1.8. Distancia del coseno 35
3.4.2. Term Frequency 36
3.4.3. Term Frequency/Inverse Document Frequency . . 37
3.4.4. Word2Vec 38
3.4.4.1. Motivación de Word2Vec 39
3.4.4.2. Modelo Skip-Gram 40
3.4.4.3. Modelo Continuous Bag of Works (CBOW) 42
3.4.5. FastText 42
3.4.5.1. Modelo sub-palabra 43
3.4.6. Semantic Distance 44
3.4.6.1. El método 44
3.4.6.2. Similaridad semántica entre palabras . . 45
3.4.6.3. Similaridad semántica entre frases 46
3.4.6.4. Similaridad de orden entre frases 47
3.4.6.5. Similaridad total entre frases 48
3.5. Ensamble de Clustering 48
3.5.1. Clustering 48
3.5.1.1. Definición de Cluster 49
3.5.1.2. Algoritmos de Clustering 49
3.5.1.3. Particionamiento Alrededor de Medoids (PAM) 49
3.5.2. Ensamble de clustering 51
3.5.2.1. Definición formal 51
3.5.2.2. Acumulación de Evidencias 52

Capitulo 4. Problema de investigación y propuesta 54
4. Problema de investigación y propuesta 54
4.1. Hipótesis de trabajo 54
4.2. Metodología de investigación 54
4.3. Método propuesto 56
4.3.1. Generación de conjunto de particiones 56
4.3.2. Construcción de la matriz de co-asociación 57
4.4. Arquitectura de procesamiento de datos 58
4.4.0.1. Escalabilidad horizontal 60
4.4.0.2. Escalabilidad y complejidad temporal . . 60
4.5. Implementación en un sistema de recomendación de tiempo real 63
4.5.1. Arquitectura general 64
4.5.2. Procesamiento batch 65
4.5.3. Consulta de una pregunta existente 67
4.5.4. Agregar una nueva pregunta 68

4.3.5. Condiciones institucionales para el desarrollo de la tesis. Infraestructura y equipamiento 6 69

Capitulo 5. Experimentos **70**

5. Experimentos 70
5.1. Estado del arte 70
5.1.1. Análisis de rendimiento 70
5.1.2. Medidas de desempeño y error 73
5.2. Preprocesamiento del conjunto de datos 74
5.3. Muestreo del conjunto de datos 75
5.4. Generación de particiones 76
5.4.1. Cálculo de similaridades 76
5.4.2. Clustering y etiquetado 78
5.4.2.1. Entrada y configuración inicial 78
5.4.2.2. Proceso de clustering 79
5.4.2.3. Estructura de los resultados 80
5.5. Ensamble de clustering 80
5.6. Método de validación 83
5.6.1. Generación de conjuntos estadísticamente significativos 83
5.6.1.1. Elección de los tamaños de muestra . . . 84
5.6.1.2. Elección del número de clusters 84
5.6.2. Estructura de las matrices de confusión 85
5.6.2.1. Preparación de los datos 86
5.6.2.2. Construcción y elección del umbral correcto 87
5.6.2.3. Construcción de las matrices de confusión 89
6. Resultados 90
7. Conclusiones 91

Bibliografía **92**

Agradecimientos

Índice de Tablas

1. Matriz de coasociación salida del proceso EQuAL. 66
2. Ejemplo de un registro de la base de datos de similaridad entre preguntas. 66
3. Análisis de rendimiento de los algoritmos de similaridad del estado del arte. 73
4. Matrices de confusión para los cinco algoritmos de medidas de similaridad. 73
5. Ejemplo de la estructura de los subconjuntos de muestreo. 76
6. Ejemplo de la estructura de matriz triangular en formato de tabla. 77
7. Ejemplo de la estructura de matriz de similaridad en formato de tabla. 77
8. Ejemplo de la estructura del conjunto de preguntas individuales de la muestra en curso. 78
9. Ejemplo de la estructura del resultado de la ejecución del algoritmo de clustering. 80
10. Ejemplo de asignación de clusters a preguntas individuales para la ejecución 1. 81
11. Ejemplo de asignación de clusters a preguntas individuales para la ejecución 2. 81
12. Ejemplo de asignación de clusters a preguntas individuales para la ejecución 3. 81
13. Conjunto de datos de asignación de clusters agrupados por pregunta individual para generación de tuplas $(run\_id, cluster\_id)$ . . 82
14. Conjunto de datos intermedio que indica cuando dos preguntas coinciden en el mismo cluster/ejecución mediante un arreglo de tuplas. 82
15. Ejemplo de matriz de co-asociación salida del proceso de ensamble de clustering. 83
16. Matriz de confusión para validación de resultados. 86
17. Muestras de pares de preguntas que se utilizó como entrada del método EQuAL. 86
18. Matriz de co-asociación generada a partir de la muestra de la tabla 17. 87
19. Filtrado de la tabla 18 con los pares de preguntas que se encuentran en la tabla 17. 87
20. Asignación binaria de los resultados de similaridad obtenidos en la tabla 18, teniendo en cuenta un umbral de 0.65. 88
21. Filtrado de la tabla 20 con los pares de preguntas que se encuentran en la tabla 17. 88
22. Utilización de un umbral de 0.9 en la matriz de co-asociación. . . 88
23. Ejemplo de conjunto de datos de entrada (reales) para validación. 89

24. Ejemplo de conjunto de datos de predichos por el método EQuAL. 89
25. Resultado de comparación de las tablas 24 y 25 para validación y construcción de matrices de confusión. 90
26. Matriz de confusión obtenida a partir de la comparación de las tablas 23 y 24. 90

Índice de Figuras

1. Pipeline para un RS basado en contenido de CQA y en una nueva medida de similaridad. 12
2. La arquitectura CBOW predice la palabra actual basado en el contexto, mientras que la arquitectura Skip-gram predice palabras vecinas dada una palabra como entrada (Mikolov et al., 2013). . . 40
3. Diagrama de cálculo de similaridad semántica. 44
4. Base de datos semántica de forma jerárquica. 45
5. Método EQuAL para la generación de matrices de co-asociación desde el conjunto de datos original. 56
6. Infraestructura de la solución para generar una matriz de coasociación para un RS. 59
7. Aproximación de funciones de complejidad temporal según la cantidad de nodos del cluster de computadoras. 62
8. Arquitectura de un sistema de recomendación a tiempo real utilizando el método EQuAL. 64
9. Procesamiento batch de actualización de la base de datos de preguntas similares. 65
10. Flujo de consulta de una pregunta existente. 67
11. Flujo en el RS a tiempo real cuando se agrega una nueva pregunta al sistema. 68
12. Metodo del codo. 85

Capitulo 1

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 la misma ha sido formulada por otro usuario con anterioridad y que, además, esta pregunta tenga la respuesta buscada. Gracias a estos mecanismos, el usuario podría leer las respuestas de dicha pregunta sin tener que esperar a que la misma sea respondida. Esto no siempre es una tarea fácil, ya que podría darse la situación en la que esta pregunta exista previamente en el sitio y haya sido respondida, pero puede estar formulada de una manera completamente diferente en el sentido léxico. Por esta razón, una correspondencia exacta (o casi exacta) no es aplicable. Consideramos el siguiente ejemplo de dos preguntas iguales: *¿Cómo elijo una revista para publicar mi artículo?* y *¿Dónde publico mi*

[1] Yahoo! Answers: `https://answers.yahoo.com/`. Último acceso: Octubre 2020.
[2] Stackexchange: `https://stackexchange.com/`. Último acceso: Octubre 2020.
[3] Stackoverflow: `https://stackoverflow.com/`. Último acceso: Octubre 2020.
[4] Quora: `https://www.quora.com/`. Último acceso: Octubre 2020.

artículo?[5] . Entre estas dos frases, existe apenas una superposición de palabras, sin tener en cuenta *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 una medida de similaridad que se puede intuir como máxima cuando son idénticas y que es inversamente proporcional a las diferencias entre ellas (Lin et al., 1998). 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 del análisis anterior 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 que los usuarios pueden querer utilizar (Ricci et al., 2011). Las sugerencias relacionan varios 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 de "*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 y, 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 comportamientos del usuario o en grupos de usuarios para encontrar sugerencias adecuadas a sus preferencias; las segundas, por su parte, son inherentes a los ítems que el RS sugerirá. Cada una de estas estrategias de recomendación se elabora a partir de diferentes conocimientos y datos recopilados por el sitio o sistema donde el RS esté aplicado. Ejemplos de tales aplicaciones incluyen la recomendación de libros, películas o ítems de compra (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 personaliza-

[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://data.quora.com/First-Quora-Dataset-Release-Question-Pairs`. Último acceso: Agosto 2018.

[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).

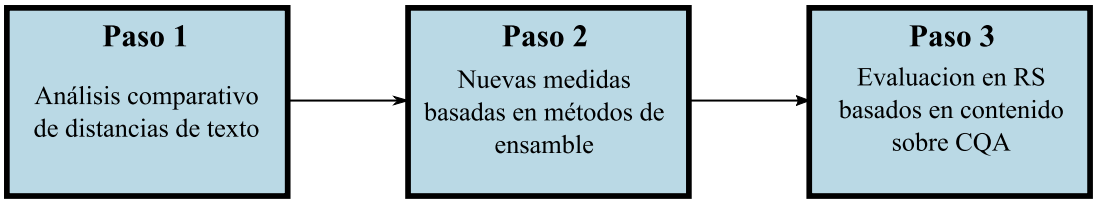

Figura 1: Pipeline para un RS basado en contenido de CQA y en una nueva medida de similaridad.

dos, 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 propondrá una 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.

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”, el cual se centra en el Paso 1 del 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, obtenidas de análisis de texto, con el fin de evaluar RS a partir de grandes conjuntos de datos; en el Paso 2, a partir de los resultados arrojados por el Paso 1, se crea una nueva medida construyendo una matriz de similaridad basada en

análisis de clustering, como una de las propuestas en los métodos *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 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 distancias obtenida en el Paso 2 en un RS basado en contenido, con el fin de evaluar su eficacia en sitios de CQA.

Para tomar dimensión del volumen de datos que es necesario manejar con este enfoque basado en clustering, si, por ejemplo, tomáramos el conjunto de datos Quora (404301 pares de preguntas, es decir, 808602 preguntas totales), y quisiéramos generar una sola matriz de distancias cruzando absolutamente 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. Esta matriz considera sólo una de las distancias de similaridad de texto del estado del arte, por lo cual si deseamos combinar varias medidas mediante un método de ensamble, deberíamos generar al menos una matriz por cada distancia del estado del arte, y luego usar las mismas para aplicar EAC, por lo cual estaríamos generando un número total de cálculos considerablemente mayor al que puede procesar una computadora clásica. 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. Esto conlleva considerar una arquitectura Big Data, optimización de código y técnicas de ejecución paralela entre gran número de núcleos 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, por lo cual, es necesario un esquema de almacenamiento optimizado para la implementación de este método. Con respecto a la velocidad de procesamiento, pruebas preliminares en 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, apuntará entonces a construir una medida de similaridad novedosa desde un enfoque Big Data, tal como se describe en el Paso 2 del pipeline, por lo cual es necesario crear un nuevo software basado en una arquitectura y patrones de Big Data, tomando como punto de partida el desarrollo del estado de arte.

1.3. Objetivo general

El presente trabajo de investigación tiene como objetivo construir una arquitectura Big Data que se aplique a grandes conjuntos de datos de preguntas de CQA y permita encontrar 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.

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

2. 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.

3. 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 sea extensible a otras medidas

4. Brindar conclusiones, pautas y recomendaciones para trabajar con medidas de comparación de textos en grandes volúmenes de datos en sitios de CQA utilizando arquitecturas basadas en Big Data.

[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).

Capitulo 2

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 (González y otros, 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. FastText y Word2Vec están fuertemente afectados en el orden en el cual aparecen las palabras. Adicionalmente, ninguna de estas técnicas tiene en cuenta la semántica de las palabras y sus relaciones, como si lo hace Semantic Distance. Sin embargo, esta última técnica, según el trabajo tomado como estado del arte, tampoco alcanza medidas de rendimiento apropiadas para un RS en un sitio de CQA, como por ejemplo: buen desempeño debido a su complejidad inherente.

Resultados experimentales de medidas de rendimiento obtenidas en el trabajo que se toma como punto de partida de esta tesis, arrojan entre un 66 % y un 68 % de precisión y entre un 32 % y un 33.5 % de error usando cada uno de los algoritmos de recomendación descritos anteriormente. Estos valores son

considerados prometedores, ya que las medidas de rendimiento son consistentes en todos los algoritmos seleccionados, lo que denota que la complejidad inherente del conjunto de datos no afecta significativamente la performance de cada uno de ellos. Además, los resultados de prueba no varían significativamente con respecto a los resultados de validación. Dicho esto, la motivación de este trabajo de tesis, así como el de las futuras líneas de investigación, es la creación de un método novedoso que combine medidas de similaridad existentes que pueda aplicarse como entrada para un RS que pueda ser implementado en sitios de CQA. Adicionalmente, se propondrá una arquitectura de software que soporte el procesamiento del método propuesto de una forma eficiente y escalable. Para tal fin, se crearán matrices de distancias (o similaridad), usando cada una de las preguntas del conjunto de datos en estudio, para luego combinarlas usando métodos de ensamble de clustering, ya que, como existen cientos algoritmos de clustering, es difícil identificar un solo algoritmo que pueda manejar todos los tipos de forma y tamaños de cluster, e incluso, decidir qué algoritmo sería el mejor para un conjunto de datos en particular. Fred y Jain (2005) introducen el concepto de clustering de acumulación de evidencias, que mapea las particiones de datos individuales en un ensamble de clustering dentro de una nueva medida de similaridad entre patrones, sumarizando la estructura entre-patrón percibido de esos clusters. La partición de datos final es obtenida aplicando el método *single-linkage* a la nueva matriz de similaridad. El resultado de este método muestra que, la combinación de algoritmos de clustering "débiles" como el *k-means*, pueden conducir a la identificación de clusters subyacentes verdaderos con 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 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 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. El resultado será un 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), más afectará negativamente en el tiempo de 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 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 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 un nuevo método y una arquitectura que lo soporte, 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.

2.2. Importancia científico-tecnológica

Con respecto a los sitios de CQA en particular, la importancia de este trabajo radica tanto en la posibilidad de construir un RS que, desde el punto de vista del usuario, reduzca tanto el tiempo promedio en que se encuentra una respuesta como, a su vez, mejore 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 evitará 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 de la satisfacción y fidelidad del usuario (Ricci et al., 2011).

Por último, 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, 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 congresos tales como AGRANDA[8] , CONAIISI[9] , o RecSys[10] ; o 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 y/o posgrado.
- 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.
- Desarrollar insumos para el armado de cursos tanto de formación académica como abiertos a la comunidad relacionados con Big Data o análisis de texto.

[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.

2.4. Importancia socio-económica

El tema posee una importancia social y económica que permitiría construir contactos y alianzas -económicas, académicas y de naturaleza mixta- con instituciones extranjeras. En otras palabras, a nivel social podría utilizarse para actividades de investigación y en los diferentes niveles educativos, según se adecúen las explicaciones y el vocabulario utilizado, las actividades y los diversos usos. Buscar 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 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. 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).

Capitulo 3

3. Marco teórico

En este capítulo, se explicarán los conceptos utilizados en el método propuesto que se desarrollará en el presente trabajo de tesis, estructurados en cinco grandes aristas: sitios de CQA, Sistemas de Recomendación, Big Data y medidas de similaridad y clustering. Todos estos conceptos se combinarán para luego, en el siguiente capítulo, desarrollar un método que satisfaga los objetivos del trabajo de tesis, de una manera superadora.

3.1. Sitios de CQA

Los servicios de Community Question Answering CQA, son un tipo especial de servicios *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 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 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, por ejemplo, un framework para predecir la calidad de las respuestas con características no textuales (Jeon et al., 2006), incorporar información de legibilidad en el proceso de recomendación (Anuyah et al., 2017), encontrar a los expertos apropiados (Li y King, 2010) o recomendar la mejor respuesta a una pregunta dada, entre otros. Sin embargo, el mecanismo existente en el cual se responden las preguntas en los sitios de CQA todavía no alcanza a satisfacer las expectativas de los usuarios por varias razones: (1) baja probabilidad de encontrar al experto: una nueva pregunta, en muchos casos,

[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.

puede no encontrar a la persona con la habilidad de responder de manera correcta, resultando en respuestas tardías y que distan de ser óptimas; (2) 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; (3) preguntas archivadas y poco consultadas: muchas preguntas de los usuarios son similares. Antes de formular una pregunta, un usuario podría beneficiarse de buscar ya formuladas, y por consiguiente, sus respuestas (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 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 de correo electrónico destinado a resolver el problema de manejar grandes cantidades de emails filtrando según cuán interesantes son los documentos, utilizando un enfoque basado en el contenido de los mismos y también filtros colaborativos, lo que después se denominaría RS no personalizados y personalizados por Ricci, Rokach y Shapira en el año 2011. Se ha trabajado mucho en mejorar y desarrollar nuevos enfoques con respecto a 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 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 predicciones (Armstrong, 2001), ciencias de la gestión

[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).

(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 items a un usuario, que posiblemente utilizará. Haciendo énfasis particularmente en 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 querer o necesitar. Esto implica, aumentar el ratio de conversión, es decir, la cantidad de ventas que es posible efectuar sobre un ítem sobre del total de oportunidades que un usuario selecciona el mismo. Por ejemplo, en un sitio de delivery de comida online, cuantas 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 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 usar 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; pudiendo utilizar este conocimiento para mejorar otros sistemas relacionados, como control de stock o 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 utilizad

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 RSs 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 son limitados a recomendar 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, la ventaja es que la información histórica no es necesaria. Por ejemplo, una aplicación podría ser 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 "personas-a-personas", como los sistemas colaborativos, pero utilizando distinta naturaleza de 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 usuarios e ítems a recomendar para generar la recomendación, razonando acerca de que items 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 RSs 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 estos RSs no es necesario recolectar información acerca de un usuario en particular porque las recomendaciones que realizan son exclusivamente basadas en las elecciones de un usuario en particular. 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, tienen 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

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*. Manyika et al. (2011) 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. Nótese que esta definición es agnóstica del tamaño del conjunto de datos, y no define un tamaño mínimo del mismo, sino que, asume que la tecnología avanza constantemente como así también las herramientas, por lo cual, la definición se "mueveçon 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 como "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 de Big Data.

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

El comienzo de sobrecarga de información, recientemente mencionado, data del año 1880, cuando el censo de los Estados Unidos tarda 8 años en 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 que él diseñó 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 implicaría el impacto más grande hasta la actualidad con respecto a la generación, identificación, almacenamiento y análisis de grandes volúmenes de datos de diversa naturaleza.

El inicio de los años 90' marcan 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, Cox y Ellsworth (1997) afirman que el crecimiento de la cantidad de datos que debe

[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.

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 realizan un gran porcentaje 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 el 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 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.,

[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.

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). En lenguajes de programación funcionales así como también lenguajes de alto nivel modernos, es posible escribir expresiones de estilo lambda, en las cuales es posible paralelizar y ejecutar programas en clusters distribuidos sin la necesidad de tener en cuenta los detalles de partición de datos y subprocesamiento. En este trabajo se utilizará el framework Apache Spark[22] ,

debido a su poder de procesamiento distribuido, su arquitectura basada en datos y su compatibilidad con las librerías necesarias en este trabajo.

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[23] (Sistema de Archivos Distribuido Hadoop). 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[24] , HSFS es altamente tolerante a fallos y es diseñado para ser ejecutado en computadoras de bajo costo. Además, HDFS provee una gran productividad accediendo a los datos de una aplicación y es posible usarlo en grandes conjuntos de datos.

HDFS tiene una arquitectura maestro-nodo[25] . 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

[22] Sitio web oficial de Apache Spark: `https://spark.apache.org/`. Último acceso: Febrero 2021.

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

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

[25] Del inglés, master-node architecture.

DataNodes. Por otro lado, el NameNode ejecuta operaciones tales como abrir, cerrar, y renombrar archivos y directorios.

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.

3.3.2.2. Apache Spark

Apache Spark es una plataforma de código abierto escrita en Scala[26] , 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[27] .

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 nodos (sin contar el nodo principal), 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, 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,

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

[27] 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.

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 las cuales 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 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) se define 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, 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 y 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. El estudio y el modelado en computadoras del proceso de aprendizaje en múltiples manifestaciones constituye 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 del IR y algunos del

ML. Técnicas probabilísticas y determinísticas son utilizadas para extraer información relevante desde diferentes orígenes de datos, así como también 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, tener 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 palabras que se filtran 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 manos según su contenido semántico relativo al dominio de los documentos que están siendo indexados.

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 y tipo. 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 es 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 (Xu y Wunsch, 2008). Las medidas de proximidad son 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.

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:

$$p(t) = \frac{cantidad\,de\,palabras\,asociadas\,a\,una\,definición\;\;sus\,hijos}{cantidad\,total\,de\,palabras\,en\,el\,corpus}$$

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 $p(t) = 0$ 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 contengan un contenido de información que, por lo contrario, tienda a cero.

Resnik (1995) define, en 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).

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, 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 conceptos abstractos (nodos ubicados en posiciones más cercanas al subsumidor mínimo) son más parecidos entre sí que dos 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 conceptos en este tipo de taxonomías: (I) La cantidad de información; y (II) La ubicación relativa entre los nodos hijos respecto al subsumidor mínimo. Definida como la similaridad de Resnik (1995).

La medida de Lin es:

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

El resultado de esta medida, estará normalizada en el rango $[0, 1]$ y obtiene que nodos más generales, con menor cantidad de información, son más similares entre sí, que dos nodos específicos (ya que la cantidad de información de los mismos

aumentará el denominador de la ecuación) para el mismo subsumidor mínimo. Se ha demostrado que esta definición de similaridad produce una correlación ligeramente mayor con los juicios humanos.

3.4.1.6. Medidas de proximidad

Proximidad es la 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 condiciones. Las condiciones mencionadas, son las utilizadas por 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 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, éste 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 positivo.

Para asignar un valor numérico a un documento en una consulta, 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.

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 número i del vector $\overrightarrow{Q}$ y $W_{t_{i\vec{D}}}$ es el valor de la componente número 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 puede ser la mayor frecuentemente aplicada 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 lo 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 $i \leq i, j \leq n$. Los símbolos $\overrightarrow{D_i}$ y $\overrightarrow{D_j}$ son documentos en forma de vectores y dces 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:

$$cos(\theta) = \frac{\vec{D}_i . \vec{D}_j}{\left\|\vec{D}_i\right\| . \left\|\vec{D}_j\right\|} = \frac{\sum_{i=1}^{n} d_{ii} d_{j_i}}{\sqrt{\sum_{i=1}^{n} {d_{ii}}^2}\sqrt{\sum_{i=1}^{n} {d_j}_i^2}}$$

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), *Term Frequency*, o TF; es una técnica donde el orden exacto de los términos es ignorado, pero se basa en número de ocurrencia de cada uno de ellos (Christopher et al., 2008).

Si comparamos las frases en inglés *"Mary is quicker than John"* y *"John is quicker than Mary"*, desde esta perspectiva, son idénticas. Esto conlleva 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 son contadas en una *bolsa*, lo cual difiere de la definición matemática de *conjunto*, que no considera los elementos repetidos. Cada palabra 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:

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

Hasta ahora, no se ha mencionado que implicancia tiene la frecuencia de cada término en un documento, en el momento de realizar la consulta: un documento que menciona un término más frecuentemente tiene más similitud con la query en cuestión, por lo tanto, recibirá una valoración mayor. Lo anterior, significa que se asigna un peso a cada término, en una relación directamente proporcional al número de ocurrencias del mismo en el documento (Christopher et al., 2008).

3.4.3. Term Frequency/Inverse Document Frequency

La valoración de un término, no tiene una relación directamente proporcional a su frecuencia en un documento, como ocurre con Term Frequency. En este caso, *inverse document frequency*, o IDF; está 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 (una de las posibles variantes) con la medida TF (también una de las posibles variantes, no la suma total). Asumiendo que existen N documentos en una colección, y un término t_i aparece en n_i documentos, 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. Estas palabras, que son comunes solo 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 valor final, considerando 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 las palabras con alto TF-IDF implica que son importantes en n_j pero no son comunes en la totalidad de los documentos. Este término t_i se dice que 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. 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 con-

juntos 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 uno 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 o unidades atómicas y no tienen noción de similaridad 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 vá 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 cual, el objetivo es maximizar la precisión de estas operaciones entre vectores mediante un arquitecturas de modelos que preserven 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.

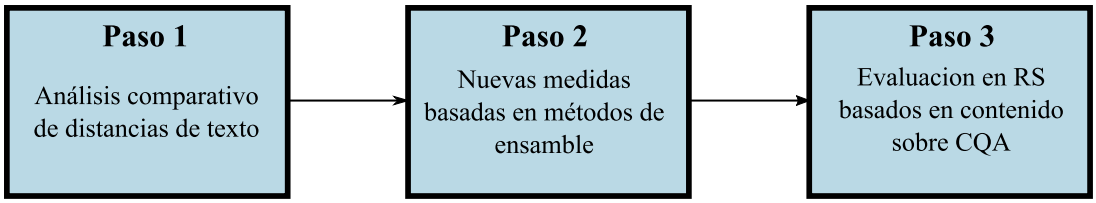

Figura 2: La arquitectura CBOW predice la palabra actual basado en el contexto, mientras que la arquitectura Skip-gram predice palabras vecinas dada una palabra como entrada (Mikolov et al., 2013).

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 vectores son resultados de propagación hacia atrás y *descenso de gradiente estocástico* (SDG por sus siglas en inglés de Stochastic Gradient Descent).

3.4.4.2. Modelo Skip-Gram

El modelo Skip-gram se basa en lo siguiente: dado un conjunto de frases (llamado *corpus*) el modelo analiza las palabras de cada sentencia y utiliza cada palabras para predecir las palabras que serán vecinas. 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 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*, 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. Dado 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^{th} representa los pesos correspondientes a la palabra i^{th} del vocabulario. Esta matriz W contiene entonces, la codificación como vector de cada una de las palabras del vocabulario (como filas), las cuales serán entrenadas mediante la red neuronal. 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, 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, 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.

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

3.4.4.3. Modelo Continuous Bag of Works (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 hde 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^{th} columna de la matriz W'. Y 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.

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

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 $n-gramas$. Y a cada uno de los n-gramas, se le asignará un vector, que luego, sumados, representarán

[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.

cada palabra. Dos características muy importantes son, (I) el método el muy rápido, permitiendo entrenar grandes conjuntos de datos; y (II) permite calcular representación de palabras que no aparecen en el conjunto de entrenamiento.

Repasando el modelo skip-gram, introducido por Mikolov et al. (2013), dado un vocabulario W, donde una palabra es identificada por su índice $w \in \{1, ..., W\}$, el objetivos es aprender la representación vectorial de cada palabra w. Dado un documento grande representado como una secuencia de palabras $w1, ..., wt$, el objetivo del modelo skip-gram es que las representaciones de la palabras sean entrenadas para predecir bien, otras palabras que aparecen en el contexto, es decir, minimizar la siguiente probabilidad logarítmica:

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

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 wes 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 conjunto de caracteres n-gram. Por ejemplo, la palabra where y con un $n = 3$ (longitud de la sub-palabras, en este caso sería, tri-gramas), obtenemos:

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

Dado un diccionario de n-gramas de tamaño G y una palabra w, el set de n-gramas que aparecen en wes $G_w \in \{1, ..., G\}$. 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:

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

31

[31] 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.

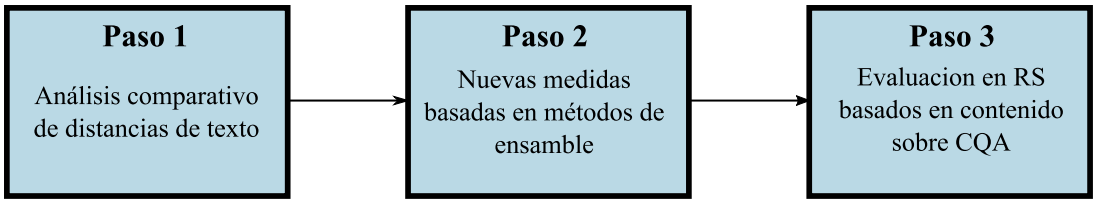

Figura 3: Diagrama de cálculo de similaridad semántica.

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

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 muy 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. El uso de una base de datos léxica posibilita modelar el sentido común humano y las estadísticas de corpus 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 compartidas funcionan bien solo 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 3, 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, un vector semántico sin procesar es derivado, utilizando la base de datos léxica. También, un vector de orden de palabras es generado a partir de cada frase utilizando la base de datos. Como

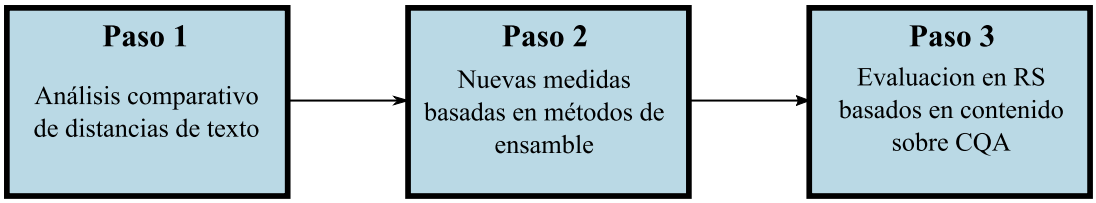

Figura 4: Base de datos semántica de forma jerárquica.

cada palabra en 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 los cuales 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. Por ejemplo, si tomamos la estructura detallada en la Figura 4, el camino más corto para conectar *boy* y *girl* sería *boy-male-person-female-girl*, y tiene una longitud de 4. El synset de personas 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.

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

También, se puede ver que la distancia entre *boy* y *teacher* es 6, mientras la distancia entre *boy* y *animal* es 4, lo cual es contraintuitivo 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. 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. 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, una similaridad semántica es calculada 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, ubicadas en orden inverso. Como las dos frases contienen las mismas palabras, un método basado en Bag of Words, resultará en que las frases son idénticas, pero para un intérprete humano, es claro que las frases sólo son similares en cierto grado. 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 una medida de similaridad de orden entre frases como:

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

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 $\delta \leq 1$ 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 objeto 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, sub-conjuntos o categorías). Everitt y Leese (2001) lo define como "Un cluster 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 medidas de proximidad.

3.5.1.2. Algoritmos de Clustering

Los algoritmos de clustering están generalmente clasificados en *clustering particional* y *clustering jerárquico*, basándose en la la forma que se generan los clusters (Xu y Wunsch, 2008). Los algoritmos de clustering jerárquico encuentran clusters anidados recursivamente en un modo aglomerativo (empezando con cada uno de los puntos de datos como si fueran clusters y combinandolos con el más similar, sucesivamente para formar una jerarquía de clusters) o en modo divisivo (de arriba hacia abajo - en inglés, top-down -) (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: una *partición*, no existen subconjuntos solapados y la unión de los subconjuntos es 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 via clustering. Por otro lado, los algoritmos particionales puede usar una matriz $n \times d$, representando n objetos en un espacio d-dimensional o, 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 famoso y ampliamente utilizado *k-medias* (MacQueen et al., 1967), y también el *Particionamiento Alrededor de Medoids (PAM).*

3.5.1.3. Particionamiento Alrededor de Medoids (PAM)

Particionamiento Alrededor de Medoids (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 medioides, los cuales son puntos "representativos" de un cluster, y tienen la particularidad de estar más centricamente ubicados en el mismo (RDUSSEEUN, 1987). 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.
$$C(x_i, x_j) = max(d(x_j, m_1) - d(x_j, x_i), 0)$$

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 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. 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 lo contrario, 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 arbitrarias, tamaños y densidades.

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, $x_i \in \mathbb{R}^2$, tal que, cuando adopten una representación vectorial. 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 clusterings dentro de una misma partición de datos viendo cada uno de esos resultados como una evidencia independiente de la organización de datos (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 generados de forma distribuida. Además, los mismos pueden ser reusados, facilitando así un análisis de datos eficiente.

Combinar evidencias Para lidiar con diferentes particiones con diferente cantidad 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, pero 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 la matriz de co-asociación), realizando, intrínsecamente, una transformación no-lineal de la original a la nueva representación de datos.

Capitulo 4

4. Problema de investigación y propuesta

4.1. Hipótesis de trabajo

A partir del relevamiento del estado del arte se infiere que las medidas de rendimiento obtenidas en las entradas de RS podrían presentar márgenes de mejora en cuanto a la experiencia de usuario y a la eficiencia de la implementación en una arquitectura Big Data, especialmente teniendo en cuenta grandes volúmenes de datos de entrada.

Por tal motivo, y como respuesta a la hipótesis planteada, se presentará un desarrollo de un método basado en una arquitectura Big Data que pueda aplicarse a grandes conjuntos de datos con el fin de obtener un procedimiento eficiente y eficaz que aproveche las ventajas adimensionales y de variabilidad de datos inherentes del ensamble de clustering con el fin de validarlo y realizar un análisis comparativo con las medidas del estado del arte.

4.2. Metodología de investigación

Este trabajo comenzará con una búsqueda de material científico relacionado a RS en general, RS no personalizados basados en análisis de texto, su aplicación en sitios de CQA, un análisis de 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. Esto puede ser realizado mediante sitios o librerías digitales, tales como Google Scholar[34] , IEEExplore

[34] Google Scholar: `https://scholar.google.com.ar`. Último acceso Febrero 2021.

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

Definida la hipótesis correctamente y el plan de trabajo, se iniciará 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 importarán 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 que se inicie el desarrollo del proyecto, serán evaluadas distintas opciones de herramientas y entornos que se utilizarán, Esto incluye:

- Lenguajes de programación y librerías inherentes al mismo.
- Almacenes de datos, frameworks y proyectos de terceros que puedan 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 identificará y documentará la nueva solución de acuerdo con los requerimientos de la Maestría en Ingeniería en Sistemas de Información, 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 realizará un registro con los indicadores resultantes, se validará la propuesta, se explicitarán los resultados obtenidos y se elaborarán las conclusiones, a fin de abrir y/o profundizar en nuevas líneas de investigación.

[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`.

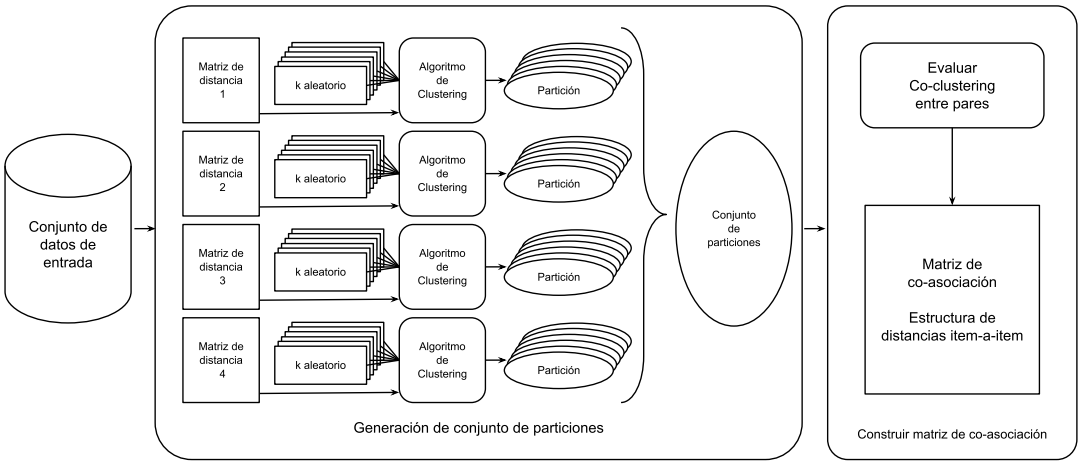

Figura 5: Método EQuAL para la generación de matrices de co-asociación desde el conjunto de datos original.

4.3. Método propuesto

Se propone el método EQuAL (*Ensemble method for community Question Answering sites based on cLustering*), que mejora la calidad y eficiencia para recomendar preguntas 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.

El desarrollo para este método está basado en dos pasos, como se muestra en la Figura 5: (I) la generación de conjunto de particiones y (II) construir matriz de co-asociación.

4.3.1. Generación de conjunto de particiones

El primer paso es la generación de un conjunto de particiones$X = \{x_1, x_2, ..., x_n\}$. Cada una de las particiones será una matriz de similaridad x_i. El procedimiento comenzará 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 tendrá 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 aplicarán c corridas de algoritmos de clustering, cada uno con un número k de elementos seleccionados al azar, que es un parámetro de entrada del algoritmo de clustering, que representa el número de clusters que se generarán. Esta combinación de n matrices y c ejecuciones

del proceso de clustering, resultará en $n \times c$ salidas del algoritmo de clustering elegido, es decir, particiones P como resultado. Cada partición Pcontará con la asignación de cada pregunta individual con un cluster específico.

Con el fin de resumir la estructura de cada una de las particiones generadas por los algoritmos de clustering, se combinan las particiones anteriormente obtenidas, dando lugar a un conjunto de particiones $\mathbb{P}$. Específicamente, si en este método utilizamos 5 algoritmos de similaridad distintos aplicados a procedimientos de clustering, obtendremos, un conjunto de 5 particiones:

$$\mathbb{P} = \{\mathrm{P}^1, \mathrm{P}^2, \mathrm{P}^3, \mathrm{P}^4, \mathrm{P}^5\};$$

siendo 5 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 combinará cada una de estas particiones, dando como salida una matriz de co-asociación, que contiene en cada posición 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, será la entrada para RS en sitios CQA. Además, tiene la característica de ser adimensional y comprende toda la variabilidad propia de los algoritmos de clustering, por lo cual, analiza la estructura de distancia item-item que es necesaria como 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 estadísticas, 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 variarán distintos

parámetros, tales como la medida de similaridad y valores de umbral involucrados en procesos de clustering, con el fin de obtener resultados confiables; lo cual redunda en múltiples ejecuciones de toda la solución. Debe destacarse que, en un primer momento, se implementarán experimentos basados en una infraestructura MapReduce aplicados con frameworks basados en Hadoop y cluster computing, desplegados en servidores elásticos en la nube, lo cual provee la ventaja de procesar grandes cantidades de datos en instancias dinámicamente escalables.

4.4. Arquitectura de procesamiento de datos

El procesamiento de datos se realizará con una infraestructura que permita el procesamiento distribuido en un cluster de computadoras, posibilitando que las operaciones de cómputo intensivo (ya sea calculo de distancia o ensambles de clustering) se realicen en distintos nodos al mismo tiempo. El conjunto de datos de entrada será convertido a una colección distribuida y particionada en P particiones de datos. Cada una de esas particiones será asignada a un nodo del cluster, que realizará el cálculo de distancia con un algoritmo determinado. Encontrar el número óptimo de particiones no es una tarea fácil, Apache Spark, por ejemplo, asigna automáticamente un número de particiones teniendo en cuenta la arquitectura del cluster y el tamaño de los archivos a procesar. Teniendo en cuenta un tamaño de archivo fija, y un cluster con 6 nodos de datos con 4 cores cada uno, Spark podría asignar $6 * 4 = 24$ particiones de datos, las cuales se procesarán en paralelo. Esto es algo relevante cuando hablamos de paralelismo en software, las tareas realizadas en distintos cores 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. El paralelismo al cual estamos acostumbrados, es 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.

Como se mencionó anteriormente, la generación de las matrices de distancia se realizarán de forma paralela, particionando el conjunto de datos de entrada y luego juntando los resultados para construir cada una de las matrices.

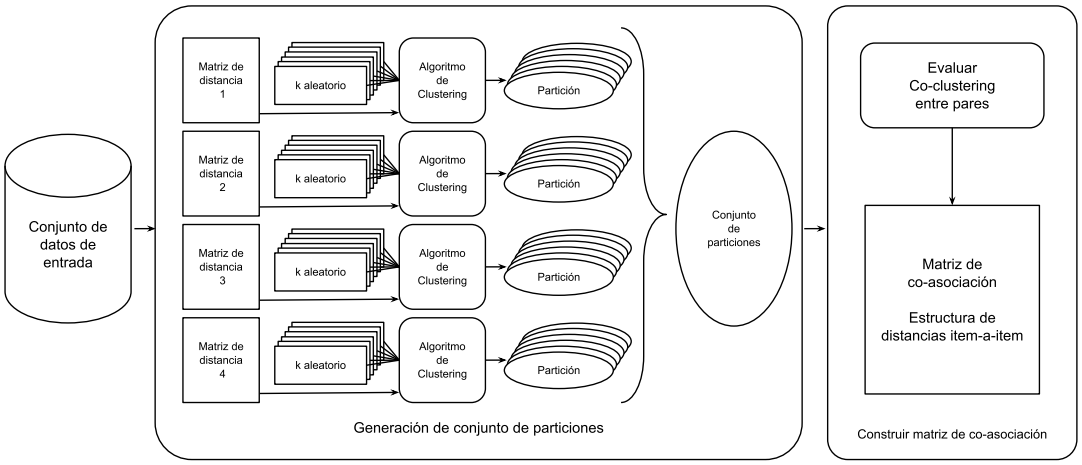

Figura 6: Infraestructura de la solución para generar una matriz de coasociación para un RS.

Una vez que las matrices de similaridad son generadas en la memoria del nodo maestro se procederá a realizar el algoritmo de cluster 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 la ejecuciones en todo el cluster de computadoras), son las siguientes:

- Apache Spark no provee la funcionalidad de "replicar" un conjunto de datos y enviar tareas a un ejecutor en particular, para luego replicar el resultado. Es decir, si quisiéramos llevar a cabo el algoritmo de clustering en un nodo de datos en particular utilizando la matriz de similaridad completa y, al mismo tiempo, duplicar esta tarea tantas veces como ejecutores existan, podría optimizarse la utilización de recursos y aprovechar el paralelismo en cada uno de ellos. Por el momento, no existe una forma fácil y computacionalmente eficiente de realizar esta tarea, en una arquitectura basada en Hadoop.

- Evitar *data-shuffling* excesivo entre 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, en cuando se realiza shuffling 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 producirá como resultados archivos persistidos en el almacenamiento local que posibiliten aplicar un paralelismo basado en datos en el cuando el ensamble de clustering es aplicado. El formato de salida de cada una de las ejecuciones del algoritmo de clustering aplicado, posibilitarán 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 matriz de coasociación.

4.4.0.1. 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, y no lo suficiente para justificar el costo. Por lo cual, la arquitectura elegida utiliza un cluster de computadoras 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 realizarlo fácilmente sin un algoritmo y una arquitectura que se adapte y sean pensados desde su concepción para este tipo de funcionalidad, tales como MapReduce y Hadoop.

4.4.0.2. Escalabilidad y complejidad temporal

En cuanto a 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. Al calcular la matriz triangular con entrada p, tenemos:

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

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 $\theta(2n2 - n)$. Esto significa que al hacer cálculos de similaridad "todos contra todos" la complejidad temporal aumenta cuadráticamente, es decir $O(n2)$[40] [41] . Siendo estrictos, $\theta(2n2 - n)$. 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 de iteraciones, para un tiempo t para calcular la distancia entre dos preguntas, tenemos una complejidad de $O(n \cdot k \cdot i \cdot t)$, suponiendo que todas las iteraciones son realizadas y dejando de lado la etapa de optimización de medoides para el algoritmo PAM. Por otro lado, la arquitectura propone que el cálculo de distancia se ha realizado en una etapa anterior y las mismas son almacenadas en una estructura de datos en forma de *tabla hash*[42] en la cual es posible obtener la distancias de dos elementos haciendo una búsqueda indexada mediante una función hash precalculada, 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 $O(n \cdot k \cdot i)$, y para N ejecuciones, obtendremos $O(N \cdot n \cdot k \cdot i)$.

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 *group by*, una función que tiene una complejidad conocida de $O(x \cdot \log(x))$. 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 $x \cdot \log(x)$ es posible simplificar la complejidad temporal a solo $O(x^2)$, o también $O(N^2 \cdot n^2)$.

[40] 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)

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

[42] 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.

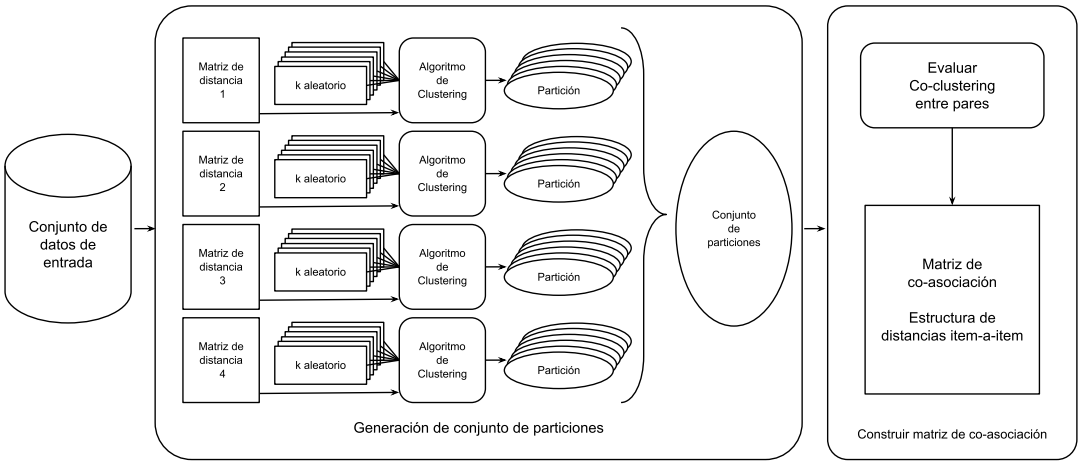

Figura 7: Aproximación de funciones de complejidad temporal según la cantidad de nodos del cluster de computadoras.

De forma simplificada, la complejidad total que ofrece el método propuesto es de $O(n^2 + Nnki + N^2n^2)$, 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, seria posible hacer que el valor de n (y por lo tanto la cantidad de cálculos) que trabaja cada nodo de ejecución y cada ejecutor en cada uno de ellos sea más pequeño, según la cantidad de nodos o servidores de ejecución. En un contexto ideal y como modelo simplificado, por ejemplo, si se cuenta con un cluster de computadoras de 8 nodos y cada uno de ellos procesa $n/8$ ($n_{nuevo} = n/número\_de\_nodos$) 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 reducen de forma cuadrática.

El ejemplo que muestra la Figura 7, es un modelo simplificado con fines didácticos que posee paralelismo de datos perfecto, cada uno de los nodos con la misma configuración y sin tener en cuenta tiempos de entrada/salida y re-particionamiento de datos. La unidad básica de Apache Spark es llamada0 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[43] , 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,

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

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 nodos, y 4 ejecutores en cada uno de ellos. 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. 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, persigue como objetivo el desarrollo de un nuevo método para implementar la matriz de distancias asociada al RS, a partir de un gran volumen de datos de entrada, en particular en CQA, y utilizando una arquitectura Big Data. Para tal fin, se propondrá un ejemplo de RS a tiempo real, que dará contexto al trabajo en cuestión. El ejemplo será desarrollado teniendo en cuenta 3 casos de uso diferentes: (I) un proceso batch 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 en que el usuario agrega una nueva pregunta al sitio.

El proceso completo del sistema de recomendación Un sistema de recomendación completo se puede analizar teniendo en cuenta 3 casos de uso, uno de los cuales es el planteado en 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 para perseguir tal fin, y utilizando la arquitectura Big data planteada. Este es el caso analizado en esta Tesis. El segundo caso de uso consiste en la consulta de una pregunta en el sitio. Este caso se nutre del caso de uso anterior, y utiliza sus resultados para obtener una lista de preguntas similares a la consultada. Por último, si bien el procesamiento batch tiene en cuenta todas las preguntas existentes, no posee la capacidad de recomendar preguntas simila-

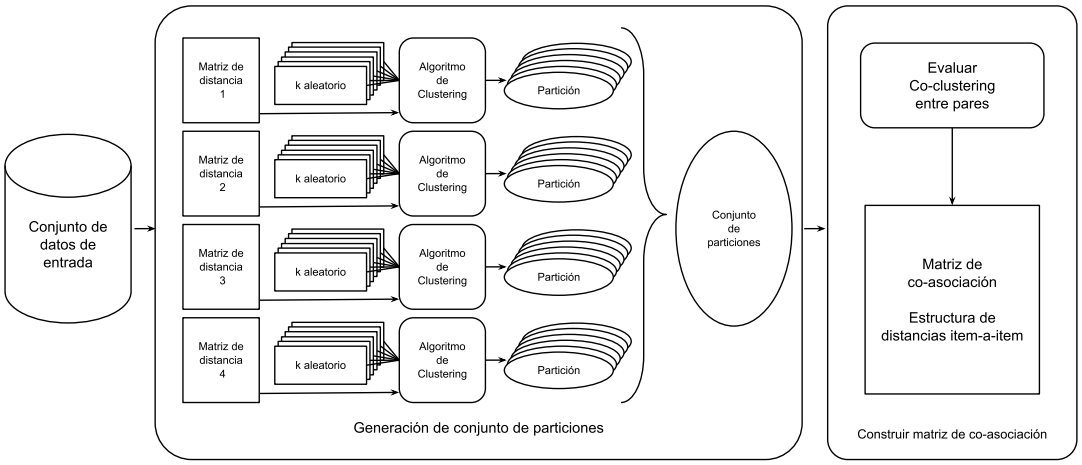

Figura 8: Arquitectura de un sistema de recomendación a tiempo real utilizando el método EQuAL.

res, en tiempo real, a una pregunta recién agregada al sistema. Por lo anterior, el tercer caso de uso se trata de una actualización, en la que un usuario agrega una nueva pregunta al sitio, y la resolución como implementación tecnológica para nutrir la base de datos original con nuevas preguntas y relaciones entre las mismas, en tiempo real.

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 el método propuesto en esta Tesis y su arquitectura subyacente.

4.5.1. Arquitectura general

La arquitectura del RS embebido en un sitio de CQA colaborativo, consiste básicamente de un microservicio dedicado a recomendación, un proceso de *streaming*[44] que realizará la función de recomendación a tiempo real de forma distribuida utilizando el método EQuAL, una base de datos altamente escalable NoSQL, y un proceso batch que generará una actualización de la misma de forma periódica.

La Figura 8 muestra cómo interactúa la totalidad de componentes relativos al RS, divididos fundamentalmente en procesamiento a tiempo real y batch; y además, muestra todas las interacciones posibles entre los componentes. Más

[44] En computación, un *stream* (en español, *flujo*) es una secuencia de elementos de datos disponibles durante un periodo de tiempo.

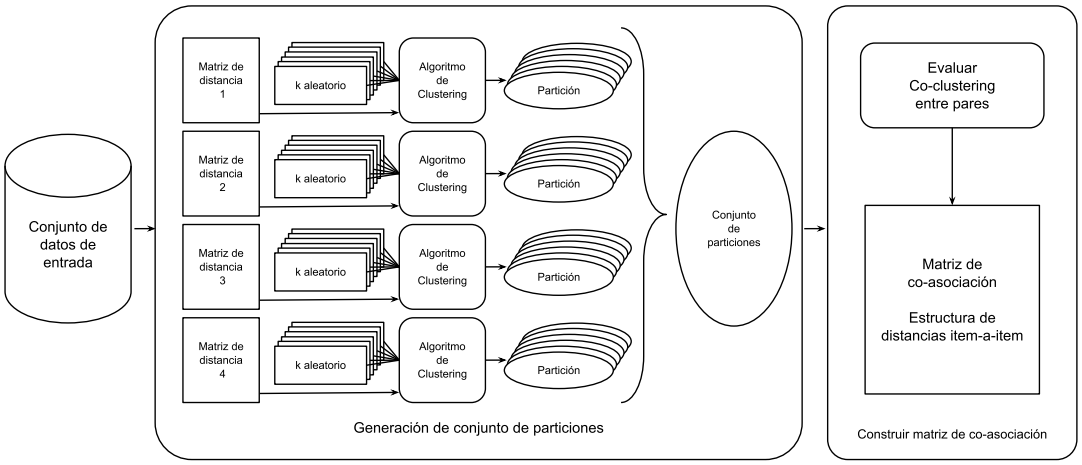

Figura 9: Procesamiento batch de actualización de la base de datos de preguntas similares.

adelante, se explicaran cada uno de los 3 casos de uso más importantes e identificables dentro de este sistema.

4.5.2. Procesamiento batch

El procesamiento batch para actualización y carga inicial de la base de datos que contendrá la matriz de coasociación en un formato adecuado para consulta, se ejecuta periódicamente pero con una carga inicial del conjunto de datos Quora en una base de datos. A continuación se detallaran cada uno de los casos.

Como punto de partida y en solo una oportunidad, se carga 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. La capacidad de seleccionar todos los registros de una tabla, tiene importancia para el proceso batch que utilizara el método EQuAL para la actualización periódica del RS.

De forma periódica, proceso batch en el cual se basa este trabajo, obtendrá todas las preguntas de la base de datos de preguntas individuales para crear la matriz de coasociación. La periodicidad podrá variar dependiendo del número de clusters elegidos para el método de clustering y la arquitectura en la cual se ejecuta, mientras más rápido sea, más frecuente podría ser programado. Podría ejecutarse diariamente, en momentos de bajo tráfico en el sitio en cuestión.

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

Tabla 1: Matriz de coasociación salida del proceso EQuAL.

question_id	similar_questions

1	[
	{
	"id": "question_id_4",
	"question": "question_desc_4",
	"similarity": "0.92"
	},
	{
	"id": "question_id_3",
	"question": "question_desc_3",
	"similarity": "0.67"
	},
	{
	"id": "question_id_2",
	"question": "question_desc_2",
	"similarity": "0.34"
	}
	]

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

Se agrega un proceso ETL[45] que tendrá como origen la matriz de coasociación y lo cargará en la base de datos que guardará la información de similaridad entre preguntas. Las características que debe tener esta base de datos será descrita en el apartado siguiente, pero la clave principal debe ser el ID de una pregunta individual para poder ser consultada fácilmente. El proceso ETL podrá ser lanzado mediante un evento que indique que la nueva matriz de coasociación ha sido generada, o bien podría ser una extensión del proceso EQuAL. Un ejemplo de la entrada y salida de este procesamiento puede ser el que se muestra en la **Tabla 1**. Luego del proceso ETL, los registros de la base de datos para el RS podrían tener la estructura que se muestra en la **Tabla 2**.

Esto permitirá consultar la pregunta mediante su ID de manera muy eficiente y veloz. Por otro lado, la carga de la tabla también será muy eficiente, y que la cantidad de registros será igual a la cantidad de preguntas individuales, en lugar de la cantidad de registros de la matriz de coasociación. La columna "similar_questions" será en formato JSON, la cual permite cargar datos con una estructura definida, y la posibilidad de serializar y deserializar los mismos de una forma

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

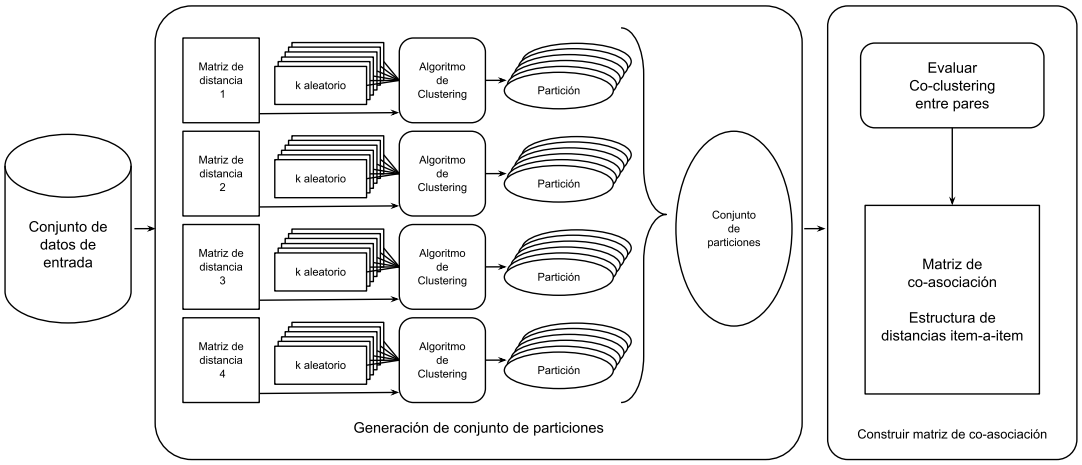

Figura 10: Flujo de consulta de una pregunta existente.

estándar.

4.5.3. Consulta de una pregunta existente

Para este apartado, se supone que la base de datos posee toda la información necesaria y está disponible, una aplicación web que consulta un servicio backend dedicado al sistema de recomendación mediante una API REST[46] . Este último consultará la base de datos para devolver las preguntas similares a un ID de pregunta dado.

En el momento que un usuario consulta una pregunta existente en el sitio, el servicio frontend enviará una solicitud GET al servicio backend, con el ID de la pregunta en cuestión. Este último buscará la pregunta en la base de datos utilizando el ID de pregunta y retornando todas las preguntas similares ordenadas por el valor de similaridad en forma decreciente. Esta operación debería ser ejecutada de manera rápida y eficiente, debido a un servicio backend dedicado y a una base de datos con las siguientes características:

1. 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.

2. Utilizando bases de datos como Apache Cassandra, la tabla 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

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

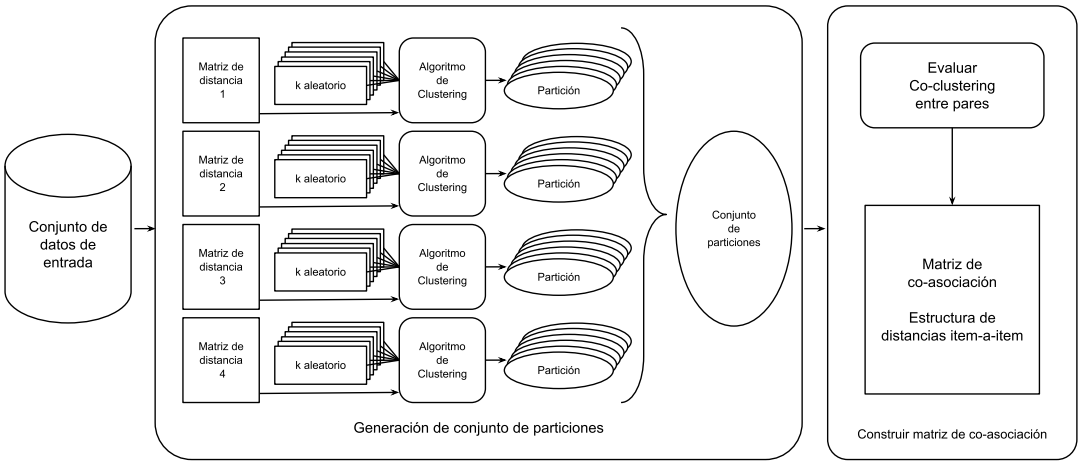

Figura 11: Flujo en el RS a tiempo real cuando se agrega una nueva pregunta al sistema.

en una función hash consistente y generada por el ID de pregunta. Esto es especial para búsquedas rápidas por PK.

3. Escalabilidad.

4. La información de preguntas similares para un ID dado, estará en formato JSON, haciendo que el servicio backend tenga un trabajo casi nulo para devolver la respuesta al frontend (una de las tareas puede ser que éste genere las URLs de las preguntas mediante los IDs).

5. Las bases de datos como Cassandra posibilitan replicación de los registros en distintos nodos de un cluster, garantizando alta disponibilidad.

4.5.4. Agregar una nueva pregunta

Agregar un nuevo item a un RS puede ser uno de los desafíos más grandes para los arquitectos de software. El propósito principal es el cálculo en tiempo real de los ítems recomendados para el ítem recién agregado al sistema, y que esto no signifique una sobrecarga del mismo. La arquitectura propuesta a continuación intenta que las preguntas similares estén disponibles, para una pregunta nueva recién agregada, con el fin de que el usuario que realiza una pregunta en el sitio, pueda ser recomendado con otras y, también, para que cualquier otro usuario pueda tener esa información disponible tan pronto como sea posible.

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[47] .

El candidato cursó la Maestría en Ingeniería en Sistemas de Información en dicha Facultad y tendrá a disposición 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 candidato utilizará equipos propios así como también los equipos de la universidad, en caso de que sea 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.

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

Capitulo 5

5. Experimentos

5.1. Estado del arte

El proyecto en el cual se realizaron los experimentos que este trabajo tiene como 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`.

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, 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. Análisis de rendimiento

Se muestra al análisis de rendimiento para el cálculo de distancias con los el proyecto del estado del arte, con el fin de tener un punto comparativo para los experimentos que se realizarán con la nueva arquitectura. Las pruebas de los algoritmos de similaridad 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
```

El cual también será utilizado para las pruebas de la nueva arquitectura. Se utilizará, como parámetro de rendimiento, el cálculo de similaridad para cada uno de los 404290 pares de preguntas. Luego de varias pruebas, se llega a una configuración de 8 hilos de ejecución y lotes de 10000 pares de preguntas, para obtener los tiempos más bajos posibles con cada uno de los algoritmos, los mismos fueron:

Bag of Words

```
[13:34:51] Starting script.
[13:34:51] Loading Quora questions file...
[13:34:52] ----- Run number 1 ------
[13:34:56] First 100000 distances calculated.
[13:34:58] First 200000 distances calculated.
[13:35:01] First 300000 distances calculated.
[13:35:03] First 400000 distances calculated.
[13:35:04] First 404290 distances calculated.
[13:35:04] Script finished. Total time: 0:00:12.895492
```

TF/IDF

```
[13:31:05] Starting script.
[13:31:05] Loading Quora questions file...
[13:31:07] ----- Run number 1 ------
[13:31:07] Generating a sample of 0 questions.
[13:31:08] Training model...
[13:32:43] First 100000 distances calculated.
[13:33:20] First 200000 distances calculated.
[13:33:58] First 300000 distances calculated.
[13:34:38] First 400000 distances calculated.
[13:34:39] First 404290 distances calculated.
[13:34:39] Script finished. Total time: 0:03:33.665420
```

FastText

```
[13:39:25] Starting script.
```

```
[13:39:25] Loading Quora questions file...
[13:39:26] ----- Run number 1 ------
[13:39:28] Training model...
Read 9M words
Number of words:  30823
Number of labels: 0
Progress: 100.0% words/sec/thread:  147553 lr:  0.000000 loss:  1.791601 ETA:
[13:40:29] First 100000 distances calculated.
[13:40:47] First 200000 distances calculated.
[13:41:06] First 300000 distances calculated.
[13:41:24] First 400000 distances calculated.
[13:41:25] First 404290 distances calculated.
[13:41:25] Script finished. Total time: 0:02:00.335008
```

Word2Vec

```
[13:42:29] Starting script.
[13:42:29] Loading Quora questions file...
[13:42:30] ----- Run number 1 ------
[13:42:34] First 100000 distances calculated.
[13:42:36] First 200000 distances calculated.
[13:42:38] First 300000 distances calculated.
[13:42:39] First 400000 distances calculated.
[13:42:40] First 404290 distances calculated.
[13:42:40] Script finished. Total time: 0:00:10.719605
```

Semantic Distance

```
[16:04:34] Starting script.
[16:04:34] Loading Quora questions file...
[16:04:35] ----- Run number 1 ------
[16:26:11] First 100000 distances calculated.
[16:47:20] First 200000 distances calculated.
[17:08:41] First 300000 distances calculated.
[17:30:08] First 400000 distances calculated.
[17:31:06] First 404290 distances calculated.
[17:31:06] Script finished. Total time: 1:26:32.941992
```

	Tiempo total (segundos)	Velocidad aproximada (calc/seg)
Bag of words	12.895492	31351.26601
TF/IDF	213.66542	1892.163926
FastText	120.335008	3359.703936
Word2Vec	10.719605	37715.00909
Semantic Distance	5192.941992	77.85374853

Tabla 3: Análisis de rendimiento de los algoritmos de similaridad del estado del arte.

			Predicho		Precisión	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		

Tabla 4: Matrices de confusión para los cinco algoritmos de medidas de similaridad.

Los resultados en **Tabla 3** muestran una clara ventaja en términos de rendimiento para el algoritmo de similaridad Word2Vec, inmediatamente seguido por Bag of Words. TF/IDF y FastText, muestran un rendimiento aceptable pero, en comparación, aproximadamente diez veces más lentos que los anteriormente mencionados. El rendimiento del algoritmo de similaridad Semantic Distance, es muy bajo y representa un gran problema al momento de analizar grandes conjuntos de datos, pero puede ser necesario utilizarlo debido a sus buenas medidas de desempeño y error y su análisis de similaridad desde una perspectiva distinta.

5.1.2. Medidas de desempeño y error

Utilizando cada para de preguntas del conjunto de datos original, se calculó la matriz de confusión para cada uno de los algoritmos de similaridad. Como se puede ver en la **Tabla 4**.

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, siendo los dos posibles 0, cuando las preguntas son distintas y 1 cuando las preguntas son iguales. Los resultados predichos, se muestran en las columnas. La

precisión obtenida de las matrices de confusión, que se calcula como la suma de los resultados acertados entre los valores predichos y reales, en todos los casos exceden el 66 %, alcanzando un máximo de 68 % para Word2Vec. Por otro lado, con respecto al error promedio, se llega a un valor máximo de 33 % en el caso de TF/IDF.

Observando las matrices de confusión, se presenta un desbalance en los dos valores tomados para calcular la precisión, siendo que los resultados obtenidos 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 $0,504–0,170 = 0,334$. 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 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 eliminarlos, también es crucial preprocesar los mismos con el fin de estandarizar algunos factores y así obtener mejores resultados.

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

1. Convertir el texto en minúscula; esto habilita a que la 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

```
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
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 la primera, es posible ver cómo se elimina la fórmula polinómica, mientras que en la segunda, todos los números fueron transformados a letras, con espacios entre ellos. En ambas, se puede ver que el signo de interrogación se elimina y todas las letras están en minúscula.

5.3. Muestreo del conjunto de datos

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

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

Tabla 5: Ejemplo de la estructura de los subconjuntos de muestreo.

Cada uno de los subconjuntos de muestreo tiene un criterio de aceptación, la proporción de pares de preguntas iguales (con indicador 1) debe 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 75 % 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 generan particiones desde los subconjuntos de muestreos generados desde el conjunto de datos original. Teniendo en cuenta que se va a utilizar un método de clustering para desarrollar el método de este trabajo, es necesario descomponer la estructura de los archivo de entrada, el cual contiene un identificador por par de preguntas, en preguntas individuales, con el fin de poder identificarlas unívocamente y poder utilizarlas como los objectos entrada del análisis de clustering. La estructura de los subconjuntos de muestreo se clarifican en la **Tabla 5**.

Luego, el conjunto el subconjunto de preguntas preparado para el cálculo de distancia será 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 identificará.

2. Se realiza una combinación de cada una de las filas contra la matriz en sí misma (eliminando resultados repetidos), generando como resultado una estructura de matriz triangular como se muestra en la Tabla 6.

La estructura de matriz triangular sirve de entrada para el cálculo de similaridad por cada una de las técnicas previamente mencionadas. Por lo cual, es

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

Tabla 6: Ejemplo de la estructura de matriz triangular 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

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

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, tal como se muestra en la Tabla 7.

Si se piensa el subconjunto de datos anterior como 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 posibilita crear particiones de datos de manera sencilla y procesarlas de forma distribuida. Por ejemplo, si tuviésemos un archivo CSV con 1,000,000 (un millón) de filas y un cluster con 8 nodos ejecutores, cada uno procesaría 125,000 filas, es decir, 1/8 del conjunto de datos total.

sequence_id	question
0	question_0
1	question_1
2	question_2
3	question_3

Tabla 8: Ejemplo de la estructura del conjunto de preguntas individuales de la muestra en curso.

5.4.2. Clustering y etiquetado

Una vez que las similaridades son calculadas en forma distribuida los resultados se almacenan y son colectados por el nodo maestro para realizar un etiquetado a partir de un análisis de clustering. Si bien, la implementación de un algoritmo de clustering completamente distribuido podría ser una mejora, sobre todo si se utiliza un algoritmo que requiere un gran costo computacional y es posible crear varias particiones de datos, se optó por un algoritmo relativamente simple que se ejecuta en un entorno multihilo en el nodo principal.

El algoritmo elegido es Clustering de Partición Alrededor de Medoids (PAM) como forma de "etiquetar" cada una de las preguntas en un determinado cluster, por cada una de las ejecuciones. Se implementó de la forma que se describe a continuación.

5.4.2.1. Entrada y configuración inicial

Como entrada es utilizada la matriz de similaridades que es resultado del paso anterior, y que servirán para obtener las distancias precalculadas entre cada uno de los elementos que participan en el proceso clustering. Además, en otro arreglo en memoria, el conjunto de preguntas individuales con un identificador secuencial, con el fin de facilitar el algoritmo, tal como se muestra en la **Tabla 8**.

Por cada una de las corridas de clustering (parametro de configuracion) se proporcionará un k inicial, que representa al número de clusters (o medoides) que definirá el algoritmo, y en los cuales las preguntas serán distribuidas (proceso de etiquetado); y además, son identificadas con un identificador único universal (UUID[48] por sus siglas en inglés) con el fin de poder recuperar cada uno de

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

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 un motivos de rendimiento y simplicidad, los experimentos son 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.

1. Se busca su similaridad con cada uno de los medoides.
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:

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 $(id\_pregunta, id\_pregunta\_medoide)$

run_id	question_id	assigned_medoid
638154671365754285511315930570649807 70	336	856
638154671365754285511315930570649807 70	342	856
638154671365754285511315930570649807 70	26	358
638154671365754285511315930570649807 70	1364	437

Tabla 9: Ejemplo de la estructura del resultado de la ejecución del algoritmo de clustering.

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 último conjunto de clusters generado el que se tomará 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 9**. Cada uno de los archivos almacena un UUID único (identico en cada una de las filas, para facilitar el algoritmo de ensamble), todas las preguntas individuales del muestreo en cuestión, 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. 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 estructura Spark, con la misma estructura que los archivos físicos, pero en memoria y en todos los nodos del cluster Hadoop, con el fin de poder realizar en ensamble de una forma eficiente y escalable.

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

Tabla 10: Ejemplo de asignación de clusters a preguntas individuales para la ejecución 1.

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 2.

El conjunto de etiquetas es agrupado por ID de pregunta, y aplicando una función de grupo que obtendrá una lista de tuplas $(run\_id, cluster\_id)$. 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 pueden ser representadas como archivos con formato CSV y procesada de manera distribuida. Cada uno de esos archivos generados para cada ejecución, contendrá información del ID 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 identificadores de ejecución *run_uuid*_1, *run_uuid*_2 y *run_uuid*_3, pueden representarse como se muestra en las **tablas 10, 11 y 12**.

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, representadas por la pregunta 3. La intuición para esta ejecución en particular, es que las preguntas 1, 2 y 3 son "más similares"

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

Tabla 12: Ejemplo de asignación de clusters a preguntas individuales para la ejecución 3.

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)]

Tabla 13: Conjunto de datos de asignación de clusters agrupados por pregunta individual para generación de tuplas $(run\_id, cluster\_id)$.

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	[]

Tabla 14: Conjunto de datos intermedio que indica cuando dos preguntas coinciden en el mismo cluster/ejecución mediante un arreglo de tuplas.

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 genere tuplas $(run\_id, cluster\_id)$ para el ejemplo anterior resultaría tal como se muestra en la **Tabla 13**.

Hasta ahora, este conjunto de datos muestra, por cada una de las preguntas, a qué cluster fue asignada, por cada ejecución del algoritmo. Este formato de representación, facilita el cálculo de la cantidad de veces que dos preguntas fueron asignadas al mismo cluster, para una misma ejecución. 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 $(run\_uuid\_1, 1)$ -. Para obtener el resultado de este cálculo, es necesario realizar una intersección de arreglos para cada una de las combinaciones de preguntas (eliminando las combinaciones en donde los ID de pregunta son iguales). Para este ejemplo, la intersección de arreglos se muestra en la **Tabla 14**.

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.

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

Tabla 15: Ejemplo de matriz de co-asociación salida del proceso de ensamble de clustering.

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 esta teoría, calculamos:

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

Siendo el numerador de la expresión, la cantidad de veces que dos preguntas coincidieron en el mismo cluster para una misma ejecución, y el denominador la cantidad total de ejecuciones, respondiendo a la fórmula:

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

Considerando el ejemplo en cuestión, el conjunto de datos resultado de aplicar la fórmula anterior se muestra en la **Tabla 15**. Obteniendo una matriz de co-asociación, que indica que las preguntas 1 y 3 coincidieron en el 100 % de las ejecuciones, mientras que las preguntas 1 y 2 solo en un tercio de las mismas, y las preguntas 1,4 nunca coincidieron.

5.6. Método de validación

5.6.1. Generación de conjuntos estadísticamente significativos

Para generar resultados estadísticamente significativos se ejecutara 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). Para cada tamaño de muestra, se realizaron 10 muestras aleatorias manteniendo un k fijo. Por ejemplo, para un número de clusters $k = 5$ y para un tamaño de

muestra 100, se realizaron 10 ejecuciones con un conjunto de datos de entrada aleatorio. Dando un total de $5 \times 10 = 50$ matrices de co-asociación resultado, para un k fijo.

5.6.1.1. Elección de los tamaños de muestra

El motivo de la elección del tamaño de muestra en el apartado anterior, se basa en que los experimentos en forma local, por la facilidad que éste brinda en 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 estandarizar los mismos a lo largo de todos los tamaños de muestras, es decir, no variar el número de clusters entre ellas, con fines de 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 serán validados en conjunto mediante el rendimiento de la matriz de co-asociación. Partiendo de la base del *método del codo* para la evaluación de la performance de un algoritmo de clustering, el cual evalúa el porcentaje de variabilidad explicada en

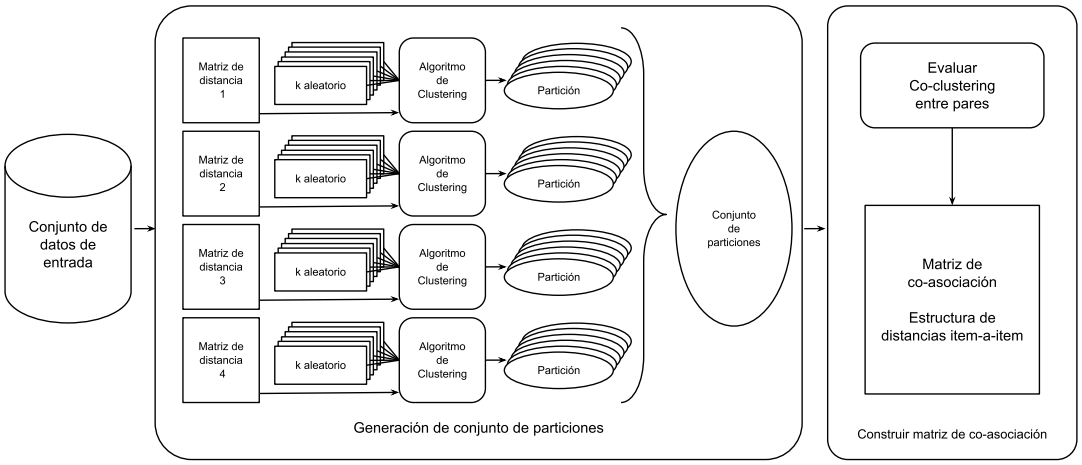

Figura 12: Metodo del codo.

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 agregaran una información considerable al modelo, pero en cierto punto la ganancia marginal caerá dramáticamente, dando un ángulo en el gráfico (Bholowalia y Kumar, 2014).

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 aplicará utilizando la matriz de coasociación generada a partir del ensamble. Para cuantificar y evaluar el performance, se utilizarán matrices de confusión denotarán 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.

5.6.2. Estructura de las matrices de confusión

Una matriz de confusión, es una matriz que muestra clasificaciones predichas y 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 16**.

Los indicadores de desempeño que se evaluarán en los experimentos realizados con fines comparativos, son los siguientes:

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

Tabla 16: Matriz de confusión para validación de resultados.

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

- **Exactitud:** $(a + d)/(a + b + c + d)$
- **Error:** $(b + c)/(a + b + c + d)$
- **Precisión positivos:** $d/(d + b)$
- **Precisión negativos:** $a/(a + c)$
- **Sensitividad:** $d/(c + d)$
- **Especificidad:** $a/(a + b)$

donde $a + b + c + d = 1$.

5.6.2.1. Preparación de los datos

Para evaluar el rendimiento del algoritmo de ensamble, se toma la matriz de coasociación generada como resultado del proceso total y la muestra de pares de preguntas que se utilizó como entrada para ese proceso. Por ejemplo, considerando la siguiente muestra de preguntas en la Tabla 17 y la matriz de co-asociación de la Tabla 18, se filtran solo los pares de preguntas de la matriz de coasociación que se encuentran en la muestra de pares de preguntas, ya que son las única con la cual su similaridad puede compararse con fines de validación, es decir, el conjunto de datos mostrado en la Tabla 19. Lo que nos deja con un conjunto de pares de preguntas que es posible comparar en su totalidad con la muestra original.

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.

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

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

Tabla 19: Filtrado de la tabla 18 con los pares de preguntas que se encuentran en la tabla 17.

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 ¿Cuando consideramos a esas preguntas iguales y cuando no? La respuesta es simple, 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 (1) y distintas si sucede lo contrario (0), clarificando:

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

La elección del mejor umbral, se realiza eligiendo valores en el intervalo $(0, 1)$ y evaluando cual de ellos conlleva a un mejor desempeño, 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 con intervalos 0,05 formando un arreglo como $[0{,}05, 0{,}1, 0{,}15, ..., 0{,}90, 0{,}95]$, por cada uno de ellos:

1. Se iteran todos los pares de preguntas.

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).

question_id_1	question_id_2	question_1	question_2	equal
question_10	question_11	contenido	contenido	1
question_10	question_20	contenido	contenido	0
question_10	question_21	contenido	contenido	0
question_11	question_20	contenido	contenido	0
question_11	question_21	contenido	contenido	0
question_20	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 0.65.

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 21: Filtrado de la tabla 20 con los pares de preguntas que se encuentran en la tabla 17.

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 *precisión* del método.

El valor de umbral que arroje la mayor precisión, 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 20** (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). Filtrando los pares de preguntas que se encuentran en la muestra de entrada, obtenemos el conjunto de datos de la **Tabla 21**, Lo cual muestra un resultado idéntico al conjunto de entrada. Lo anterior significa que los pares predichos son iguales a los pares predichos. 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 22**, 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.

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

Tabla 22: Utilización de un umbral de 0.9 en la matriz de co-asociación.

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

question_id_1	question_id_2	question_1	question_2	equal
question_10	question_20	contenido	contenido	1
question_11	question_21	contenido	contenido	1
question_12	question_22	contenido	contenido	0
question_13	question_23	contenido	contenido	1
question_14	question_24	contenido	contenido	1
question_15	question_25	contenido	contenido	0
question_16	question_26	contenido	contenido	0
question_17	question_27	contenido	contenido	1

Tabla 24: Ejemplo de conjunto de datos de predichos por el método EQuAL.

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. Cuanto 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 23, Y el resultado obtenido luego de la elección del mejor umbral, es la Tabla 24. Por lo cual, comparando los valores de la columna “equal’, obtendremos el resultado de la Tabla 25. Sumarizando los resultados, la matriz de confusión derivada se muestra en la Tabla 26. La precisión 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).

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

Tabla 25: Resultado de comparación de las tablas 24 y 25 para validación y construcción de matrices de confusión.

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

Tabla 26: Matriz de confusión obtenida a partir de la comparación de las tablas 23 y 24.

6. Resultados

7. Conclusiones

Bibliografía

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.

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.

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.