Universidad CENFOTEC

Maestría en Ingeniería del Software con énfasis en Inteligencia Artificial Aplicada

Documento final de Proyecto de Investigación Aplicada 2

Evaluación de un nuevo enfoque de operadores de relevancia normalizados en máquinas de búsqueda

José Alberto Aguilar Romero
Jean Carlo Álvarez Ramírez

Julio 2023

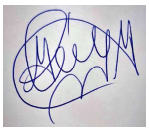
Declaratoria de derecho de autor

Se declara que el presente proyecto de investigación fue realizado por los autores José Alberto Aguilar Romero y Jean Carlo Álvarez Ramírez, fundamentando los diferentes capítulos del trabajo en diferentes fuentes bibliográficas, literatura citada, las cuales tienen su respectiva referencia, respetando los derechos de autor de dichos trabajos.

Se autoriza la reproducción total o parcial de este trabajo, para ser usados como referencia de trabajos futuros de tipo académico y científico, en este caso, se solicita incorporar la referencia de este trabajo respetando los derechos de los autores.

TRIBUNAL EXAMINADOR

Este proyecto fue aprobado por el Tribunal Examinador de la carrera: **Maestría Profesional en Ingeniería del Software con énfasis en Inteligencia Artificial Aplicada,** requisito para optar por el título de grado de **Maestría,** para los estudiantes: **Aguilar Romero José Alberto y Álvarez Ramírez Jean Carlo.**



Digitally signed by MIGUEL PEREZ MONTERO (FIRMA)
Date: 2023.07.12 10:22:17 -06'00'

M.Sc. Miguel Pérez Montero
Tutor

Dr. Luis Carlos Naranjo Zeledón
Lector 1

IGNACIO TREJOS ZELAYA (FIRMA)

Firmado digitalmente por IGNACIO TREJOS ZELAYA (FIRMA)
Fecha: 2023.07.12 16:04:13 -06'00'

M.Sc. Ignacio Trejos Zelaya
Lector 2

Es copia fiel del original firmado digitalmente, el cual debe ser guardado junto al Documento final

San José, Costa Rica, 7 de julio de 2023

Firmada digitalmente, de conformidad con la Ley de Certificados, Firmas Digitales y Documentos Electrónicos N° 8454, destacando el artículo 9°-

Tabla de Contenidos

1 Resumen 1

2 Introducción 2

2.1 Generalidades 2

2.2 Antecedentes del Problema 2

2.3 Definición y Descripción del Problema 3

2.4 Justificación 3

2.5 Viabilidad 4

2.5.1 Punto de Vista Técnico 4

2.5.2 Punto de Vista Operativo 5

2.5.3 Punto de Vista Económico 5

2.6 Objetivos 6

2.6.1 Objetivo general 6

2.6.2 Objetivos específicos 6

2.7 Alcances y Limitaciones 7

2.7.1 Alcances 7

2.7.2 Limitaciones 7

2.8 Estado de la cuestión 8

2.8.1 Planificación de la Revisión 8

2.8.2 Ejecución de la Revisión 10

2.8.3 Calidad de fuentes de información 41

2.8.4 Journal of Documentation 42

2.8.5 International Journal on Digital Libraries 42

2.8.6 International Journal of Medical System 43
2.8.7 Communications of the ACM 44
2.8.8 IEEE Data Engineering Bulletin 44
2.8.9 Foundations and Trends in Information Retrieval 44
2.8.10 Cambridge University Press 45
2.8.11 Text REtrieval Conference (TREC) & International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS) & Proceedings of the 7th conference on Information technology education 45
2.8.12 United States Patent 46
2.8.13 Journal of American Society for Information Science and Technology 46
2.8.14 VFAST Transactions on Software Enginnering 47
2.8.15 The Scientific World Journal 48
2.8.16 Resumen de Resultados 48
3 Marco Conceptual 51
3.1 Conceptos de la Recuperación de Información 52
3.1.1 Modelo de Espacio Vectorial (VSM) 54
3.1.2 Medición de resultados 55
3.1.3 Explicación de los resultados 56
3.2 Conceptos de Lucene y Elasticsearch 57
3.3 BM25 63
3.4 ElasticSearch 67
3.4.1 Arquitectura de ElasticSearch 70
4 Marco Metodológico 73
4.1 Tipo de Investigación 73

4.2 Alcance Investigativo 73
4.3 Enfoque 73
4.3.1 Dimensión Epistemológica 74
4.3.2 Dimensión Ontológica 74
4.3.3 Dimensión Axiológica 75
4.4 Diseño 76
4.5 Población y Muestreo 76
4.5.1 Colección de prueba Cranfield 77
4.5.2 Colección de prueba Medline 80
4.5.3 Colección de prueba Time 82
4.5.4 Comparación de las colecciones 85
4.6 Instrumentos de Recolección de Datos 91
4.6.1 Análisis Documental 91
4.6.2 Obtención de datos de la evaluación 92
4.7 Técnicas de Análisis de Información 95
5 Análisis del Diagnóstico 98
5.1 Term Query 98
5.2 Match Query 99
5.2.1 Match Query como un operador booleano 102
6 Propuesta de Solución 103
6.1 Resumen de la solución 103
6.2 Alcance de la implementación 109
6.2.1 Operador normTerm 110
6.2.2 Operador normOr 112

6.3 Implementación 114
6.3.1 NormTermQuery 115
6.3.2 NormOrQuery 119
6.3.3 NormOpsSimilarity 123
6.3.4 NormOpsPlugin 126
6.4 Evaluación de la solución 128
6.5 Resultados de la evaluación 135
6.5.1 Parametrización del operador NormOps 136
6.5.2 Evaluación colección Medline 139
6.5.3 Evaluación colección Cran 140
6.5.4 Evaluación colección Time 140
6.5.5 Evaluación de la normalización de los Score 141
6.5.6 Evaluación final 144
7 Conclusiones y Recomendaciones 148
7.1 Conclusiones 148
7.1.1 Conclusiones del Objetivo 1 148
7.1.2 Conclusiones del Objetivo 2 148
7.1.3 Conclusiones del Objetivo 3 149
7.1.4 Conclusiones del Objetivo 4 149
7.1.5 Conclusiones del Objetivo 5 150
7.1.6 Conclusión Objetivo General 151
7.2 Recomendaciones 152
8 Reflexiones Finales 153
9 Trabajos a Futuro 154

10 Bibliografía 155

Tabla de Ilustraciones

Ilustración 1 Validación de la calidad de la fuente Journal of Documentation. 42
Ilustración 2 Validación de la calidad de la fuente International Journal on Digital Libraries 43
Ilustración 3 Validación de la calidad de la fuente Journal of Medical Systems. 43
Ilustración 4 Validación de la calidad de la fuente Communications of the AMC..... 44
Ilustración 5 Validación de la calidad de la fuente Foundations and Trends in Information Retrieval 45
Ilustración 6 Validación de la calidad de la fuente Journal of the American Society for Information Science and Technology..... 47
Ilustración 7 Validación de la calidad de la fuente VFAST Transactions on Software Enginnering 47
Ilustración 8 Validación de la calidad de la fuente The Scientific World Journal 48
Ilustración 9 Nube de Palabras - Conceptos de Recuperación de Información 51
Ilustración 10 Conceptos Generales de la Recuperación de Información..... 52
Ilustración 11 Fórmula del puntaje F1 55
Ilustración 12 Fórmula del puntaje DCG..... 56
Ilustración 13 Fórmula del Puntaje nDCG..... 56
Ilustración 14 Fórmula de la similitud del coseno 59
Ilustración 15 Fórmula de la puntuación conceptual de Lucene..... 60
Ilustración 16 Fórmula de la puntuación práctica de Lucene..... 61
Ilustración 17 Fórmula de la Frecuencia del Término 62
Ilustración 18 Fórmula de la Frecuencia del documento inversa..... 62
Ilustración 19 Fórmula del BM25..... 63
Ilustración 20 Fórmula de la frecuencia inversa..... 65
Ilustración 21 Fórmula de la frecuencia inversa mejorada 65
Ilustración 22 Arquitectura Elastic Search..... 71

Ilustración 23 Proceso de consulta en un clúster de ElasticSearch 72
Ilustración 24 Ontología de las fórmulas y operadores de relevancia utilizados por Lucene. 74
Ilustración 25 Ejemplo de un documento en la colección Cranfield 78
Ilustración 26 Ejemplo de consulta de la colección Cranfield 79
Ilustración 27 Ejemplo del documento de relevancia de la colección Cranfield 80
Ilustración 28 Ejemplo de un documento en la colección Medline 81
Ilustración 29 Ejemplo de consulta en la colección Medline 81
Ilustración 30 Ejemplo del documento de relevancia de la colección Medline 82
Ilustración 31 Ejemplo de la etiqueta inicio de los documentos de la colección Time 83
Ilustración 32 Ejemplo de un documento de la colección Time 83
Ilustración 33 Ejemplo de consulta de la colección Time 84
Ilustración 34 Ejemplo del documento de relevancia de la colección Time 84
Ilustración 35 Diagrama de flujo de análisis de información. 93
Ilustración 36 Ejemplo de la estructura Match Query 100
Ilustración 37 Reescritura del operador "match" de Elasticsearch a una consulta booleana (simplificado) 102
Ilustración 38 Fórmula de la función Max. Fuente (P. E. Nelson & David, 2021) 105
Ilustración 39 Fórmula de la función Sum. Fuente (P. E. Nelson & David, 2021) 105
Ilustración 40 Fórmula de la función Sigmoide. Fuente (P. E. Nelson & David, 2021) 106
Ilustración 41 Gráfico de la función Sigmoide. Fuente Elaboración propia. 107
Ilustración 42 Fórmula de la función Sigmoide Truncada. Fuente (P. E. Nelson & David, 2021) 108
Ilustración 43 Ilustración 39 Gráfico de la función Sigmoide Truncada. Fuente Elaboración propia. 108
Ilustración 44 Ejemplo de uso del operador normTerm desarrollado en ElasticSearch. Fuente: Elaboración Propia. 111
Ilustración 45 Ejemplo de uso del operador normOr desarrollado en ElasticSearch. Fuente: Elaboración Propia. 113
Ilustración 46 Estructura de las clases de Lucene involucradas en la ejecución de la consulta.Fuente: adaptado de (P. Nelson, 2019) 115

Ilustración 47 Diagrama de clases para la clase NormTermQuery. Mostrando las clases implementadas para su funcionalidad. Fuente: Elaboración Propia. 116
Ilustración 48 Diagrama de clases para la clase NormOrQuery. Mostrando las clases implementadas para su funcionalidad. Fuente: Elaboración Propia. 120
Ilustración 49 Diagrama de clases para la clase NormOpsSimilarity. Mostrando las clases implementadas para su funcionalidad. Fuente: Elaboración Propia. 124
Ilustración 50 Diagrama de clases para la clase NormOpsPlugin. Mostrando las clases implementadas para su funcionalidad. 127
Ilustración 51 Implementación del método getQueries() utilizado para registrar los operadores desarrollados en Elasticsearch Fuente: Elaboración Propia. 128
Ilustración 52 Secuencia de pasos para la carga de los documentos de las colecciones a Elasticsearch para su indexado. Fuente: Elaboración Propia 131
Ilustración 53 Secuencia de pasos para la carga de las consultas y de los archivos de juicios de las colecciones, así como la ejecución de las consultas en Elasticsearch y el cálculo de las métricas de las consultas. Fuente: Elaboración Propia 133

Tabla de Figuras

Figura 1 Cantidad de palabras por documento según cada colección. Fuente Elaboración propia. 86
Figura 2 Cantidad de palabras por consulta según cada colección. Fuente Elaboración propia. 86
Figura 3 Cantidad de palabras promedio en los documentos por colección. Fuente Elaboración propia. 88
Figura 4 Cantidad de palabras media en los documentos por colección. Fuente Elaboración propia. 88
Figura 5 Cantidad de palabras máximas en los documentos por colección. Fuente Elaboración propia. 89
Figura 6 Cantidad de palabras promedio en las consultas por colección. Fuente Elaboración propia. 90
Figura 7 Cantidad de palabras media en las consultas por colección. Fuente Elaboración propia. 90
Figura 8 Cantidad de palabras máximas en las consultas por colección. Fuente Elaboración propia. 91
Figura 9 Resultados de las pruebas realizadas a la colección Medline 139
Figura 10 Resultados de las pruebas realizadas a la colección Cran 140
Figura 11 Resultados de las pruebas realizadas a la colección Time 141
Figura 12 Score del Operador NormOps en la colección Medline 142
Figura 13 Score del Operador de ElasticSearch en la colección Medline 142
Figura 14 Score del Operador NormOps en la colección Cran 143
Figura 15 Score del Operador de Elasticsearch en la colección Cran 143
Figura 16 Score del Operador NormOps en la colección Time 144
Figura 17 Score del Operador de Elasticsearch en la colección Time 144

Tabla de Tablas

Tabla 1 Desglose de salarios 5
Tabla 2 Costo de la Investigación 6
Tabla 3 Palabras Claves 9
Tabla 4 Extracción Fuente 1 11
Tabla 5 Extracción fuente 2 12
Tabla 6 Extracción Fuente 3 14
Tabla 7 Extracción fuente 4 15
Tabla 8 Extracción Fuente 5 16
Tabla 9 Extracción Fuente 6 17
Tabla 10 Extracción Fuente 7 18
Tabla 11 Extracción Fuente 8 20
Tabla 12 Extracción Fuente 9 22
Tabla 13 Extracción fuente 10 24
Tabla 14 Extracción Fuente 11 25
Tabla 15 Extracción Fuente 12 27
Tabla 16 Extracción Fuente 13 29
Tabla 17 Extracción Fuente 14 30
Tabla 18 Extracción Fuente 15 32
Tabla 19 Extracción Fuente 16 33
Tabla 20 Extracción Fuente 17 35
Tabla 21 Extracción Fuente 18 36
Tabla 22 Extracción fuente 19 37
Tabla 23 Extracción fuente 20 38
Tabla 24 Extracción Fuente 21 39
Tabla 25 Extracción fuente 22 40
Tabla 26 Resultados de Literatura 49
Tabla 27 Dimensión Axiológica 75
Tabla 28 Estadísticas básicas de las colecciones de Glasgow 85

Tabla 29 Comparativo de las estadísticas de los documentos de las colecciones de Glasgow.... 87
Tabla 30 Comparativo de las estadísticas de las consultas de las colecciones de Glasgow..... 89
Tabla 31 Descripción del proceso de análisis de información..... 93
Tabla 32 Pruebas de los parámetros utilizados en la colección Cran..... 136
Tabla 33 Pruebas de los parámetros utilizados en la colección Medline..... 137
Tabla 34 Pruebas de los parámetros utilizados en la colección Time 138
Tabla 35 Resultados de la Evaluación de los Operadores 145

1 Resumen

En la presente investigación se aborda la problemática de la asignación de puntajes variados a documentos bajo distintas circunstancias en las actuales máquinas de búsqueda que utilizan algoritmos TF/IDF, particularmente en ElasticSearch. La investigación evalúa un enfoque innovativo propuesto en la patente USP 11204920 mediante la implementación y evaluación de operadores de relevancia normalizados que generan un puntaje entre 0 y 1, buscando minimizar la variación en la asignación de relevancia de un documento en diferentes contextos de búsqueda. El estudio concluye que, aunque la superioridad del nuevo enfoque no se puede determinar de forma concluyente, los resultados en dos de las tres colecciones de datos probadas se acercaron lo suficiente a los métodos existentes, lo que sugiere que es viable continuar con la implementación de estos operadores como una alternativa a los métodos de relevancia actuales. La investigación también innova con la creación de un marco de trabajo que facilitó la comparación de los operadores y que puede ser utilizado en futuros estudios para comparar diferentes formas de calcular la relevancia.

2 Introducción

2.1 Generalidades

Con esta investigación se pretende implementar, en el motor de búsqueda de ElasticSearch, un subconjunto de los operadores propuestos en (P. E. Nelson & David, 2021) y poder llegar a evaluar sus beneficios en comparación con la implementación actual que ofrece ElasticSearch.

2.2 Antecedentes del Problema

Desde la adopción comercial de Internet, a mediados de la década de 1990, la recuperación de información pasó a ser la principal forma como las personas buscan y obtienen información(Manning et al., 2008). Esta adopción también se ha dado dentro de las empresas, donde es necesario realizar búsquedas en diversas fuentes de datos heterogéneas, como repositorios de documentos, páginas web, bases de datos transaccionales, *datalakes*, información de personal, y otros.

La mayoría de las máquinas de búsqueda actualmente utilizan, de forma importante, algoritmos de relevancia basados en TF/IDF (Beel et al., 2016) y están basadas en la biblioteca *Apache Lucene* (Solr, Elasticsearch, OpenSearch, AWS CloudSearch, Azure Search, Attivio, Swifttype) (Wikipedia Contributors, 2022). Estos algoritmos calculan la relevancia de los documentos con respecto a una consulta utilizando dos datos principalmente: la frecuencia de los términos en un documento (*TF*) y el número de documentos que contienen el término (I*DF*), y basado en estos números se le asigna un puntaje al documento con respecto a la consulta, el cual no está limitado en su magnitud en ninguna forma (Apache Software Foundation, 2022c).

Asimismo, cuando se realiza una consulta con operadores, como por ejemplo operadores booleanos AND y OR, Lucene suma los valores TF/IDF de cada término o campo, en el caso del AND, o elije el valor máximo en el caso del OR. Se comporta de manera similar para los demás operadores posibles de una consulta.

2.3 Definición y Descripción del Problema

En la práctica, se ha notado que estos algoritmos, basados en TF/IDF, tienden a asignar puntajes a los documentos de forma no esperada bajo ciertas circunstancias, como por ejemplo, si diferentes conjuntos de datos tienen un número diferente de documentos o una distribución diferente del término en la colección (cambiando el IDF), o si una misma oración se encuentra en documentos de diferente tamaño (variando el TF). Esto lleva a que un documento va a tener puntajes diferentes dependiendo de la colección en la que se encuentre (P. E. Nelson & David, 2021). Además, lleva a que búsquedas en colecciones heterogéneas, o inclusive en campos diferentes de un mismo documento, obtengan valores de relevancia que no deberían de combinarse directamente.

Esta investigación busca evaluar los operadores de relevancia de los motores de búsqueda actuales, en este caso los que se utilizan en el motor de búsqueda ElasticSearch, respecto a nuevos operadores de relevancia planteados por (P. E. Nelson & David, 2021).

Estos operadores generan un puntaje normalizado entre 0 y 1 en todos los puntos de la consulta. Con esto se busca poder normalizar, de cierta manera, la relevancia de un documento en una búsqueda a este puntaje, ya que actualmente la búsqueda de un mismo documento, bajo los mismos criterios, bajo el mismo conjunto de datos con los operadores de relevancia actuales, puede resultar en un puntaje de relevancia distinto, por ende, con este nuevo enfoque de relevancia se buscará normalizar estos resultados y que el puntaje de relevancia no sea tan variado y así poder determinar la efectividad de este nuevo enfoque de relevancia normalizado sobre los actuales.

2.4 Justificación

La recuperación de información es un área de suma importancia tanto en el mundo académico como en la industria. La necesidad de buscar y encontrar información relevante en grandes conjuntos de datos ha impulsado el desarrollo de sistemas más eficientes y precisos. En este contexto, los motores de búsqueda son herramientas esenciales para la gestión de información. Sin embargo, sus métodos de relevancia pueden dar lugar a resultados inesperados en ciertas

circunstancias. Por lo tanto, es esencial buscar alternativas que puedan mejorar la precisión y la efectividad de estos sistemas.

Por otro lado, la propuesta de operadores de relevancia de (P. E. Nelson & David, 2021) representa un nuevo enfoque en la forma en que se calcula la relevancia en los motores de búsqueda. El objetivo es proporcionar una solución a los problemas identificados con los algoritmos actuales, generando puntajes normalizados entre 0 y 1 en todos los puntos de la consulta. Este nuevo enfoque tiene el potencial de mejorar la precisión de los resultados y hacer más fácil la comparación y la interoperabilidad entre diferentes consultas y conjuntos de datos.

El aporte y la innovación de la presente investigación es la implementación, el análisis y la evaluación de estos operadores de relevancia normalizados y su comparación con los operadores existentes. Esto no solo nos permitirá evaluar la efectividad del nuevo enfoque, sino que también puede conducir a mejorar la precisión y eficiencia de las máquinas de búsqueda en el futuro. De igual manera, otro aporte importante es el mecanismo para comparar los dos tipos de operadores que pueda adaptarse a comparar varias métricas de recuperación de información a futuro.

2.5 Viabilidad

Si bien la investigación sobre algoritmos de recuperación de información y motores de búsqueda, a nivel regional, no es muy diversa y no existen estudios similares, es un tema que se ha tratado por diversos autores a nivel internacional, los cuales dan relevancia a la misma. Además, a continuación, se detallan los aspectos técnicos, operativos y económicos por lo que realizar esta investigación es viable.

2.5.1 Punto de Vista Técnico

Los autores de la presente investigación cuentan con títulos académicos en disciplinas relacionadas con las ciencias de la computación, así como experiencia en desarrollo de software y en resolución de problemas técnicos de la industria.

2.5.2 Punto de Vista Operativo

La máquina de búsqueda donde se realizará la implementación, necesaria para la investigación, es de código fuente abierto, y todo el código y la documentación de esta está disponible para su descarga y uso. Así mismo, existen diversas colecciones de documentos para la evaluación del desempeño de máquinas de búsqueda, las cuales están disponibles y se pueden utilizar para las evaluaciones y comparaciones del estudio.

2.5.3 Punto de Vista Económico

Ya que el proyecto puede ofrecer resultados para mejorar las máquinas de búsqueda que se utilizan para fines comerciales, el costo teórico del mismo, para su desarrollo, es dado en términos de horas de investigación, las cuales corren por cuenta de los autores de esta investigación.

Para sacar los costos teóricos de este proyecto se utilizan de referencia los salarios base según un estudio realizado (Deloitte, 2021) en conjunto por Deloitte Costa Rica, la Cámara Costarricense de Tecnologías de Información y Comunicación (CAMTIC) y el Colegio de Profesionales en Informática y Computación (CPIC) en el año 2021.

Considerando el estudio mencionado y el rango académico de los autores, el estudio estima que el salario anual ronda entre los $42000 y los $65000, utilizando el apartado de la compensación total a los colaboradores según este estudio. Cabe aclarar que dichas cifras en dólares se tomaron pasando los salarios propuestos en colones según al tipo de cambio de 683.76 colones tomado el día 12 de julio del 2022

A continuación, se detalla el desglose de estos salarios por cada autor:

Tabla 1 Desglose de salarios

Desglose	Salario Anual	Salario Mensual	Salario Diario	Salario por Hora
José Aguilar	$65000	$5416.66	$180.55	$7.5

Jean Carlo Álvarez	$42000	$3500	$166.66	$4.86

Fuente Elaboración propia basada en la información del estudio (Deloitte, 2021)

Por otro lado, se detalla el costo de la investigación según las horas estimadas y según el desglose de salarios previamente realizado.

Tabla 2 Costo de la Investigación

Investigación	Horas Semanales	Duración TFG Meses	Duración TFG Horas	Costo estimado por Horas
José Aguilar	30	4	480	$3600
Jean Carlo Álvarez	30	4	480	$2332.8
	600	8	960	$5932.8

Fuente Elaboración propia basada en la información del estudio (Deloitte, 2021)

2.6 Objetivos

Se usa la taxonomía de Bloom de 1956 debido a que existe extensa documentación al respecto, su estructuración jerárquica facilita la definición de los objetivos desde los niveles más generales a los más específicos y por su amplia utilización.

2.6.1 Objetivo general

Evaluar un nuevo enfoque de operadores de relevancia normalizados en máquinas de búsqueda por medio de su implementación y comparación con los operadores existentes para determinar su efectividad.

2.6.2 Objetivos específicos

1. Describir los diferentes métodos de relevancia en máquinas de búsqueda de alto uso para obtener una perspectiva de los métodos más utilizados.

2. Enumerar los métodos de evaluación de relevancia de máquinas de búsqueda con el fin de escoger un método de evaluación a utilizar para comparar los diferentes enfoques.
3. Comprender los operadores de relevancia expuestos en (P. E. Nelson & David, 2021) para que generen puntajes normalizados para la máquina de búsqueda de alto uso.
4. Construir un subconjunto de los operadores de relevancia planteados en la patente expuesta en (P. E. Nelson & David, 2021) para la máquina de búsqueda de alto uso para poder compararlos con los métodos utilizados actualmente.
5. Comparar los nuevos operadores en contraste con los incluidos en la máquina de búsqueda de alto uso para determinar su efectividad o sus beneficios.

2.7 Alcances y Limitaciones

2.7.1 Alcances

Documento detallado de la investigación en donde se incluyen el conjunto de datos en inglés, la descripción de los operadores implementados para la comparación y los resultados de la investigación.

2.7.2 Limitaciones

- Se considera realizar la investigación en el motor de búsqueda ElasticSearch.
- Se limita la comparación a los operadores incluidos e implementados en ElasticSearch.
- Solamente se implementará lo necesario para poder realizar una comparación contra los operadores incluidos e implementados en Elasticsearch necesarios para realizar una búsqueda de texto de acuerdo con las colecciones seleccionadas para la evaluación.
- Dado que corresponde a la implementación de unos operadores descritos en una patente, es necesario revisarlos legalmente antes de poder distribuirlos como código fuente abierto. Si bien los autores de la patente piensan que puede ser posible su publicación, será en un futuro que se publique el código desarrollado junto con algunas modificaciones con el fin de no infringir la patente.

2.8 Estado de la cuestión

Existe una amplia documentación sobre la recuperación de información, así como de los modelos y operadores de relevancia utilizados, tanto en el mercado como en las investigaciones que se han realizado de los mismos, buscando mejorar su funcionamiento y efectividad a la hora de poder recuperar resultados de consultas relevantes en una búsqueda para el usuario. Se busca realizar una compilación de documentos, los cuales puedan llegar a ampliar la investigación, para ello se realiza un análisis de cada uno de estos buscando aquellos que sean relevantes para esta investigación y realizar así una selección de estos.

2.8.1 Planificación de la Revisión

Se realiza una búsqueda exhaustiva de la documentación existente acerca del tema a investigar y evaluar, con esto se pretende conocer el desarrollo de este, tanto en el ámbito académico como en el investigativo a nivel internacional, y así ver aquellos trabajos que pueden aportar algún valor a la investigación y permitir realizar una evaluación más efectiva y amplia, además de evitar esfuerzos similares realizados en esta documentación consultada.

Para realizar esta búsqueda de documentación se debe formular, de manera clara y bien definida, la pregunta que permita limitar la búsqueda de información para la investigación y permita tener nada más aquella documentación que puede llegar a ser relevante para el tema, esto con el objetivo de que dé una contribución al trabajo. Por otro lado, se debe centralizar la búsqueda en documentación sobre modelos de puntajes de relevancia en la recuperación de información, así como documentos técnicos sobre operadores de relevancia en motores de búsqueda. Además, se realiza una lista de los términos claves e importantes del tema para la búsqueda de información, con algunas palabras claves, tanto en español como en inglés, esto ya que la mayor parte de las publicaciones al respecto se encuentran en el idioma inglés.

A continuación, se detalla una pequeña tabla con algunas de estas palabras:

Tabla 3 Palabras Claves

Palabra	Equivalente en Ingles
Relevancia	Relevancy
Motor	Engine
Normalizado	Normalized
Recuperación	Retrieval
Modelos	Models
Búsqueda	Search
Información	Information
Operadores	Operators
Comparable	Comparable

Fuente: Elaboración propia.

Por otra parte, se realiza una selección de las fuentes más importantes que se utilizan para consultar información al respecto, para esto se toma en cuenta la popularidad de la fuente en la comunidad, que la misma posea una gran variedad de documentación y sea de fácil acceso o bien contar con las credenciales necesarias para poder acceder a dicha información. Toda esta búsqueda se realiza, como ya se mencionó anteriormente, tanto en español como en el idioma inglés por las razones previamente dadas. Entre las fuentes consultadas dada su relevancia académica y a la cantidad de artículos sobre el tema se encuentran:

1. Journal of Documentation
2. International Journal on Digital Libraries
3. Journal of Medical Systems
4. Journal of the American Society for Information Science and Technology
5. Communications of the AMC
6. IEEE
7. Text REtrieval Conference (TREC)
8. International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS)
9. United States Patent

10. VFAST Transactions on Software Enginnering
11. Foundations and Trends in Information Retrieval
12. Cambridge University Press
13. Proceedings of the 7th cnference on Information technology education
14. The Scientific World Journal

Además de estas fuentes, también se tienen en cuenta algunos libros relevantes en el tema, así como algunas patentes sobre modelos de relevancia.

A pesar de no contar con un criterio experto, se ha realizado la selección de estas fuentes con base en su popularidad en la comunidad y la cantidad de artículos respecto al tema que contenían, por otro lado, se tomó en cuenta que muchos de ellos son citados en diversas investigaciones.

Ya definidas las fuentes, se procede a seleccionar los artículos y documentos de mayor relevancia para la investigación, para ello se toma en cuenta que los mismos posean información con base en las palabras claves previamente definidas, todos aquellos artículos de recuperación de información que no se enfoquen en modelos y operadores de relevancia son excluidos, esto ya que se busca incluir únicamente aquellos artículos que permitan expandir el tema y se encuentren alineados a los objetivos planteados y así poder seleccionar los más relevantes.

2.8.2 Ejecución de la Revisión

Se analiza cada uno de los documentos obtenidos en las fuentes de información mencionadas con anterioridad, además se hace la selección de aquellos documentos que se utilizan en la investigación dada su relevancia para la misma.

2.8.2.1 Literatura evaluada

Se presenta la literatura evaluada para el proyecto, la misma se detalla según cada una de las fuentes, y en cada una de ellas se encuentra información relevante sobre la misma.

Cada una de estas búsquedas se realizó utilizando los criterios de búsqueda mencionados previamente y usando aquellas palabras claves que permitan encontrar documentos relevantes de acuerdo al tema planteado para esta investigación. A continuación, se detalla aquella literatura recuperada de dichas fuentes:

2.8.2.2 Fuente Journal of Documentation

Tabla 4 Extracción Fuente 1

Titulo	Understanding Inverse Document Frequency: On Theoretical Arguments for IDF
Publicación	Journal of Documentation
Autores	Robertson, S.
Referencia	Robertson, S. (2004). Understanding Inverse Document Frequency: On Theoretical Arguments for IDF. Journal of Documentation - J DOC, 60, 503–520. https://doi.org/10.1108/0022041041056058 2
Descripción	
Área	Recuperación de Información, Máquinas de búsqueda, Relevancia, Modelos de Relevancia
Resumen	La función de ponderación de términos conocida como IDF se propuso en 1972 y, desde entonces, se ha utilizado ampliamente, generalmente como parte de una función TF*IDF. A menudo se describe como una heurística y se han escrito muchos artículos (algunos basados en la teoría de la información de Shannon) que buscan establecer alguna base teórica para ello. Se revisan algunos de estos intentos y se muestra

	que los enfoques de la Teoría de la Información son problemáticos, pero que existen buenas justificaciones teóricas tanto de IDF como de TF*IDF en el modelo probabilístico tradicional de recuperación de información.
Aspectos Que Destacar	
Bases teóricas del modelo que utiliza TF/IDF	

Fuente: Elaboración propia

Tabla 5 Extracción fuente 2

Titulo	A STATISTICAL INTERPRETATION OF TERM SPECIFICITY AND ITS APPLICATION IN RETRIEVAL
Publicación	Journal of Documentation
Autores	Sparck Jones, K.
Referencia	Sparck Jones, K. (1972). A STATISTICAL INTERPRETATION OF TERM SPECIFICITY AND ITS APPLICATION IN RETRIEVAL. Journal of Documentation, 28(1), 11–21. https://doi.org/10.1108/eb026526
Descripción	
Área	Recuperación de Información, Máquinas de búsqueda, Relevancia, Modelos de Relevancia

Resumen	La exhaustividad de las descripciones de los documentos y la especificidad de los términos del índice generalmente se consideran independientes. Se sugiere que la especificidad debe ser interpretada estadísticamente, como una función del uso del término más que del significado del término. Se examinan los efectos sobre la recuperación de las variaciones en la especificidad de los términos, con tres colecciones de prueba que muestran en particular que ocurren con frecuencia. Se requieren términos adicionales para un buen desempeño general. Se argumenta que los términos deben ponderarse de acuerdo con la frecuencia de recopilación, de modo que las coincidencias en los términos menos frecuentes y más específicos sean de mayor valor que las coincidencias en términos frecuentes. Los resultados de las colecciones de prueba muestran que las mejoras en el rendimiento se obtienen con este procedimiento muy simple.
Aspectos Que Destacar	
Bases del modelo que incluye TF	

Fuente: Elaboración propia

2.8.2.3 Fuente Text REtrieval Conference (TREC)

Tabla 6 Extracción Fuente 3

Titulo	The Excalibur TREC-4 System, Preparations, and Results
Publicación	Proceedings of The Fourth Text REtrieval Conference
Autores	Nelson, Paul
Referencia	Nelson, P. E. (1995). The Excalibur TREC-4 System, Preparations, and Results. In D. K. Harman (Ed.), Proceedings of The Fourth Text REtrieval Conference, TREC 1995, Gaithersburg, Maryland, USA, November 1-3, 1995 (Vols. 500–236). National Institute of Standards and Technology (NIST). http://trec.nist.gov/pubs/trec4/papers/excalibur.ps.gz
Descripción	
Área	Recuperación de Información, Máquinas de búsqueda, Relevancia
Resumen	
Aspectos Que Destacar	
Describe cómo funciona la arquitectura de una máquina de búsqueda comercial, incluidos los diferentes módulos que interactúan para definir la relevancia y el *ranking* de los documentos.	

Describe la interacción entre las diferentes partes en una máquina de búsqueda comercial para determinar la relevancia de un documento. Describe el peso de los términos en los documentos.

Fuente: Elaboración propia

Tabla 7 Extracción fuente 4

Titulo	Okapi at TREC-3
Publicación	TREC-3
Autores	Robertson, S., Walker, S., Jones, S., Hancock-Beaulieu, M. M., & Gatford, M.
Referencia	Robertson, S., Walker, S., Jones, S., Hancock-Beaulieu, M. M., & Gatford, M. (1995). Okapi at TREC-3. 109–126.
Descripción	
Área	Recuperación de Información, Máquinas de búsqueda, Relevancia, Modelos de Relevancia
Resumen	El software Okapi utilizado para TREC-3 era similar al utilizado en TREC anteriores, y comprendía un sistema de búsqueda básico de bajo nivel y una interfaz de usuario para los experimentos de búsqueda manual, junto con utilidades de conversión e inversión de datos. También había varios scripts y programas para generar términos de consulta, ejecutar lotes de pruebas y realizar evaluaciones. […]

Aspectos Que Destacar
Modelo de relevancia Okapi que utiliza Elasticsearch

Fuente: Elaboración propia

2.8.2.4 Fuente Journal of Medical System

Tabla 8 Extracción Fuente 5

Titulo	Current Status of the Evaluation of Information Retrieval
Publicación	Journal of Medical Systems
Autores	Kagolovsky, Y., & Moehr, J. R
Referencia	Kagolovsky, Y., & Moehr, J. R. (2003). Current Status of the Evaluation of Information Retrieval. Journal of Medical Systems, 27(5), 409–424. https://doi.org/10.1023/A:1025603704680
Descripción	
Área	Recuperación de Información, Evaluación, Relevancia
Resumen	El artículo brinda un resumen del estado actual de cómo se evalúan los sistemas de recuperación de información, las razones por las que se evalúan los sistemas de IR, además de dar a conocer los diferentes tipos de evaluación que se utilizan y analiza varios enfoques. Por otra parte, da a conocer los problemas existentes a la hora de realizar una evaluación de este tipo de sistemas, y

	da una amplia explicación del modelo de evaluación de Cranfield y trata de presentar sus problemas. Por último, busca dar un pequeño vistazo a posibles alternativas de evaluación a este modelo, así como los nuevos paradigmas de la evaluación.
Aspectos Que Destacar	
Amplia vista al Modelo de Cranfield Medidas de Recall y precisiones basadas en la relevancia Paradigma Cognitivo centrado en el usuario	

Fuente: Elaboración propia

2.8.2.5 Fuente IEEE Data Engineering Bulletin

Tabla 9 Extracción Fuente 6

Titulo	Modern Information Retrieval: A Brief Overview
Publicación	IEEE Data Engineering Bulletin
Autores	Singhal, A., & Google
Referencia	Singhal, A., & Google, I. (2001). Modern Information Retrieval: A Brief Overview. IEEE Data Engineering Bulletin, 24.
Descripción	

Área	Recuperación de Información, Máquinas de búsqueda, Relevancia, Modelos de Relevancia
Resumen	Durante miles de años, la gente se ha dado cuenta de la importancia de archivar y encontrar información. Con la llegada de las computadoras se hizo posible almacenar grandes cantidades de información; y encontrando útil la información de tales colecciones se convirtió en una necesidad. Nació el campo de la Recuperación de Información (IR) en la década de 1950 por esta necesidad. Durante los últimos cuarenta años, el campo ha madurado considerablemente. Varios sistemas IR son utilizados a diario por una amplia variedad de usuarios. Este artículo es una breve descripción de los avances clave en el campo de la recuperación de información, y una descripción de dónde se encuentra el estado del arte en el campo.
Aspectos Que Destacar	
Revisión de los diferentes componentes de la recuperación de la información.	

Fuente: Elaboración propia

2.8.2.6 Fuente International Journal on Digital

Tabla 10 Extracción Fuente 7

Titulo	Research-paper recommender systems

Publicación	*International Journal on Digital Libraries*
Autores	Beel, J., Gipp, B., Langer, S., & Breitinger, C
Referencia	Beel, J., Gipp, B., Langer, S., & Breitinger, C. (2016). Research-paper recommender systems: A literature survey. International Journal on Digital Libraries, 17(4), 305–338. https://doi.org/10.1007/s00799-015-0156-0
Descripción	
Área	Recuperación de Información, Máquinas de búsqueda, Relevancia
Resumen	El articulo habla sobre cómo se realizó un estudio a artículos sobre sistemas de recomendación y se presentaron algunas estadísticas de los mismos dando a conocer porcentajes de algunos de los enfoques que tienen ciertos sistemas de recomendación, además se da una discusión sobre algunos avances y deficiencias de estos sistemas como el uso de conjuntos de datos podados, pocos participantes en los estudios además de la poca información que proporcionan ciertos investigadores de los algoritmos utilizados para estos sistemas, por otro lado se da una descripción general de los conceptos más importantes así como una explicación sobre los enfoques de recomendación más comunes, además al final se logró ver como pocos artículos realmente llegan a tener en los sistemas de

	recomendación, todo esto abre varias acciones que se pueden tomar en cuenta para cambiar el panorama en la investigación de sistemas de recomendación como darle mayor enfoque a aspectos como la modelación del usuario o bien una plataforma para que los investigadores intercambien información al respecto.
Aspectos Que Destacar	
Se habla de la importancia de la satisfacción del usuario y no solo de la precisión Se habla de poder llegar a desarrollar un marco de evaluación común La mitad de las recomendaciones se basan en el contenido	

Fuente: Elaboración propia

2.8.2.7 Fuente International Journal of the American Society for Information Science and Technology

Tabla 11 Extracción Fuente 8

Titulo	Form and function: The impact of query term and operator usage on Web search results
Publicación	Journal of the American Society for Information Science and Technology
Autores	Lucas, W., & Topi, H
Referencia	Lucas, W., & Topi, H. (2002). Form and function: The impact of query term and operator usage on Web search results. Journal of the American

	Society for Information Science and Technology, 53(2), 95–108. https://doi.org/10.1002/asi.10013
Descripción	
Área	Recuperación de Información, Máquinas de búsqueda, Consulta, Operadores
Resumen	El artículo es un estudio realizado para poder determinar qué tanto impacto en la relevancia tiene el resultado de una consulta cuando es realizada utilizando términos y operadores adecuados a la hora de realizar una búsqueda en un motor de búsqueda web, para esto se realiza una comparación entre expertos y no expertos en el uso de términos y operadores de búsqueda. Dicho estudio pretende dar a conocer la diferencia porcentual de la relevancia de un resultado con base en el uso correcto y erróneo de estos operadores. Además, resalta la necesidad de mejorar las interfaces de estos motores para poder guiar a un no experto en el uso de los operadores y términos ya que como bien se ve en el estudio realizado, la diferencia de resultado es evidente cuando se usan bien estos términos y operadores en los motores de búsqueda
Aspectos Que Destacar	
La relevancia del uso de operadores y términos correctos en un motor de búsqueda Las interfaces pobres de los motores de búsqueda para dar soporte a no expertos	

Fuente: Elaboración propia

2.8.2.8 Fuente Communications of the ACM

Tabla 12 Extracción Fuente 9

Titulo	IR evaluation methods for retrieving highly relevant documents
Publicación	ACM
Autores	Järvelin, K., & Kekäläinen, J.
Referencia	Järvelin, K., & Kekäläinen, J. (2017). IR evaluation methods for retrieving highly relevant documents. ACM SIGIR Forum, 51(2), 243–250. https://doi.org/10.1145/3130348.3130374
Área	Recuperación de Información, Máquinas de búsqueda, Relevancia
Resumen	Este artículo propone métodos de evaluación basados en el uso de juicios de relevancia no dicotómicos en experimentos de IR. Se argumenta que los métodos de evaluación deberían dar crédito a los métodos de IR por su capacidad para recuperar documentos de gran relevancia. Esto es deseable desde el punto de vista del usuario en entornos IR modernos de gran tamaño. Los métodos propuestos son (1) una aplicación novedosa de curvas P-R y cálculos de precisión promedio basados en bases de recuperación separadas para documentos de diferentes grados de relevancia, y (2) dos medidas novedosas que calculan la ganancia acumulada que el usuario

	obtiene al examinar el resultado de la recuperación, a una posición clasificada determinada. Luego se demuestra el uso de estos métodos de evaluación en un estudio de casos sobre la efectividad de los tipos de consulta, basados en combinaciones de estructuras de consulta y expansión, en la recuperación de documentos de diversos grados de relevancia. La prueba se ejecutó con un sistema de recuperación de mejores coincidencias en una base de datos de texto que consiste en artículos de periódicos. Los resultados indican que las estructuras de consulta sólidas probadas son más efectivas para recuperar documentos muy relevantes. Las diferencias entre los tipos de consulta son prácticamente esenciales y estadísticamente significativas. En términos más generales, los nuevos métodos de evaluación y el caso demuestran que las evaluaciones de relevancia no dicotómicas son aplicables en los experimentos de IR, y pueden revelar fenómenos interesantes y permitir pruebas más duras de los métodos de IR.
Aspectos Que Destacar	
Documenta como se puede evaluar la relevancia de los documentos. En la presente investigación aporta formas de evaluar el comportamiento de la relevancia de nuevos operadores.	

Fuente: Elaboración propia

Tabla 13 Extracción fuente 10

Titulo	A vector space model for automatic indexing
Publicación	Communications of the ACM
Autores	Salton, G., Wong, A., & Yang, C. S.
Referencia	Salton, G., Wong, A., & Yang, C. S. (1975). A vector space model for automatic indexing. Communications of the ACM, 18(11), 613–620. https://doi.org/10.1145/361219.361220
Descripción	
Área	Recuperación de Información, Máquinas de búsqueda, Relevancia
Resumen	En un entorno de recuperación de documentos u otro entorno de coincidencia de patrones donde las entidades almacenadas (documentos) se comparan entre sí o con patrones entrantes (solicitudes de búsqueda), parece que el mejor espacio de indexación (propiedad) es aquel en el que cada entidad se encuentra lo más lejos que sea posible de las demás; en estas circunstancias, el valor de un sistema de indexación puede expresarse en función de la densidad del espacio del objeto; en particular, el rendimiento de recuperación puede correlacionarse inversamente con la densidad del espacio. Se utiliza un enfoque basado en cálculos de densidad espacial para elegir un vocabulario de indexación

	óptimo para una colección de documentos. Se muestran los resultados típicos de la evaluación, lo que demuestra la utilidad del modelo.
Aspectos Que Destacar	
Artículo importante en la recuperación de información que introduce y detalla el modelo más utilizado en las máquinas de búsqueda.	

Fuente: Elaboración propia

2.8.2.9 Fuentes United States Patent

Tabla 14 Extracción Fuente 11

Titulo	United States Patent: 11204920 - Utilizing search engine relevancy ranking models to generate normalized and comparable search engine scores
Publicación	United States Patent
Autores	Nelson, Paul; David, Mark
Referencia	Nelson, P. E., & David, M. (2021). United States Patent: 11204920 - Utilizing search engine relevancy ranking models to generate normalized and comparable search engine scores (Patent No. 11204920).
Descripción	
Área	Recuperación de Información, Máquinas de búsqueda, Relevancia

Resumen	Un dispositivo puede recibir una consulta especificada por el usuario y puede construir la consulta especificada por el usuario como un árbol de consulta de operadores que incluyen cero o más operadores. El dispositivo puede producir, para cada uno de los operadores, un peso que indica cuán valiosa o confiable es una determinación de la relevancia de cada uno de los operadores en relación con los hermanos en el árbol de consulta. El dispositivo puede normalizar los pesos de los hermanos en el árbol de consulta con un operador principal según una fórmula de normalización. El dispositivo puede producir, para cada uno de los operadores, una puntuación, normalizada a un valor de cero a uno y calculado para el documento, que representa cuán relevante es el documento para la consulta especificada por el usuario. El dispositivo puede aplicar el árbol de consulta al documento en su conjunto, así como a las posiciones dentro del documento para determinar el puntaje normalizado final para el documento.
Aspectos Que Destacar	
Es la base del presente trabajo. El trabajo consiste en implementar lo anterior y determinar sus beneficios.	

Fuente: Elaboración propia

2.8.2.10 Fuente International Conference on Energy, Communication, Data Anlytics and Soft Computing (ICEDS)

Tabla 15 Extracción Fuente 12

Titulo	Information retrieval from heterogeneous data sets using moderated IDF-cosine similarity in vector space model
Publicación	International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS)
Autores	Pathak, B., & Lal, N.
Referencia	Pathak, B., & Lal, N. (2017). Information retrieval from heterogeneous data sets using moderated IDF-cosine similarity in vector space model. 2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS), 3793–3799. https://doi.org/10.1109/ICECDS.2017.8390174
Descripción	
Área	Recuperación de Información, Máquinas de búsqueda, Relevancia
Resumen	En un mundo digital, la recuperación de información se utiliza para varias aplicaciones, como buscar información en un documento, buscar documentos en sí mismos, buscar

	metadatos que representan datos y bases de datos como texto, HTML, XML, imágenes, audio, etc. Muchas universidades, escuelas y las empresas utilizan estos sistemas IR para proporcionar acceso a libros, revistas y otros documentos requeridos en las empresas. Los motores de búsqueda son la aplicación IR más observable. Nuestro propósito es componer un sistema que busque en nuestra colección de conjuntos de datos y recupere el documento que coincida con la consulta del usuario. Estas colecciones de documentos son archivos de texto, PDF, HTML, archivos XML que llamamos conjuntos de datos heterogéneos. Para obtener documentos relevantes en los resultados finales, primero debemos indexar los documentos y luego clasificarlos según la puntuación de los valores de similitud del coseno para cada documento que coincida con la consulta del usuario que estamos ingresando. Estamos utilizando la técnica de similitud de coseno en el modelo de espacio vectorial principalmente para nuestro propósito de análisis y comparando con la similitud moderada de IDF-Coseno para obtener mejores resultados de búsqueda utilizando R Studio. Nuestro enfoque brinda mejores resultados en el corpus del mundo real en comparación con el

	enfoque mejorado de similitud IDF-Cosine en el modelo de espacio vectorial.
Aspectos Que Destacar	
Proponen un modelo que modifica el IDF para obtener una comparación válida entre fuentes de datos heterogéneas. Esto se alinea con uno de los problemas que se quieren resolver con el presente trabajo.	

Fuente: Elaboración propia

2.8.2.11 VFAST Transactions on Software Enginnering

Tabla 16 Extracción Fuente 13

Titulo	Comparison and Evaluation of Information Retrieval Models
Publicación	VFAST Transactions on Software Engineering
Autores	Rehma, et. al.
Referencia	Rehma, A. A., Awan, M. J., & Butt, I. (2018). Comparison and Evaluation of Information Retrieval Models. VFAST Transactions on Software Engineering, 6(1), 7–14. https://doi.org/10.21015/vtse.v13i1.496
Descripción	
Área	Recuperación de Información, Máquinas de búsqueda, Relevancia

Resumen	Recientemente, los datos están creciendo día a día en Internet. Los datos están en forma de naturaleza estructurada, no estructurada y semiestructurada. La recuperación de información es el campo que se refiere al estudio de la recuperación de documentos no estructurados o semiestructurados. Para cada aspecto en el que se utiliza IR, se utilizan diferentes modelos en ellos. Hay tantos modelos en tantas aplicaciones, cada uno con alguna relación entre sí. En este documento evaluaremos y compararemos varias técnicas y algoritmos de modelos IR y veremos qué modelo sobresale en qué campo de aplicación.
Aspectos Que Destacar	
Comparación de modelos de recuperación de información. Tomar formas de compararlos.	

Fuente: Elaboración propia

2.8.2.12 Fuente Proceedings of the 7th conference on Information technology education

Tabla 17 Extracción Fuente 14

Titulo	An infrastructure for the evaluation and comparison of information retrieval systems

Publicación	Proceedings of the 7th conference on Information technology education
Autores	Broadbent, R. E., Saunders, G. S., & Ekstrom, J. J.
Referencia	Broadbent, R. E., Saunders, G. S., & Ekstrom, J. J. (2006). An infrastructure for the evaluation and comparison of information retrieval systems. Proceedings of the 7th conference on Information technology education, 123–127. https://doi.org/10.1145/1168812.1168842
Descripción	
Área	Recuperación de Información, Máquinas de búsqueda, Relevancia
Resumen	El mismo presenta los conceptos que se utilizan para la evaluación de los sistemas de recuperación de información y a la vez da a conocer la implementación de un marco, el cual permite evaluar el rendimiento de la indexación y permite, a través de dicho marco, comparar productos IR dando un entorno para la medición subjetiva, utilizando evaluadores humanos. Todo esto mostrado a través de un pequeño caso de prueba. Esto lo que busca es poder llegar a comparar con precisión diferentes sistemas de recuperación de información y eventualmente dar con un modelo que lo permita.
Aspectos Que Destacar	

El uso de Lucene, CLucene y Glimpse para realizar el caso de prueba Se compara la velocidad de indexación de cada solución

Fuente: Elaboración propia

2.8.2.13 Fuente Foundations and Trends in Information Retrieval

Tabla 18 Extracción Fuente 15

Titulo	Information Retrieval: The Early Years
Publicación	Foundations and Trends® in Information Retrieval
Autores	Harman, D
Referencia	Harman, D. (2019). Information Retrieval: The Early Years. Foundations and Trends® in Information Retrieval, 13(5), 425–577. https://doi.org/10.1561/1500000065
Descripción	
Área	Recuperación de Información, Máquinas de búsqueda, Relevancia
Resumen	El mismo trata de explicar la evolución y el crecimiento de la ciencia detrás de los motores de búsqueda, lo que hoy conocemos como recuperación de información desde sus inicios. Busca dar un vistazo de cómo fue creciendo esta ciencia a través de los aportes de algunos matemáticos y lingüistas, los cuales buscaban cada vez mejores formas de indexar textos para luego utilizar algoritmos que les permitieran

	buscar estos índices, pero en aquella época era algo difícil ya que la tecnología del momento era limitada, además de no contar con suficiente materia prima (texto legible), sin embargo, a través de la década se vieron grandes progresos en este campo gracias a la aparición de lo que hoy conocemos como internet y la aparición de motores de búsqueda.
Aspectos Que Destacar	
Contiene una línea de tiempo de la evolución de la recuperación de información Detalla los avances de diferentes motores de búsqueda comerciales y académicos	

Fuente: Elaboración propia

2.8.2.14 Fuente Cambridge University Press

Tabla 19 Extracción Fuente 16

Titulo	Introduction to Information Retrieval
Publicación	Cambridge University Press
Autores	Manning, C. D., Raghavan, P., & Schütze, H.
Referencia	Manning, C. D., Raghavan, P., & Schütze, H. (2008). Introduction to Information Retrieval. Cambridge University Press. https://nlp.stanford.edu/IR-book/information-retrieval-book.html
Descripción	

Área	Recuperación de Información, Máquinas de búsqueda, Relevancia
Resumen	Se da a conocer y se explican todos los conceptos básicos sobre la recuperación de información, todo aquello que envuelve a esta ciencia. Se conocen términos y vocabulario de la misma, así como los diferentes modelos más conocidos que se utilizan en la recuperación de información, como el booleano, el vectorial y el probabilístico. Además, se ven conceptos como indexar, el peso de un término, el puntaje de relevancia, los sistemas de búsqueda y los motores de búsqueda web y como estos funcionan. Por otro lado, se conocen conceptos como la semántica de una consulta o bien las matrices de descomposición e incluso el uso de "machine learning" en estos sistemas. Se da un vistazo a todo aquello que envuelve a la recuperación de información y todos los conceptos básicos que se deben conocer a la hora de querer trabajar, estudiar o investigar en este campo.
Aspectos Que Destacar Detalla los conceptos básicos sobre la recuperación de información	

Fuente: Elaboración propia

2.8.2.15 The Scientific World Journal

Tabla 20 Extracción Fuente 17

Titulo	Introduction to Information Retrieval
Publicación	The Scientific World Journal
Autores	Samimi, P., & Ravana, S. D.
Referencia	Samimi, P., & Ravana, S. D. (2014). Creation of Reliable Relevance Judgments in Information Retrieval Systems Evaluation Experimentation through Crowdsourcing: A Review. The Scientific World Journal, 2014, 135641. https://doi.org/10.1155/2014/135641
Descripción	
Área	Recuperación de Información, Máquinas de búsqueda, Relevancia
Resumen	La recopilación de pruebas se utiliza para evaluar los sistemas de recuperación de información en la experimentación de evaluación basada en laboratorio. En un entorno clásico, la generación de juicios de relevancia involucra evaluadores humanos y es una tarea costosa y que requiere mucho tiempo. Los investigadores y los profesionales aún enfrentan el desafío de realizar una evaluación confiable y de bajo costo de los sistemas de recuperación. El crowdsourcing como método novedoso de adquisición de datos se utiliza ampliamente en muchos campos de investigación. Se ha demostrado que el

	crowdsourcing es una solución rápida y económica, así como una alternativa confiable para crear juicios de relevancia. Una de las aplicaciones de crowdsourcing en IR es juzgar la relevancia del par de documentos de consulta. Para tener un experimento de crowdsourcing exitoso, las tareas de juicio de relevancia deben diseñarse precisamente para enfatizar el control de calidad. Este artículo tiene como objetivo explorar diferentes factores que influyen en la precisión de los juicios de relevancia realizados por los trabajadores y cómo intensificar la confiabilidad de los juicios en el experimento de crowdsourcing.
Aspectos Que Destacar Habla sobre el crowdsourcing para los juicios de relevancia en la evaluación de sistemas de recuperación de información	

Fuente: Elaboración propia

2.8.2.16 Literatura técnica

Como parte de la implementación de los operadores en la presente investigación, es necesario comprender la

2.8.2.17 Documentación técnica

Tabla 21 Extracción Fuente 18

Titulo	Lucene 9.2.0 Core API
Publicación	Apache Software Foundation

Autores	Apache Software Foundation
Referencia	(Apache Software Foundation, 2022c) Apache Software Foundation. (2022). Overview (Lucene 9.2.0 core API). https://lucene.apache.org/core/9_2_0/core/index.html
Descripción	Documentación de la biblioteca Lucene
Área	Lucene, Implementación, Similaridad
Resumen	Punto de entrada a la documentación técnica de Lucene, la biblioteca de código fuente abierto de recuperación de información utilizada por máquinas de búsqueda como Solr, Elasticsearch, y otras.
Aspectos Que Destacar Contiene los detalles de implementación de la biblioteca Lucene. La mejor referencia para la implementación de clases que interactúan con Lucene	

Fuente: Elaboración propia

Tabla 22 Extracción fuente 19

Titulo	Package org.apache.lucene.search.similarities
Publicación	Apache Software Foundation
Autores	Apache Software Foundation
Referencia	(Apache Software Foundation, 2022b) Apache Software Foundation. (2022).

	Org.apache.lucene.search.similarities (Lucene 9.2.0 core API). https://lucene.apache.org/core/9_2_0/core/org/apache/lucene/search/similarities/package-summary.html#package.description
Descripción	Documentación de la implementación de la similaridad entre documentos de la biblioteca Lucene
Área	Lucene, Implementación, Similaridad
Resumen	Detalles sobre la implementación de la similaridad entre documentos de Lucene, la biblioteca de código fuente abierto de recuperación de información utilizada por máquinas de búsqueda como Solr, Elasticsearch, y otras.
Aspectos Que Destacar Detalle de la implementación de la similaridad (diferentes tipos de similaridad) en la biblioteca Lucene.	

Fuente: Elaboración propia

Tabla 23 Extracción fuente 20

Titulo	Lucene - Core - ASF JIRA
Publicación	Apache Software Foundation
Autores	Apache Software Foundation
Referencia	(Apache Software Foundation, 2022a)
Descripción	Sitio de manejo de proyecto de Apache Lucene

Área	Lucene, Implementación, Similaridad
Resumen	Sitio del proyecto Lucene, donde se define la funcionalidad a desarrollar en el proyecto a través de tiquetes utilizando el sistema de administración de proyecto Jira.
Aspectos Que Destacar Contiene las discusiones técnicas de la implementación de diversas funcionalidades de Apache Lucene. Por ejemplo, en este tiquete https://issues.apache.org/jira/browse/LUCENE-2091 se tiene la discusión de la implementación del algoritmo de *similaridad y ranking* Okapi BM25 que actualmente es el que se usa por defecto en Lucene.	

Fuente: Elaboración propia

2.8.2.18 Libros de referencia

Pese a que la documentación más reciente sobre la plataforma tecnológica a utilizar, Lucene y Elasticsearch, se encuentra disponible en línea, existen algunos libros de referencia que se consideran importantes para dar un entendimiento sobre cómo funcionan estas plataformas.

Tabla 24 Extracción Fuente 21

Titulo	Lucene in Action, Second Edition
Publicación	Manning Publications Co
Autores	McCandless, M., Hatcher, E., & Gospodnetić, O.
Referencia	McCandless, M., Hatcher, E., & Gospodnetić, O. (2010). Lucene in Action, Second Edition. Manning Publications Co.
Descripción	Describe el uso y la implementación de Lucene
Área	Recuperación de Información, Lucene

Resumen	Cuando apareció Lucene por primera vez, este motor de búsqueda ultrarrápido era increíble. Su API de alto rendimiento y fácil de usar, funciones como campos numéricos, cargas útiles, búsqueda casi en tiempo real y grandes aumentos en la indexación y la velocidad de búsqueda la convierten en la herramienta de búsqueda líder.
Aspectos Que Destacar Describe el uso y la implementación de Lucene	

Fuente: Elaboración propia

Tabla 25 Extracción fuente 22

Titulo	Taming Text
Publicación	Manning Publications Co
Autores	Ingersoll, G. S., Morton, T. S., & Farris, A. L.
Referencia	(Ingersoll et al., 2013) Ingersoll, G. S., Morton, T. S., & Farris, A. L. (2013). Taming Text. https://www.manning.com/books/taming-text
Descripción	Describe la teoría de cómo implementar máquinas de búsqueda con ejemplos prácticos.
Área	Recuperación de Información, Lucene, Máquinas de Búsqueda

Resumen	Taming Text es una guía práctica basada en ejemplos para trabajar con texto no estructurado en el contexto de aplicaciones del mundo real. Este libro explora cómo organizar automáticamente el texto utilizando enfoques como la búsqueda de texto completo, el reconocimiento de nombres propios, la agrupación, el etiquetado, la extracción de información y el resumen. El libro guía a través de ejemplos que ilustran cada uno de estos temas, así como los cimientos sobre los cuales se construyen.
Aspectos Que Destacar Si bien utiliza Solr para la implementación de los ejemplos, muchos de estos son útiles para entender la implementación y lo algoritmos de máquinas de búsqueda. Los capítulos 3 (*Searching)* y 4 (*Fuzzy String Matching*) van a ser particularmente importantes para ayudar a la implementación del proyecto.	

Fuente: Elaboración propia

2.8.3 Calidad de fuentes de información

La calidad de estos artículos es aceptable en su mayoría, no solo por ser elaborados por autores altamente conocidos o en revistas altamente conocidas, sino que adicional a esto se realizó una revisión de calidad de la revista y los artículos en su mayoría utilizando el sitio www.scimagojr.com, el cual ayuda, ya que brinda información de los H índices de las revistas vistas, por ende se buscaron aquellas con H índices elevados, lo cual permite dar una certeza de la calidad de las publicaciones.

A continuación, se detalla la información dada para cada una de las revistas que aparecen en la revisión de literatura:

2.8.4 Journal of Documentation

El mismo posee las siguientes características:

- H-Índex de 67
- Se encuentra en el Cuartil 1
- Posee una citación de documentos por arriba del 2.1

Siendo de una revista de un rango bastante alto dando una certeza de su calidad.

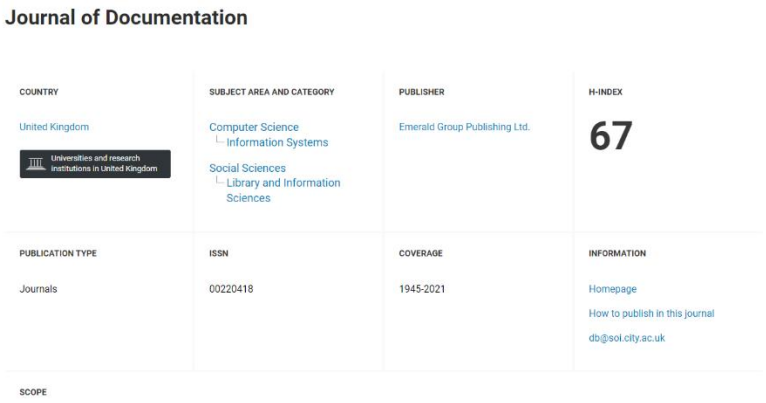


Ilustración 1 Validación de la calidad de la fuente Journal of Documentation.

Fuente: Tomado del sitio Scimagojr

2.8.5 International Journal on Digital Libraries

El mismo posee las siguientes características:

- H-Índex de 34
- Se encuentra en el Cuartil 2
- Posee una citación de documentos de 1.6

Si bien estos números podrían ser aún mejores, son aceptables para determinar que sus artículos también son realmente de calidad.

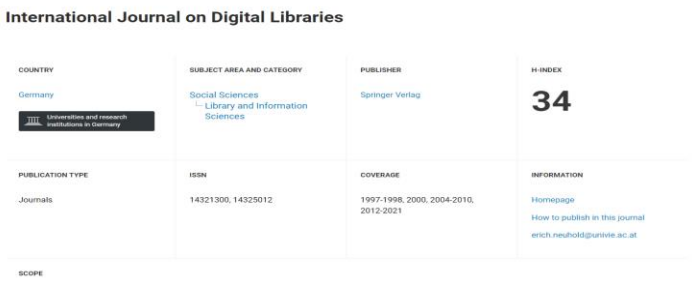


Ilustración 2 Validación de la calidad de la fuente International Journal on Digital Libraries

Fuente: Tomado del sitio Scimagojr

2.8.6 International Journal of Medical System

El mismo posee las siguientes características:

- H-Índex de 89
- Se encuentra en el Cuartil 1
- Posee una citación de documentos por arriba del 6.1

Estos números dejan ver que la revista es de una alta calidad, ya que sus números están por encima del promedio de revistas ya con números altos.

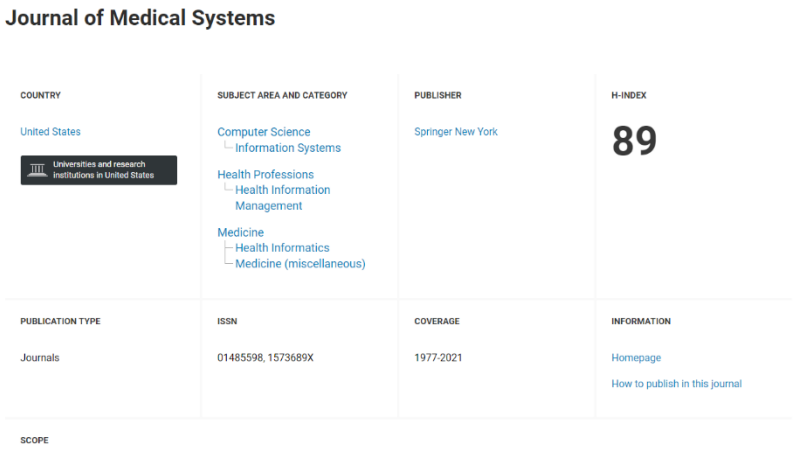


Ilustración 3 Validación de la calidad de la fuente Journal of Medical Systems.

Fuente: Tomado del sitio Scimagojr

2.8.7 Communications of the ACM

Este posee las siguientes características:

- H-Índex de 220
- Se encuentra en el Cuartil 1
- Posee una citación de documentos por arriba del 5.6

Dado este número se puede determinar que es una de las revistas con mejores números, lo que da certeza de su calidad.

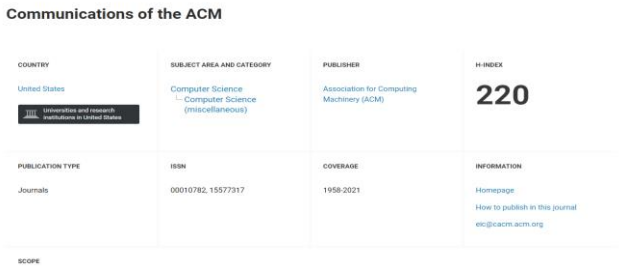


Ilustración 4 Validación de la calidad de la fuente Communications of the AMC

Fuente: Tomado del sitio Scimagojr

2.8.8 IEEE Data Engineering Bulletin

Si bien este no se evaluó utilizando el sitio www.scimagojr.com, se puede determinar que es una fuente de calidad, dado que según Google Scholar, el mismo ha sido citado más de 2250 veces, por ende, se intuye que ha sido de gran relevancia para otros trabajos, dando esto una certeza de que su contenido es de calidad.

2.8.9 Foundations and Trends in Information Retrieval

El mismo posee las siguientes características:

- H-Índex de 34
- Se encuentra en el Cuartil 1
- Posee una citación de documentos de 18.8

Si bien estos números podrían ser aún mejores, son aceptables para determinar que sus artículos también son realmente de calidad y también se toma en cuenta a los autores del estudio.

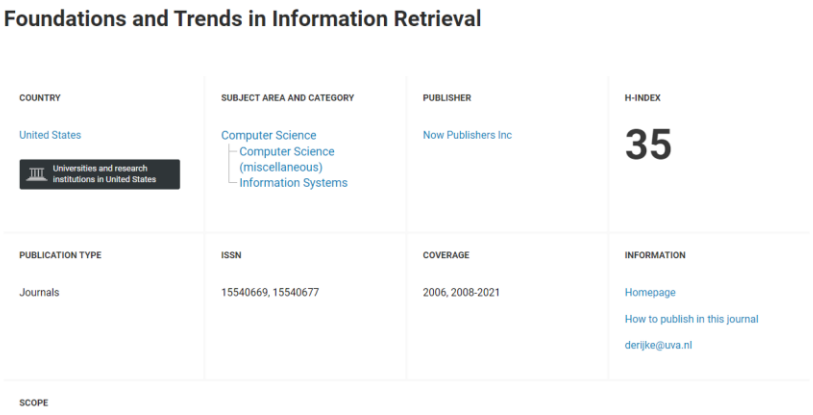


Ilustración 5 Validación de la calidad de la fuente Foundations and Trends in Information Retrieval

Fuente: Tomado del sitio Scimagojr

2.8.10 Cambridge University Press

Si bien esta fuente no se evaluó utilizando el sitio mencionado con anterioridad, para la evaluación de la calidad de las fuentes, se determina que esta fuente es de calidad, ya que según los datos obtenidos de Google Scholar, la misma posee una citación mayor a las 24600 citas, por ende, se puede inferir que el mismo ha sido de alta relevancia para otras investigaciones similares por lo que se considera una buena fuente de información.

2.8.11 Text REtrieval Conference (TREC) & International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS) & Proceedings of the 7th conference on Information technology education

Por otro lado, se da certeza de la calidad de las conferencias de Text REtrieval Conference (TREC), dada la importancia que han adquirido las mismas en el campo de la recuperación de información, ya que es una de las conferencias más relevantes del campo a nivel mundial, además de que sin estas conferencias los usuarios hubieran pasado una gran cantidad de horas haciendo

búsquedas adicionales sobre la recuperación de información según (Rowe et al., 2010), dándole esto un valor de alta calidad a cada uno de los temas expuestos en la misma. De la misma manera, se da certeza de la calidad del contenido de las conferencias de International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS) y Proceedings of the 7th conference on Information technology education, dado el número de conferencias que se han realizado, así como la calidad de los expositores en las mismas.

2.8.12 United States Patent

Dado que uno de los artículos de interés para la investigación es una patente, la misma se considera de calidad, ya que la misma es dada por un ente gubernamental, además de que, para patentar algún proceso, modelo, entre otros se debe cumplir con varios requerimientos y es revisado y analizado por diferentes profesionales, por lo que el contenido de esta se puede considerar de calidad.

2.8.13 Journal of American Society for Information Science and Technology

El mismo posee las siguientes características:

- H-Índex de 18
- Es citado más de 90 veces

Estos datos fueron obtenidos desde el sitio www.scijournal.org, el cual ayuda a evaluar la calidad de los artículos y revistas, dados estos números, a pesar de ser bajos se considera una fuente de calidad.

Ilustración 6 Validación de la calidad de la fuente Journal of the American Society for Information Science and Technology

Fuente: Tomado del sitio Scijournal

2.8.14 VFAST Transactions on Software Enginnering

El mismo posee las siguientes características:

- H-Índex de 6
- Es citado más de 167 veces

Estos datos fueron obtenidos desde Google Scholar, el cual da información sobre el H-índex de los artículos, gracias a esto podemos determinar que, aunque el mismo es bajo, todavía se considera una fuente de calidad.

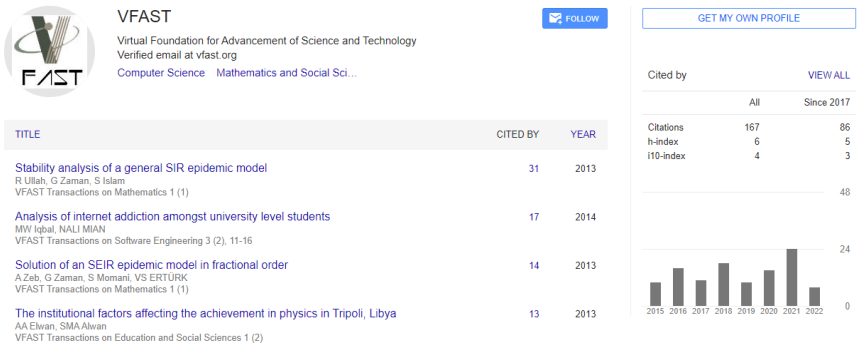


Ilustración 7 Validación de la calidad de la fuente VFAST Transactions on Software Enginnering

Fuente: Google Scholar

2.8.15 The Scientific World Journal

El mismo posee las siguientes características:

- H-Índex de 103
- Se encuentra en el Cuartil 3
- Posee una citación de documentos de 2.2

Dado este número en su H-Índex y aunque el mismo se encuentra en el cuartil 3, se puede determinar que es una de las revistas que da certeza de su calidad.

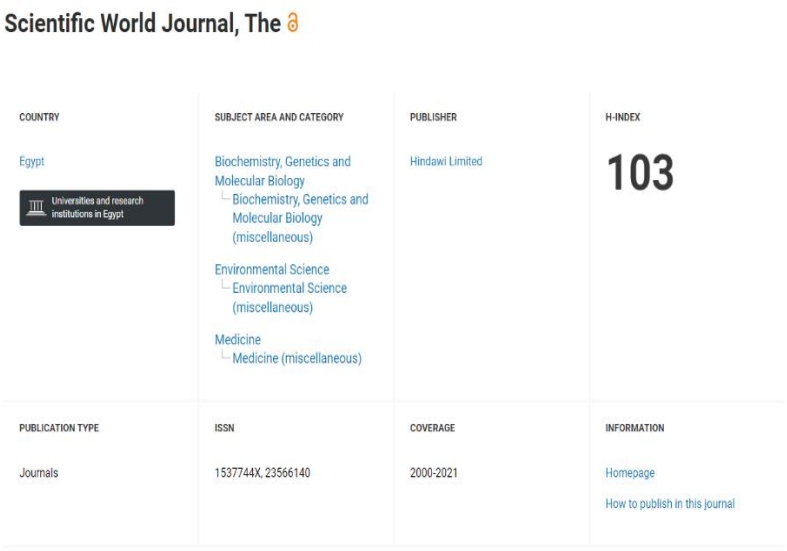


Ilustración 8 Validación de la calidad de la fuente The Scientific World Journal

Fuente: Tomado del sitio Scimagojr

2.8.16 Resumen de Resultados

Al final, la revisión de literatura analizó alrededor de 16 estudios incluidos en revistas, conferencias y libros, de estos, todos fueron considerados de importancia para la investigación, lo anterior basado en el contenido de estos, no solo por sus años de publicación y la calidad de los autores y revistas en las que se encontraron, sino también por el contenido valioso sobre el tema en cuestión.

Tabla 26 Resultados de Literatura

Fuentes	Cantidad de Estudios	Estudios Relevantes
Journal of Documentation	2	2
International Journal on Digital Libraries	1	1
Journal of Medical Systems	1	1
Journal of the American Society for Information Science and Technology	1	1
AMC Digital Library	2	2
IEEE	1	1
Text REtrieval Conference (TREC)	2	2
International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS)	1	1
United States Patent	1	1

VFAST Transactions on Software Enginnering	1	1
Foundations and Trends in Information Retrieval	1	1
Cambridge University Press	1	1
Proceedings of the 7th cnference on Information technology education	1	1
The Scientific World Journal	1	1

Fuente: Elaboración propia

3 Marco Conceptual

Para identificar los conceptos más importantes y facilitar la elaboración del marco conceptual se elaboró la siguiente nube de palabras. Para la generación de esta nube de palabras se utilizó el texto del artículo de referencia (Singhal & Google, 2001). La inclusión de estos se debe a que el primero describe la patente que se desea implementar como parte del presente proyecto y el siguiente contiene una revisión y resumen de la literatura del campo de la recuperación de información que a criterio de los investigadores cubre los conceptos más relevantes.

Ilustración 9 Nube de Palabras - Conceptos de Recuperación de Información

Fuente: Elaboración propia

La nube de palabras fue generada utilizando el sitio https://monkeylearn.com/word-cloud/.

La siguiente sección presenta un resumen de los conceptos mencionados, así como la definición de estos. Estos se presentan en un orden de más general a más específico para que las definiciones existan dentro de un concepto.

3.1 Conceptos de la Recuperación de Información

A continuación, se presentan los conceptos base del campo de la recuperación de información:

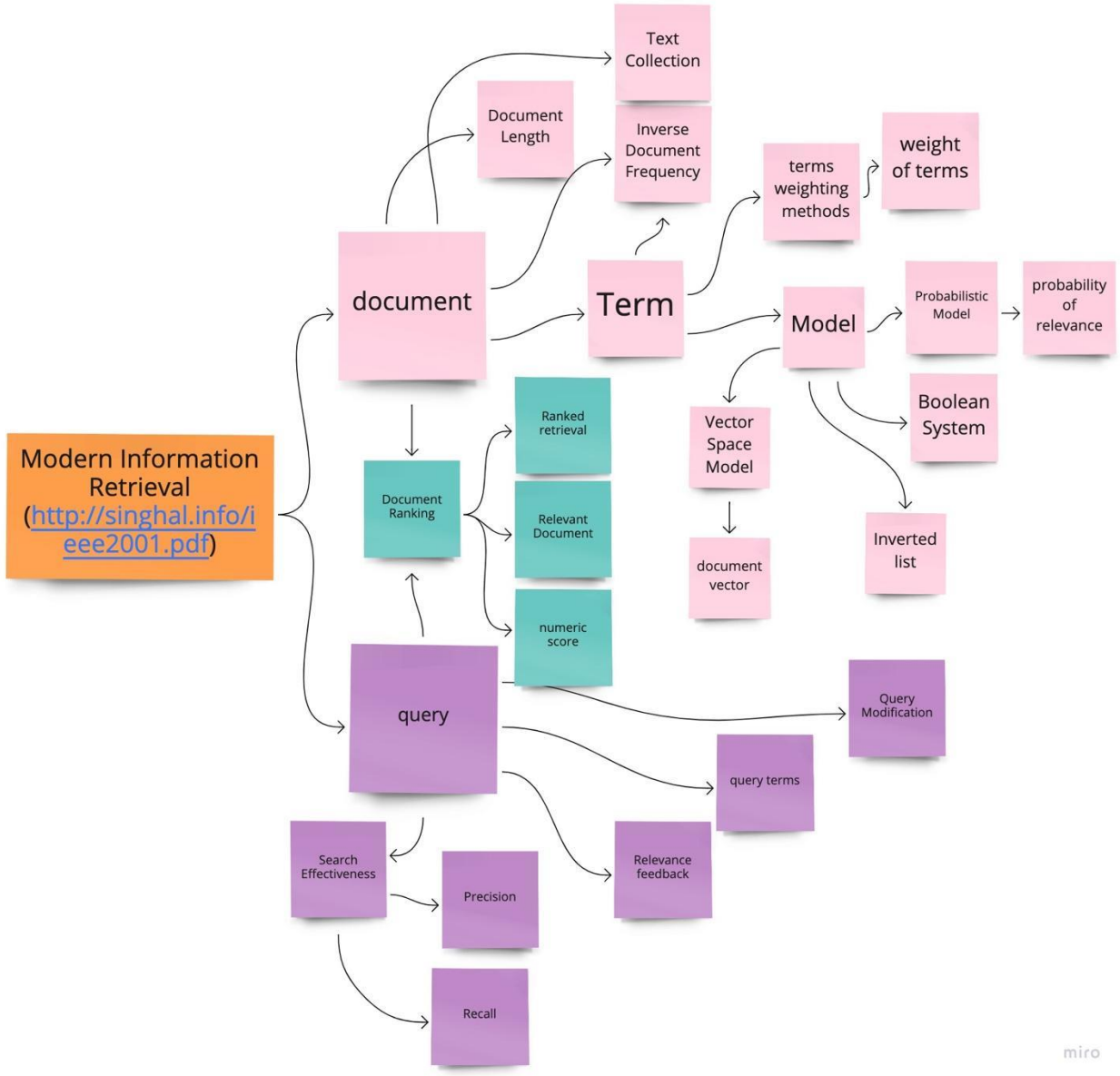


Ilustración 10 Conceptos Generales de la Recuperación de Información

Fuente: Elaboración propia

La ***Recuperación de Información*** es la disciplina que trata con buscar materiales (usualmente documentos) de naturaleza no estructurada (normalmente texto) que satisfacen una necesidad de información de una colección grande (normalmente almacenada en computadoras). (Manning et al., 2008).

Dos conceptos centrales a la disciplina son los documentos y las consultas (query). Los ***documentos*** son las unidades almacenadas en un sistema de recuperación de información que son el objetivo de la búsqueda, y se componen de palabras o términos (***terms***). Las ***consultas*** son la forma por la cual el usuario del sistema expresa la necesidad de información, normalmente por medio de palabras clave o frases que describen esta necesidad. Estas palabras o frases se denominan *términos* (***terms***), tanto dentro de los documentos como en la consulta.

Dentro del sistema de recuperación de información existe el ***modelo*** que modela los documentos que pertenecen a una ***colección de texto***, también conocida como ***corpus*** o como el ***índice***. Estos modelos en general contienen un ***índice o lista invertidos*** que tiene una lista de términos y apunta a los documentos que tienen estos términos. Los documentos como tales se representan comúnmente como un vector de términos (***document vector***), el cual se utiliza posteriormente para realizar las operaciones de búsqueda y relevancia.

Estos modelos pueden ser de diferente tipo:

- ***Modelo Booleano***: Es el primer modelo que se utilizó, en donde se buscan los términos en el índice y se considera que un documento debe ser retornado en la búsqueda si el término existe o no (junto con la ayuda de algunos operadores booleanos como *AND, OR*, …) (Harman, 2019)
- ***Modelo Probabilístico***: Los modelos probabilísticos son aquellos que se basan en la probabilidad que un documento satisfaga la necesidad de información de un usuario, o, en otras palabras, la probabilidad de relevancia (***probability of relevance***). Introducidos en (ROBERTSON, 1977)

- ***Modelo de Espacio Vectorial*** (***Vector Space Model***): Este tipo de modelo representa cada documento, así como cada consulta por un vector, por ende, el modelo lo que busca hacer es buscar en función de la similitud de los vectores de los documentos, así como de la consulta, cuales son aquellos documentos más relevantes. Introducido por (Salton et al., 1975)

3.1.1 Modelo de Espacio Vectorial (VSM)

Dentro del modelo de espacio vectorial existen dos conceptos base:

- ***Term Frequency*** o Frecuencia del Término: Se refiere a cuantas veces aparece el término dentro de un documento.
- ***Inverse Document Frequency*** o Frecuencia Inversa del Documento: Se refiere al número de documentos en los que aparece el término.

Estos dos conceptos se combinan en una medida conocida como ***TF/IDF***, para determinar un puntaje para cada documento (***numeric score***) con respecto a la consulta para determinar el rango del documento (***document ranking***), es decir, su posición dentro de los resultados (***ranked retrieval***) y, por lo tanto, determinar qué tan relevante es para la consulta (***relevant document***). Los términos pueden tener diferentes pesos (***weight of terms***) dependiendo de diferentes métodos. La descripción del cálculo de TF/IDF se elabora en la siguiente sección del Marco Conceptual. (Manning et al., 2008)

Con respecto a la consulta como tal, esta consiste igualmente de términos (***query terms***) que se utilizan para realizar la búsqueda. Al ejecutar una búsqueda, normalmente esta se modifica (***query modifications***) para tratar de mejorar los resultados que puede producir. Estas modificaciones pueden incluir transformaciones de los términos por medio de tematización o derivación (***stemming***), inclusión de sinónimos, eliminación de palabras de parada (***stop words***), y otras.

3.1.2 Medición de resultados

Para medir la efectividad de los resultados de las búsquedas (***search effectiveness***) existen dos medidas principales:

- Precisión (***precision***): El número de documentos retornados en una consulta que son relevantes para la necesidad de información de esta, divido por el número de documentos retornados en total para la consulta.
- Índice de Recuperación (***recall)***: El número de documentos relevantes para la consulta que fueron retornados por la consulta, dividido por el número de documentos relevantes para la consulta en la colección o índice.
- Puntuación F1 (***F-score***): Es la media harmónica de la precisión y el índice de recuperación, calculada de la siguiente forma:

$$F1 = 2 * \frac{Precision * Recal}{Precision + Recall}$$

Ilustración 11 Fórmula del puntaje F1

Fuente: (Marwah & Beel, 2020)

- *Normalized Discounted Cumulative Gain*: Es una medida en la cual se evalúa la relevancia de los documentos de acuerdo con su posición en los resultados – se parte de la premisa que documentos relevantes que aparecen en posiciones más bajas de los resultados son penalizados por una escala de relevancia logarítmica. La escala va de 0 a 1, donde 1 se considera que todos los resultados son relevantes en las primeras posiciones. La fórmula del *Discounted Cumulative Gain* es la siguiente:

$$DCG_p = \sum_{i=1}^{p} rel_i / \log_2(i+1)$$

Ilustración 12 Fórmula del puntaje DCG

Fuente: (Marwah & Beel, 2020)

Con:

$$i = posición\ en\ los\ resultados rel_i = relevancia\ del\ resultado\ en\ la\ posición\ ip = resultados\ a\ evaluar$$

Para normalizar el *DCG*, se utiliza un DCGS ideal en la posición *p*, y se normaliza de la siguiente manera:

$$nDCG_p = DCG_p / IDCG_p$$

Ilustración 13 Fórmula del Puntaje nDCG

Fuente: (Marwah & Beel, 2020)

Estas dos medidas normalmente son inversamente proporcionales – al incrementar una se reduce la otra. Esto se puede entender dado que, al restringir la consulta para que solamente traiga documentos que se tiene seguridad que son relevantes (aumentar la precisión), se pueden dejar por fuera documentos que también lo son (reduciendo el índice de recuperación), y viceversa, al flexibilizar las condiciones de la consulta para asegurarse que retorne todos los documentos relevantes (incrementar el índice de recuperación) es posible que se incluyan documentos no relevantes (reduciendo la precisión). (Manning et al., 2008).

3.1.3 Explicación de los resultados

La capacidad de explicar los resultados de una búsqueda es un problema importante en el ámbito de la recuperación de información. Esto se da sobre todo en ambientes profesionales, como por

ejemplo en los campos legales, de salud, reclutamiento, y otros, donde es importante tener un alto índice de recuperación, pero a la vez el orden de los resultados es importante. En estos ambientes no incluir un resultado relevante puede traer consecuencias como perder un caso legal, no contratar a un candidato indicado, o invalidar una investigación. Por esto es importante poder explicar por qué los resultados de una búsqueda son obtenidos (Russell-Rose & MacFarlane, 2020).

Los principios que debe tener un sistema para poder explicar los resultados y ofrecer transparencia sobre los mismos son: (Russell-Rose & MacFarlane, 2020; Russell-Rose & Shokraneh, 2020)

1. Soportar transparencia entre las estructuras lógicas y físicas de los datos
2. Adaptar formalismos escalables para acomodar la complejidad de la búsqueda
3. Delegar operaciones sintácticas de bajo nivel al sistema
4. Proveer retroalimentación en tiempo real de la efectividad de la consulta (Scells & Zuccon, 2018)
5. Proveer facilidades para colaborar en equipo.

3.2 Conceptos de Lucene y Elasticsearch

Para la implementación de los operadores deseada en este estudio es necesario elaborar la plataforma tecnológica sobre la cual se realizará la implementación.

Lucene es una API de código libre bajo la licencia de Apache, la cual se utiliza para la recuperación de información, esta es, por su utilidad y su implementación, la más utilizada en motores de búsqueda, ya que la misma es útil para aquellas aplicaciones que requieran indexado o bien realizar búsqueda de texto, por ende, Lucene va más allá de búsquedas en una base datos. Esta se compone de dos grandes procesos para realizar búsquedas las cuales son: (Apache Software Foundation, 2022b)

Indexación: es el proceso mediante el cual se crea un índice a toda la información de un documento relevante lo que va a permitir un acceso rápido a la información. Para realizar este

proceso, Lucene hace uso de LuceneCreateIndexApp la cual utiliza diversas clases para realizar esta indexación como:

- IndexWrite: el corazón del proceso se encarga de encapsular el índice creado.
- Directory: sirve para representar la ubicación del índice, ya sea en la memoria RAM, archivo, entre otros.
- Analizer: filtra la información importante con la que se va a crear el índice y se encarga de eliminar el resto, como las palabras cortas.
- Field: guarda información de cada documento que se quiere guardar en el Document.
- Document: se encarga de almacenar los campos (Field) con la metainformación que se desea indexar.

Búsqueda: este proceso consiste en consultar el índice para obtener resultados donde son similares o iguales las palabras o la expresión utilizada en la consulta. En este proceso se hace uso de LucenSearchApp la cual utiliza algunas clases para realizar el proceso de búsqueda como:

- IndexSearch: a través de este se realizan las consultas (query) que van a devolver los resultados. Esta consulta es un conjunto de términos (Term).
- Term: son las condiciones de la búsqueda.
- Query: permite que se defina la consulta sobre la que se desea realizar la búsqueda.
- Hits: posee los documentos (Document) que cumplen las condiciones de la búsqueda realizada.

Lucene, aparte de estos dos procesos, realiza un ranking de los resultados de la búsqueda, es decir una calificación donde ubica en orden los documentos según su relevancia o bien su importancia, para esto Lucene hace uso del algoritmo TFIDFSimilarity, el cual se encarga de realizar este proceso.

El TFIDFSimilarity es el encargado de definir los puntajes (scoring) que son los encargados de mostrar la relevancia de los documentos, Lucene combina el modelo Booleano (BM) con el modelo espacial vectorial (VSM), es decir, que los modelos calificados como “aprobados” por BM posteriormente son calificados por VSM (Apache Software Foundation, 2022a).

Los documentos y consultas en el VSM son representados por el peso de los vectores en un espacio multidimensional y cada término de índice que sea distinto se considera una dimensión y los pesos son valores TF-IDF. Si bien el VSM no necesita que estos pesos sean TF-IDF, esto quiere decir que para el término t en un documento (document) x, el TF(t,x) varía según el número de ocurrencias del término t en x, es decir, cuando uno aumenta también lo hace el otro y para id(t) hace una variación de manera similar, pero con el inverso del número de documentos de índices que contiene a t. (Apache Software Foundation, 2022a; Salton et al., 1975).

Para obtener la puntación VSM de un documento "d" a través de una consulta "q" se obtiene a través de la fórmula de la similitud del coseno.

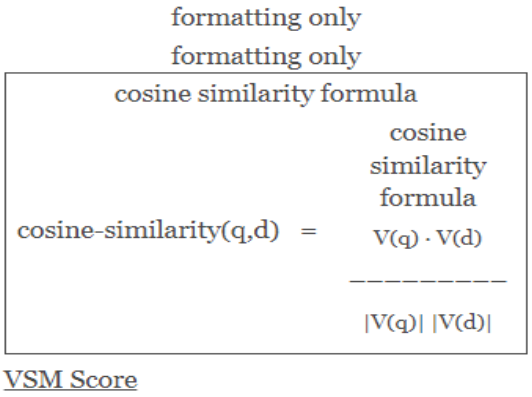


Ilustración 14 Fórmula de la similitud del coseno

Fuente: (Apache Software Foundation, 2022a).

Este puntaje obtenido (VSM) es refinado por Lucene tanto para la calidad de la búsqueda, es decir, que esta sea lo más asertiva posible, como la usabilidad de está buscando obtener el puntaje conceptual de Lucene, esto se realiza a través de la normalización de longitud de documentos diferentes, la cual normaliza un vector similar o de mayor tamaño al vector unitario doc-len-norm(d). Por otro lado, en la etapa de indexación, permite a los usuarios determinar que ciertos documentos son más importantes o bien más relevantes que otros, esto se hace asignándole un impulso al documento doc-boost(d), para esto la puntuación del documento se

multiplica por este valor de impulso dándole así un valor más alto en cuanto a la relevancia de este.

Lucene funciona con base en campos, por ende, cada término de consulta es aplicado a uno solo de estos campos, la normalización de la longitud del documento, mencionada con anterioridad, es por la longitud del campo y además influye el aumento de la cantidad de documentos. El mismo campo se puede agregar varias veces durante el proceso de indexación, dicho aumento del campo en el documento es la multiplicación de los aumentos de las partes de este campo dentro del mismo.

Al momento de la búsqueda los usuarios pueden especificar los impulsos para cada una de las consultas o subconsultas, así como especificar cada término de la misma, por lo que la contribución de un término de consulta se multiplica por el impulso de ese término de consulta query-boost(q) a la puntuación del documento. Un documento puede coincidir con otra consulta de varios términos sin la necesidad de tener todos los términos de dicha consulta, aunque esto solo aplica para algunas consultas.

Dado lo mencionado anteriormente y bajo el supuesto de un solo campo en el índice, se tiene la siguiente fórmula, la cual indica la puntuación conceptual de Lucene:

$$\text{score}(q,d) = \text{query-boost}(q) \cdot \underbrace{\frac{V(q) \cdot V(d)}{|V(q)|}}_{\text{formatting only Lucene conceptual scoring formula}} \cdot \text{doc-len-norm}(d) \cdot \text{doc-boost}(d)$$

Ilustración 15 Fórmula de la puntuación conceptual de Lucene

Fuente: (Apache Software Foundation, 2022a).

Dicha fórmula, de manera simplificada, indica que los términos y los documentos se presentan en el campo y que los aumentos son por término de consulta y no por la consulta como tal.

Una vez obtenida esta puntuación conceptual, Lucene busca derivar de ella la función de puntuación práctica, para realizar dicho cálculo se calculan algunos componentes y se agregan por adelantado. Algunos de estos cálculos son:

- El impulso de consulta para cada término de la consulta, este se debe conocer cuando se comienza la búsqueda.
- La consulta de la norma euclidiana |V(q)| la cual se calcula al comienzo de la búsqueda, esto ya que es totalmente independiente del documento que se está calificando. Esta optimización, o bien esta normalización, se realiza por dos buenas razones:
    - Las puntuaciones de un documento para dos consultas diferentes tienen que ser comparables y esto es lo que permite esa comparabilidad de dos o más consultas realizar la normalización en el vector de consulta V(q).
    - La similitud de coseno permite encontrar que tan similares son dos documentos, esto se puede utilizar con Lucene para agrupar en clústeres y poder utilizar un documento como consulta para calcular la similitud con otros documentos.
- Se calcula la multiplicación de la longitud del documento doc-len-norm(d) por el impulso del documento doc-boost(d) y este valor se guarda en norm(d) como valor único.

Una vez con los cálculos necesarios realizados se aplica la siguiente fórmula para obtener la función de puntuación práctica.

Lucene conceptual scoring formula

$$\text{score}(q,d) = \sum_{t \text{ in } q} \left(\text{tf}(t \text{ in } d) \cdot \text{idf}(t)^2 \cdot t.\text{getBoost}() \cdot \text{norm}(t,d) \right)$$

Ilustración 16 Fórmula de la puntuación práctica de Lucene

Fuente: (Apache Software Foundation, 2022c).

De dicha fórmula se encuentra lo siguiente:

- TF (t in d): es definida como la frecuencia de término, es decir, el número de veces que un término “t” aparece en un documento puntuado actualmente “d”. Para aquellos

documentos con mayor frecuencia en un término se les da una puntuación más alta. El cálculo para obtener dicha frecuencia sería el siguiente:

term frequency computation

$$tf(t \text{ in } d) = \text{frequency}^{1/2}$$

Ilustración 17 Fórmula de la Frecuencia del Término

Fuente: (Apache Software Foundation, 2022c).

- IDF (t): la cual se entiende como la frecuencia de documento inversa, la cual indica el número de documentos en los que aparece el término "t". Esto permite que los términos más raros den un mayor valor a la puntuación total. El cálculo se realiza de la siguiente manera:

inverse document frequency computation

$$idf(t) = 1 + \log\left(\frac{\text{inverse document frequency computation}}{\frac{\text{docCount}+1}{\text{docFreq}+1}}\right)$$

Ilustración 18 Fórmula de la Frecuencia del documento inversa

Fuente: (Apache Software Foundation, 2022c).

- t. getBoost (): es un aumento del tiempo de búsqueda de un término “t” en una consulta “q” como fue especificado en el impulso del término.
- norm (t, d): es el factor de impulso de tiempo del índice que depende de la cantidad de tokens del campo en el documento, por lo que los campos cortos otorgan más a la puntuación final.

De esta manera, Lucene, utilizando el TFIDF, permite asignar los puntajes de relevancia y posteriormente crear el ranking de resultados según la consulta, los términos usados en dicha consulta y el puntaje obtenido a través de los procesos y fórmulas ya descritas.

En esencia, TFIDF utiliza dos factores claves: la frecuencia del término y la frecuencia del documento inversa para poder determinar si un documento es similar a la consulta realizada, esto además le permite al TFIDF medir la concentración de un término “t” en un documento “d”. Por lo que, si un término es común en el documento, pero es muy raro en otros el TFIDF le asignará una puntuación alta, por ende, el TFIDF lo considera muy relevante para la consulta.

3.3 BM25

Además del TFIDF existe otro algoritmo, el cual mejora a este TFIDF en cierta medida, al mostrar la relevancia más como un problema de probabilidad, a este algoritmo se le conoce como el BM25. Este algoritmo fue introducido en 1994 en la Tercer Conferencia de Recuperación de Textos (TREC) por Stephen E. Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu y Mike Gatford. (Robertson et al., 1995)

EL BM25(Best Match 25) busca clasificar los documentos en el corpus según la relevancia del documento para una consulta en concreto. La fórmula que utiliza para realizar este cálculo de relevancia es la siguiente:

$$\mathrm{BM25}(D,Q) = \sum_{i=1}^{n} IDF(q_i, D) \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i) + k_1 \cdot (1 - b + b \cdot |D|/d_{avg}))}$$

Ilustración 19 Fórmula del BM25

Fuente: (Apache Software Foundation, 2022a).

En donde:

- F (q_i, D): representa el número de veces donde el término qi esta o parece en el documento D.
- |D|: es el número de palabras en el documento D.
- d_{avg}: es el promedio de palabras en el documento.
- b y k_1 son los híper parámetros para el BM25.
    - b: controla la longitud del documento el cual debe normalizar los valores de la frecuencia de los términos y estos deben estar en un rango de 0 y 1, donde el valor 0 inhabilita la normalización de la longitud. El valor predeterminado del mismo usualmente es de 0.75.
    - k_1: este híper parámetro calibra la saturación de la frecuencia del término y este posee valores nulos y positivos, donde un valor de 0 hace que la frecuencia de los términos sea ignorada por completo y donde los valores de k1 altos aumentan el impacto de la frecuencia de los términos en la puntación. El valor predeterminado es 1.2.
- IDF (q_i, D): representa la frecuencia inversa de un documento.

Ya entendiendo que significa cada parámetro de la fórmula, se explica qué realiza cada una de las partes de esta.

f (qi, D) y k1. f (qi, D)

Esta parte busca calcular cuantas veces aparece un término en un documento, mientras más veces aparezca mayor puntuación tendrá dicho documento. Además, aquí se observa la importancia del híper parámetro K1, el cual, según lo mencionado anteriormente, determina la saturación de frecuencia en un término, por ende, mientras más grande sea este valor, más lenta va a ser la saturación, es decir, que si los términos aparecen más veces le agregan un valor adicional a la puntuación.

|D|/davg

Esta sección, ubicada en el denominador, permite controlar que cuantos más términos de la consulta no coincidan con un documento, menor será la puntuación asignada a dicho documento, es decir, que si un documento de 200 páginas menciona el término de la consulta pocas veces es menos probable que dicho documento tenga que ver con la consulta realizada. Por otro lado, esto implica que si la consulta es realizada en un documento de poca extensión es mucho más rápido encontrar coincidencias de los términos de la consulta, el caso contrario ocurre en documentos de larga extensión donde se puede llevar un poco más de tiempo.

IDF (qi, D)

Para un número de N documentos dicha frecuencia inversa se calcula de la siguiente manera.

$$\mathrm{IDF}(q_i, D) = \log \frac{N - N(q_i) + 0.5}{N(q_i) + 0.5}$$

Ilustración 20 Fórmula de la frecuencia inversa

Fuente: (Apache Software Foundation, 2022a).

Donde N(qi) representa el número de documentos existentes en el corpus que tiene el término qi de la consulta.

Esta parte es muy similar a la del TFIDF, donde este buscaba darles un puntaje más alto a las palabras raras y que estas sumen más a la puntuación de relevancia final del documento. Si bien son parecidas la fórmula el IDF mencionada en el TFIDF posee un inconveniente, dado que cuando esta se utiliza con términos que salen más allá de la mitad del corpus, ya asigna un valor negativo por lo que afecta la puntuación final, para evitar esto se le agrega a la fórmula un 1 a la ecuación quedando de la siguiente manera.

$$\mathrm{IDF}(q_i) = \log \left(1 + \frac{N - N(q_i) + 0.5}{N(q_i) + 0.5}\right)$$

Ilustración 21 Fórmula de la frecuencia inversa mejorada

Fuente: (Apache Software Foundation, 2022a).

Esto hace que los valores negativos, dados por el IDF, se cambien por un valor positivo, denotado como “є”.

Ahora bien, con cada uno de los híper parámetros se suele usar un valor determinado, dando esto un buen punto de partida, pero cada uno de ellos puede ser configurado según la necesidad, pero siempre se tiene que medir el rendimiento de las configuraciones utilizadas, por lo que, para algunos parámetros como “k” y “b” es importante plantearse algunas preguntas de antemano para darles un valor. Por ejemplo, para k1 se puede empezar preguntando ¿cuándo creemos que es probable que se sature un término? Esto siempre tomando en cuenta el documento a consultar, ya que en documentos grandes es probable que un término aparezca varias veces, por lo que quizá no se busque que los términos se saturen muy rápido, por lo que se le asigna a k1 un valor mayor para evitar esto, de igual forma para casos contrarios donde el documento es de mucho más pequeño.

De igual forma pasa con el parámetro de “b” donde debería preguntarse ¿cuándo creemos que es probable que un documento sea muy largo y cuanto debería eso obstaculizar la relevancia para el término? Esto indica que si un documento habla sobre un tema específico es poco probable que la longitud sea perjudicial, por ende, colocarle un valor menor sería lo más adecuado, caso contrario de que el documento trate diversos temas, en este caso es más apropiado utilizar un valor mayor para el parámetro.

Con la configuración de los parámetros mencionados. y al buscar aplicar esta nueva fórmula que brinda el BM25, se puede abarcar el problema de la relevancia de un documento con base en los términos de la consulta, esto proyecta la relevancia como un problema de probabilidad, por lo que busca reflejar la probabilidad de que un usuario considere relevante el resultado de dicha búsqueda.

Si bien BM25 trae muchas mejoras en comparación al TFIDF, no es adecuado para todo, o bien para todos, ya que aún tiene algunas deficiencias respecto a números e imágenes y otras entidades donde la ganancia de utilizar el BM25 no está clara.

Aun así, tanto BM25 como TFIDF buscan satisfacer la experiencia de usuario al brindarle los resultados que dicho usuario considere relevantes, al final ambos algoritmos o funciones de Lucene buscan dar una buena experiencia de usuario al traer, de manera lo más eficiente posible, aquellos documentos ligados a los términos de consulta utilizados por un usuario.

3.4 ElasticSearch

Se han visto algunos de las bibliotecas más usados en motores de búsqueda distribuidos, como Lucene, así como algunas de sus funciones, o bien algoritmos, más utilizados en el mercado para definir el ranking de relevancia de una búsqueda. Ahora se busca dar un panorama de cómo funcionan estos algoritmos en un motor de búsqueda, en este caso, en el motor de búsqueda y analítica distribuido, conocido como ElasticSearch.

Elasticsearch está posicionado como líder en el Cuadrante Mágico de Gartner 2022 para motores de Insight (Emmott et. al., 2022), el cual agrupa las máquinas de búsqueda disponibles en el mercado. Tiene un amplio alcance de mercado, respaldado por una red de más de 1,400 socios, lo cual hace muy relevante su presencia en la industria. Una gran ventaja para justificar su uso en la presente investigación es el hecho que aparte de ser ampliamente utilizado en el mercado, es de código abierto y gratuito. Este motor fue desarrollado a partir de Apache Lucene y fue lanzado al mercado en el año 2010. (*What Is Elasticsearch?*, s/f)

Permite indexar, realizar búsquedas y analizar tanto documentos de texto estructurado como de no estructurado, además de datos geoespaciales. Permite almacenar los documentos e indexarlos de una forma muy sencilla, la cual le permite tener búsquedas rápidas de aquellos documentos ya indexados.

ElasticSearch es un almacén de documentos distribuidos, no obstante, no almacena la información de manera tradicional, es decir, en filas y columnas, sino que almacena la estructura de los datos en JSON. Cuando guarda un documento, primero lo indexa, una vez realizado esto el documento ya puede ser buscado por el motor, de manera eficiente, ElasticSearch utiliza una estructura llamada índice invertido para su indexación, este índice se encarga de enumerar cada

palabra única que salga en el documento e identifica todos los documentos en los que dicha palabra aparece.

Este índice se ve como la colección optimizada de los documentos, los cuales son un conjunto de campos, dichos campos funcionan como pares de clave-valor, los cuales contienen los datos. Esta indexación de los datos de cada campo la realiza ElasticSearch de manera predeterminada, dándole a dichos campos una estructura de datos óptima y dedicada. De igual forma, ElasticSearch permite indexar estos documentos sin especificar como se debe manejar cada uno de los campos del documento, esto ya que ElasticSearch posee una función de mapeo dinámico, la cual detecta y va agregando, de manera automática, nuevos campos en el índice, esto facilita la indexación y la exploración de los datos. Además de esto, ElasticSearch permite definir reglas que controlan este mapeo automático y a partir de dichas reglas definir el mapeo de cómo se guardan e indexan los campos del documento.

Realizar este mapeo con base en las reglas definidas permite:

- Diferenciar entre campos de cadena de texto completo o bien campos de cadena de valor.
- Analizar el texto específico del idioma.
- Mejorar los campos para coincidencias parciales.
- Hacer uso de formato de fechas personalizado.

Ahora bien, a pesar de que ElasticSearch se puede usar como almacén de documentos y el mismo permite realizar recuperación, no solo de los documentos sino también de los metadatos de estos; dicho motor de búsqueda se puede aprovechar de una mejorar manera cuando trabaja con Lucene, el cual le permite tener acceso al conjunto completo de las capacidades integradas en la biblioteca del motor.

ElasticSearch, integrado con Lucene, permite realizar consultas estructuradas y consultas de texto completo y a su vez realizar consultas complejas que tengan una combinación de ambas. Estas consultas estructuradas son parecidas a las que se realizan en SQL, donde se puede realizar la búsqueda por un campo específico y su índice ordena los resultados por las coincidencias de dicho campo, por otro lado, las consultas de texto completo buscan la coincidencia que encontró

en los todos los documentos donde aparece la cadena de texto de la consulta y los resultados los ordena por relevancia, es decir, qué tan bueno es el documento para los términos de búsqueda utilizados. ElasticSearch también permite realizar búsquedas con términos individuales, frases, prefijos, búsquedas de similitud y dar sugerencias de autocompletado, a su vez también permite buscar por datos geoespaciales o datos numéricos, ya que se indexan los datos no textuales en estructuras de datos óptimas que permiten realizar consultas geográficas y numéricas. Para realizar todo este tipo de consultas, ElasticSearch utiliza el lenguaje Query DSL, el cual es un lenguaje de consulta de estilo JSON, también permite realizar consultas estilo SQL.

ElasticSearch fue diseñado para estar siempre disponible y que permita ser escalable según las necesidades del usuario, esto lo vuelve un motor de búsqueda distribuido, lo cual le permite agregar nodos(servidores) a un clúster para aumentar la capacidad. De manera automática, una vez esto sucede, ElasticSearch distribuye la carga de datos y las consultas en cada uno de los nodos, esta distribución automática es posible ya que los índices son realmente una agrupación lógica de uno o más fragmentos, en donde cada uno de los fragmentos es realmente un índice autónomo, por lo cual, al distribuir los documentos en un índice en varios fragmentos y a su vez distribuir estos fragmentos en varios servidores, se puede garantizar la redundancia, lo cual hace posible no solo aumentar la capacidad de consulta cada vez que se agregan nodos al clúster, sino que además protege contra fallas del hardware, por lo que, cada vez que el clúster cambia de tamaño, ElasticSearch pasa los fragmentos de manera automática para equilibrar el clúster.

Existen dos tipos de fragmentos:

- Primarios: estos contienen cada documento en un índice
- Réplica: son una copia de un fragmento primario, estas réplicas proporcionan copias redundantes de los datos para protegerse de las fallas y permiten mejorar la capacidad de atender las consultas.

Ahora bien, la cantidad de fragmentos primarios se fija cuando se crea un índice, pero la cantidad de los fragmentos réplica puede variar en cualquier momento sin que esta interrumpa la indexación o las consultas. Es importante tener en cuenta que mientras más fragmentos

primarios mayor el gasto de rendimiento para mantener los índices y cuanto mayor sea el tamaño del fragmento más tiempo se va a tardar en equilibrar el clúster si este llega a variar su tamaño. Por lo que, para evitar esto, se recomienda tratar de mantener un tamaño de fragmento promedio de unos pocos Gigabytes y a su vez evitar la gran cantidad de fragmentos, ya que, si bien la cantidad de fragmentos que puede tener un nodo es proporcional a su espacio de almacenamiento, lo mejor es mantener una cantidad menor a 20 fragmentos por nodo, estas recomendaciones tratan de evitar problemas de rendimiento.

Por otro lado, si bien los nodos de un clúster, para tener buenas conexiones entre sí, colocan los nodos en el centro de datos y así tener siempre una alta disponibilidad, es importante evitar que se cree un punto de falla único, por lo que se busca tener un relevo en caso de que suceda una falla o interrupción en el clúster principal. Para evitar esta situación se realiza una replicación entre clústeres, la cual permite sincronizar de manera automática los índices del clúster principal con el clúster secundario, para que este funcione como una copia de seguridad, por lo que si el clúster principal falla, el clúster secundario pueda tomar el cargo. Esta técnica de replicación de clústeres no solo ayuda con este tipo de situaciones, sino que también permite crear clústeres secundarios, los cuales puedan atender consultas o solicitudes según la posición geográfica del usuario.

3.4.1 Arquitectura de ElasticSearch

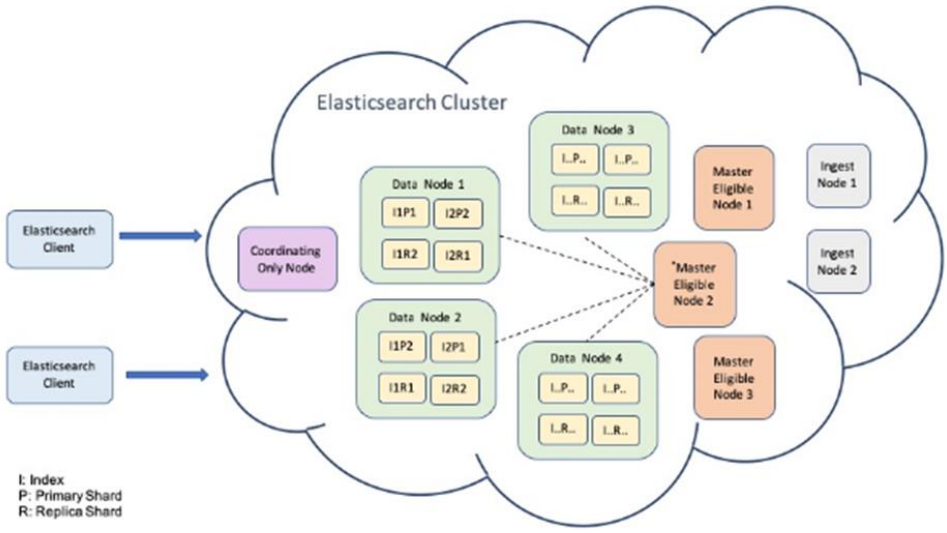


Ilustración 22 Arquitectura Elastic Search

Fuente: (Wong, 2019)

Un clúster de Elasticsearch es un grupo de uno o más nodos que están conectados entre sí, cada nodo tiene su propio propósito y responsabilidad, cada nodo puede reenviar solicitudes de clientes. Ahora bien, estos son algunos de los nodos en la arquitectura de ElasticSearch:

- Nodo principal elegible: se encarga principalmente de las operaciones ligeras en todo el clúster como la creación o eliminación de un índice, el seguimiento de los nodos del clúster y la determinación de la ubicación de los fragmentos asignados.
- Nodo de datos: contienen los documentos indexados. Maneja operaciones como la búsqueda y agregación.
- Nodo de ingesta: procesa un documento en modo de canalización antes de indexarlo de forma predeterminada.
- Nodo solo de coordinación: es un nodo de coordinación que realiza solicitudes de enrutamiento, maneja la fase de reducción de búsqueda y distribuye trabajos a través de indexación masiva. Esto solo si los otros nodos mencionados se encuentran deshabilitados.

Por otro lado, en ElasticSearch, como bien se mencionó con anterioridad, los datos están organizados en índices y cada uno de los índices es un espacio lógico, el cual permite organizar los datos. Como bien se menciona en la sección ElasticSearch, los documentos son la unidad

básica en ElasticSearch, a los cuales, gracias al proceso de indexación, le crea un índice invertido tokenizando los términos del documento, y así creando una lista ordenada de todos los términos únicos y asociando la lista de documentos con la ubicación donde se encuentran dichos términos.

Los índices constan de uno o más fragmentos. Estos fragmentos no son más que un índice de Lucene que utiliza el índice invertido o estructura de datos para guardar dichos datos. Los fragmentos pueden no tener réplicas o tener más de una. ElasticSearch se encarga de que el fragmento principal y su réplica no se coloquen en el mismo nodo.

Cada índice de Lucene, o fragmento, cuenta con segmentos, los cuales no son índices invertidos funcionales, cada uno de los fragmentos o índices de Lucene cuentan con uno o más segmentos de índice inmutables, y un segmento es un índice invertido funcional. Estos segmentos inmutables, le permiten a Lucene agregar nuevos documentos al índice de forma incremental sin necesidad de reconstruirlos. Para poder administrar la cantidad de segmentos, ElasticSearch unifica aquellos segmentos pequeños en un segmento grande, luego confirma este nuevo segmento unificado en el disco y elimina los antiguos segmentos pequeños en el momento preciso. (Wong, 2019)

En cada consulta o bien solicitud que se realiza, se buscará en todos los segmentos de Lucene de un fragmento determinado de un índice. A continuación, se muestra el proceso del clúster a la hora de recibir una consulta:

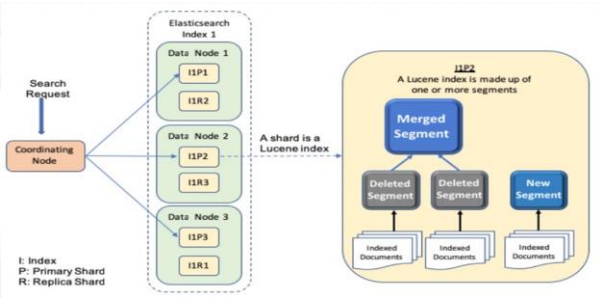


Ilustración 23 Proceso de consulta en un clúster de ElasticSearch

Fuente: (Wong, 2019)

4 Marco Metodológico

4.1 Tipo de Investigación

El objetivo general de la investigación es evaluar una propuesta de operadores de relevancia de máquinas de búsqueda por medio de su implementación en una máquina de búsqueda de amplio uso y de código abierto, se considera que la investigación es de tipo evaluativo.

4.2 Alcance Investigativo

De acuerdo con los objetivos de la investigación, se considera que tiene el siguiente alcance investigativo:

- **Alcance Descriptivo**: Debido a que estamos recolectando datos sobre diferentes consultas y sobre cómo pueden mejorarse de acuerdo con métricas que serán definidas más adelante y a interpretaciones de relevancia de diferentes consultas al utilizar los operadores propuestos, se considera que es una investigación de alcance descriptivo.
- **Alcance Explicativo**: Asimismo, se considera que la investigación puede evolucionar hacia un alcance explicativo, dado que será necesario establecer las causas de los cambios en las métricas observadas durante la ejecución de la investigación.

4.3 Enfoque

Las evaluaciones de relevancia en recuperación de información tienen un componente subjetivo (Manning et al., 2008). Los métodos como la precisión, cobertura, y exactitud se basan en juicios de valor anteriores determinados por grupos de usuarios o expertos (Samimi & Ravana, 2014). Adicionalmente a la evaluación de la relevancia como tal, en la presente investigación se evalúan operadores que serán utilizados y configurados por desarrolladores y demás personal técnico, por lo que su facilidad de configuración y uso deben ser considerados como parte del enfoque. Finalmente, la capacidad de visualizar o explicar por qué un documento es retornado en cierta posición en los resultados. Esto hace que la evaluación tenga componentes tanto cualitativos como cuantitativos.

Debido a lo anterior, se considera utilizar un **enfoque alternativo** tal y como se expone en (Naranjo-Zeledón, 2020). En este enfoque se utilizarán tanto métricas calculables para la

evaluación de algunos aspectos de esta, así como interpretaciones cualitativas con el fin de explorar los efectos y las capacidades de los operadores de relevancia propuestos.

4.3.1 Dimensión Epistemológica

Debido a que la presente investigación consiste en la comparación de métodos y operadores de relevancia existentes en contraste con una nueva propuesta de métodos y operadores de relevancia, los investigadores asumirán la posición de observadores. Con el fin de realizar las observaciones y las comparaciones, los investigadores utilizaran medidas estandarizadas de recuperación de información para evaluar los resultados, como lo son la precisión, cobertura, F1, y nDCG. Además, se utilizarán factores de evaluación cualitativos para tomar en consideración la flexibilidad, facilidad de uso y que tan fácil es explicar los resultados que proveen, tal y como se explican en la sección 4.3.3.

4.3.2 Dimensión Ontológica

La representación ontológica de los algoritmos normalmente utilizados en los operadores de las máquinas de búsqueda, y específicamente de Lucene, se pueden apreciar en la siguiente ilustración:

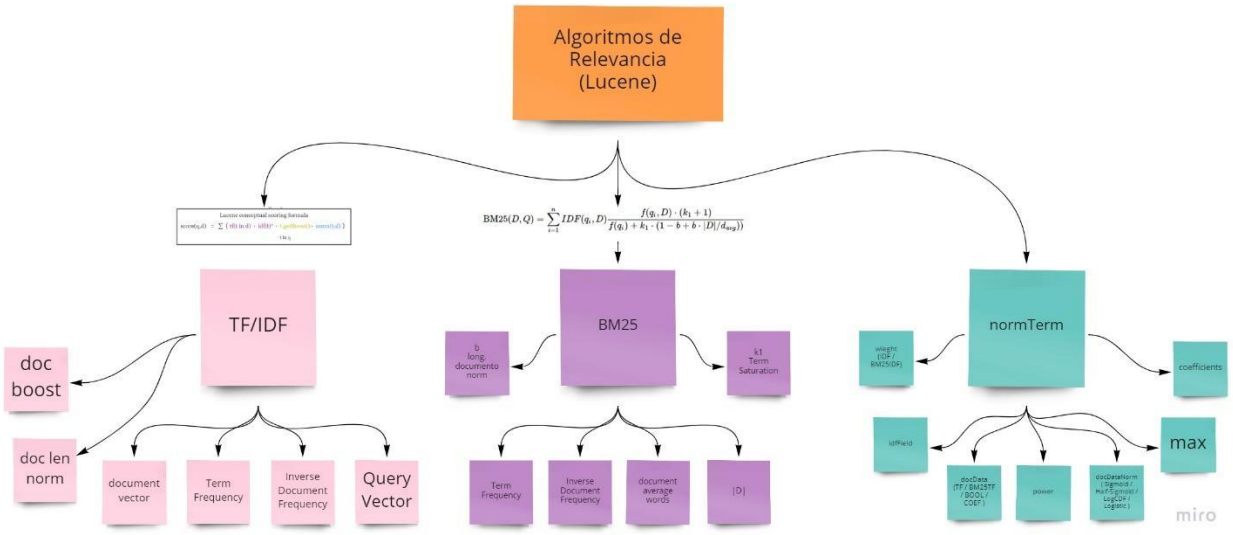


Ilustración 24 Ontología de las fórmulas y operadores de relevancia utilizados por Lucene.

Fuente: Elaboración propia

Los operadores que en esta ontología están representados por el operador *normTerm* (P. E. Nelson & David, 2021).

4.3.3 Dimensión Axiológica

La dimensión axiológica requiere un tratamiento especial en cuanto a la evaluación de los operadores propuestos. Esto debido a que si bien se tienen métricas estándar del campo de recuperación de información (Manning et al., 2008), también se deben considerar factores de uso de estos, como la flexibilidad de estos para adaptarse a diferentes tipos de escenarios de búsqueda, su facilidad de uso para el desarrollador de un motor de búsqueda, y la facilidad que tengan para explicar por qué un resultado está en cierta posición de relevancia con respecto a otro resultado.

Tabla 27 Dimensión Axiológica

Rubro	Valor	Comentarios
Evaluación de métricas de recuperación de información		
Precisión	20%	Precision@10
Cobertura	10%	Complicada de evaluar al existir la posibilidad que no se conozca la totalidad de los documentos relevantes
F1	10%	Al combinar la precisión y cobertura se considera que debe tener una importancia menor por las dificultades de la cobertura mencionadas
nDCG	20%	Los documentos relevantes deben aparecer en posiciones altas en los resultados, de lo contrario se penaliza conforme aparezcan en posiciones más bajas.
Evaluación de factores de uso de los operadores		

Flexibilidad	20%	Se pretende analizar la capacidad del operador al realizar consultas con diferentes conjuntos de datos, así como analizar cómo se compartan con varios tipos de datos.
Facilidad de uso	10%	Da un vistazo de la dificultad de poder implementar un operador u otro en la máquina de búsqueda por parte del desarrollador
Explicación de resultados	20%	Permite conocer que tan fácil le es al usuario determinar por qué el operador considero un resultado más relevante que otro.

Fuente: Elaboración propia

4.4 Diseño

El diseño de la investigación es de tipo experimental, utilizando métricas para comparar dos implementaciones de máquina de búsqueda.

4.5 Población y Muestreo

Para el desarrollo de investigaciones en el campo de la recuperación de información es necesario contar con una colección de prueba. Esta colección debe tener un conjunto de documentos de tamaño no trivial, un conjunto de consultas para ejecutar sobre los documentos, y para cada consulta, un conjunto de documentos relevantes para la misma. Esta última parte, sobre los documentos relevantes, normalmente se crea de forma manual. (Sanderson, 1994)

Para el desarrollo de la investigación se revisaron las siguientes colecciones con el fin de establecer una línea base entre la forma en que se calcula la relevancia en la máquina de búsqueda Elasticsearch y compararla con la propuesta de la presente investigación.

Para esto, se consideran las siguientes colecciones:

- Colección de prueba de Reuters-21578 (Lewis & Ringuette, 1994): Esta colección consiste en 21,578 documentos. Está disponible para uso en (David D. Lewis & Steve Finch, 1997).

- TREC Washington Post Corpus (Post, 2017): Consiste en 728,6262 artículos de noticias y entradas de blog de enero del 2012 a diciembre del 2020. Este fue utilizado para la conferencia TREC 2018 y requiere de una solicitud hacia la organización NIST para poder utilizarlo.
- TREC 2022 Fair Ranking Track (Ekstrand et al., 2022): corpus de evaluación para el programa de clasificación justa de la conferencia TREC 2022. Consiste en alrededor de 87gb de artículos de Wikimedia y un conjunto de consultas tanto de entrenamiento como de evaluación
- Collections held at Glasgow (Glasgow Information Retrieval Group, 2023) el cual es un conjunto de colecciones de documentos creada para realizar pruebas en el campo de la recuperación de la información por la universidad de Glasgow, la cual consiste de un total de 8 colecciones la cuales contiene un total de 24903 documentos divididos en estas 8 colecciones, por otro lado estas colecciones también incluyen un conjunto de consultas de entrenamiento para ayudar en la evaluación de los resultados de una búsqueda, así como su respectiva lista de documentos relevantes para cada una de las consultas.

Debido a restricciones de tiempo y capacidad computacional, así como su fácil accesibilidad, se utiliza algunas de las colecciones de prueba de “Collections held at Glasgow”, tomando en cuenta las categorías asignadas a cada documento como las consultas a las que ese documento es relevante. Además, que dicha colección nos permite no solo evaluar los operadores con un solo conjunto de documentos, sino que nos permite probar los operadores con distintas colecciones de documentos, lo cual no permite un mejor análisis de estos en diferentes escenarios.

4.5.1 Colección de prueba Cranfield

La colección consiste en 1400 documentos y una serie de 225 consultas para poder probar y evaluar los distintos motores de búsquedas y sus operadores. Esta colección trae una serie de 3 archivos los cuales se dividen de la siguiente manera:

4.5.1.1 Formato de los documentos

Dicha colección cuenta con un archivo el cual trae todos los documentos de la colección, este archivo viene en un formato en el cual tenemos el número de documento y cuerpo de dicho documento los cuales se van a indexar posteriormente a ElasticSearch. Este archivo posee el siguiente formato en una especie de etiquetas, siendo estas las siguientes:

- ". I" seguido del número de documento. Este a su vez delimita el inicio de cada documento.
- ". T" la cual es seguido del título del documento.
- ". A" esta etiqueta nos indica el autor del documento.
- ". B" etiqueta la cual hace referencia a la bibliografía del documento.
- ". W" en dicha etiqueta se coloca el texto o bien el contenido del documento.

A continuación, un ejemplo del formato del documento:

```
. I 18

.T

the flow field in the diffuser of a radial compressor .

.A

rhyming,i.l.

.B

j. ae. scs. 27, 1960, 798.

.W

the flow field in the diffuser of a radial compressor .

this note discusses the two-dimensional diffuser flow field
```

Ilustración 25 Ejemplo de un documento en la colección Cranfield

Fuente: Tomado del archivo la colección Cranfield

4.5.1.2 Formato de las Consultas

Por otro lado, la colección posee un archivo el cual tiene una serie de consultas de prueba las cuales permiten realizar una evaluación posterior al operador utilizado para realizar dicha búsqueda. Esta consulta posee una estructura con formato similar a los documentos:

- ". I" etiqueta la cual marca el inicio de una nueva estructura, además esta es seguida del índice del número de consulta.
- ". W" etiqueta la cual contiene el contenido de la consulta.

El siguiente es un ejemplo de una consulta:

```
. I 001

.W

what similarity laws must be obeyed when constructing aeroelastic models
```

Ilustración 26 Ejemplo de consulta de la colección Cranfield

Fuente: Tomado del archivo la colección Cranfield

4.5.1.3 Formato de los documentos de relevancia

Por último, la colección trae un archivo el cual indica la relevancia de cada uno de los documentos de acuerdo con las consultas de prueba. Este archivo está formado por el siguiente formato:

1 184 2

1 29 2

1 31 2

1 12 3

1 51 3

1 102 3

Ilustración 27 Ejemplo del documento de relevancia de la colección Cranfield

Fuente: Tomado del archivo la colección Cranfield

Teniendo esta estructura un formato en el cual el primer digito de cada fila indica el número de consulta realizada de acuerdo al archivo de consultas de la colección, posteriormente el número de documento que es relevante para dicha consulta y por ultimo un número de relevancia del -1 al 4 de dicho documento para la consulta asociada, siendo los documentos con relevancia 4 los más importante o bien relevantes para la consulta y siendo los de relevancia -1 aquellos documentos con poca relevancia para la consulta. Esta estructura nos permite evaluar el operador ya que nos indica cuales documentos son relevantes para una consulta dada.

4.5.2 Colección de prueba Medline

La colección posee un total de 1033 documentos y al igual que la colección anterior una serie de consultas para poder probar y evaluar los operadores. Esta colección trae una serie de 3 archivos los cuales se dividen de la siguiente manera:

4.5.2.1 Formato de los documentos

Esta colección posee un archivo el cual contiene la información de todos los documentos de la colección el cual esta demarcado por una serie de etiquetas que tienen el siguiente formato:

- ". I" seguido del número de documento. Este a su vez delimita el inicio de cada documento.

- ". W" en dicha etiqueta se coloca el texto o bien el contenido del documento.

Ejemplo de un documento de la colección Medline:

```
. I 13
.W
analysis of mammalian lens proteins by electrophoresis .
  lens proteins of different mammalian species were analyzed by
two-dimensional starch gel electrophoresis . the number of fractions
detected by this means varied from 11-20 . a-crystallin was resolved
into two to three components, b-crystallin into 5-11, and y-crystallin
```

Ilustración 28 Ejemplo de un documento en la colección Medline

Fuente: Tomado del archivo de la colección Medline

4.5.2.2 Formato de las Consultas

Además, esta colección tiene un archivo el cual contiene todas las consultas de prueba siendo un total de 30 que se pueden realizar para dichos documentos los cuales poseen la siguiente estructura:

- . ". I" etiqueta la cual marca el inicio de una nueva estructura, además esta es seguida del índice del número de consulta.
- ". W" etiqueta la cual contiene el contenido de la consulta.

A continuación, un ejemplo de una consulta:

```
. I 1
.W
```

Ilustración 29 Ejemplo de consulta en la colección Medline

Fuente: Tomado del archivo de la colección Medline

4.5.2.3 Formato de los documentos de relevancia

Esta colección además trae un archivo la cual nos ayuda a tener una guía de la relevancia de los documentos en base a la consulta de prueba realizada. Este archivo tiene el siguiente formato:

Esta colección además trae un archivo la cual nos ayuda a tener una guía de la relevancia de los documentos en base a la consulta de prueba realizada. Este archivo tiene el siguiente formato:

```
1 0 13 1
1 0 14 1
1 0 15 1
```

Ilustración 30 Ejemplo del documento de relevancia de la colección Medline

Fuente: Tomado del archivo de la colección Medline

La cual muestra una serie de números, siendo el primero el número de consulta, seguido de un campo con valor "0" le cual no se toma en cuenta, luego el número de documento que es relevante para la consulta y por último el número de relevancia de dicho documento para la consulta asociada. Esta colección da una misma relevancia a todos los documentos relevantes para una consulta determinada por lo que no le da más importancia a un documento u otro para una consulta.

4.5.3 Colección de prueba Time

Esta colección tiene un total de 423 documentos y posee 83 consultas para poder probar y evaluar los operadores de un motor de búsqueda. Esta colección trae una serie de 3 archivos los cuales se dividen de la siguiente manera:

4.5.3.1 Formato de los documentos

Esta colección contiene un archivo con los documentos los cuales van a ser indexados al motor de búsqueda, este archivo tiene los documentos de la colección dividas por número de documentos asociado al cuerpo del documento. Este archivo de documentos tiene el siguiente formato:

Este formato posee una "etiqueta" la cual inicia de la siguiente manera:

```
*TEXT 020 01/04/63 PAGE 021
```

Ilustración 31 Ejemplo de la etiqueta inicio de los documentos de la colección Time

Fuente: Tomado del archivo de la colección Time

El cual marca el inicio de cada uno de los documentos. Esta etiqueta a su vez está compuesta de la siguiente manera:

Primero "*TEXT" el cual marca el inicio de la etiqueta seguido del número de documento, luego por la fecha del documento y por último la página del contenido del texto. Después de dicha etiqueta viene el texto o bien contenido de dicha página del documento. A continuación, un ejemplo del formato del documento de la colección Time:

```
*TEXT 020 01/04/63 PAGE 021

THE ROAD TO JAIL IS PAVED WITH NONOBJECTIVE ART SINCE THE

KREMLIN'S SHARPEST BARBS THESE DAYS ARE AIMED AT MODERN ART AND "

WESTERN ESPIONAGE, " IT WAS JUST A MATTER OF TIME BEFORE THE KGB'S COPS

WOULD TURN UP A VICTIM WHOSE WRONGDOINGS COMBINED BOTH EVILS . HE

TURNED OUT TO BE A LENINGRAD PHYSICS TEACHER WHOSE TASTE FOR ABSTRACT

PAINTING ALLEGEDLY LED HIM TO JOIN THE U.S . SPY SERVICE . POLICE SAID
```

Ilustración 32 Ejemplo de un documento de la colección Time

Fuente: Tomado del archivo de la colección Time

4.5.3.2 Formato de las consultas

Esta colección al igual que las demás posee un archivo el cual trae una serie de consultas de prueba las cuales permiten evaluar después el operador usado en ElasticSearch. Este archivo posee el número de consulta y la consulta a realizar, el cual tiene el siguiente formato:

```
*FIND      1

 KENNEDY ADMINISTRATION PRESSURE ON NGO DINH DIEM TO STOP

SUPPRESSING THE BUDDHISTS.
```

Ilustración 33 Ejemplo de consulta de la colección Time

Fuente: Tomado del archivo de la colección Time

Esta estructura está compuesta en dos partes:

- Primero la palabra "*FIND" seguido del número de consulta, esta a su vez delimita el inicio de una consulta.
- La segunda parte está compuesta por la consulta o bien el contenido de la consulta a realizar.

4.5.3.3 Formato de los documentos de relevancia

Esta colección también tiene un archivo el cual nos ayuda o bien brinda una guía para saber cuáles son los documentos más relevantes para alguna de las consultas de prueba. Este archivo tiene el siguiente formato:

```
1 268 288 304 308 323 326 334

2  326 334

3  326 350 364 385
```

Ilustración 34 Ejemplo del documento de relevancia de la colección Time

Fuente: Tomado del archivo de la colección Time

En la cual solo asocia los documentos relevantes a una consulta, es decir, no le da ningún tipo de relevancia numérica a dichos documentos solo asocia los documentos relevantes en "general" para la consulta dada. Siendo el primer digito de la fila para el número de la consulta realizada, seguida de los documentos que son relevantes para esa consulta.

4.5.4 Comparación de las colecciones

Cada una de estas colecciones posee características distintas en cuanto tamaño de esta, número de documentos y consultas, etc. Esto se debe a que se busca el poder evaluar los operadores con distintas colecciones de documentos con el fin de analizar el rendimiento y comportamiento de cada operador según la colección. A continuación, se realiza un comparativo de las estadísticas de las colecciones a utilizar y descritas en la sección 4.5.

Dicho comparativo se puede apreciar en la siguiente tabla:

Tabla 28 Estadísticas básicas de las colecciones de Glasgow

Estadísticas Básicas	Colección Cranfield	Colección Time	Colección Medline
Número de documentos	1400 documentos	423 documentos	1033 documentos
Número de consultas	225 consultas	83 consultas	30 consultas
Tamaño en megabytes (MB) de la colección.	1.6 MB	1.5 MB	1.1 MB

Fuente: Elaboración propia

Además de estas estadísticas básicas las colecciones poseen estadísticas respecto a los documentos en las cuales podemos apreciar datos como:

- Máximo número de palabras en un documento
- Promedio de palabras por documento
- Media de palabras por documento

Por otro lado, también se tiene estadísticas respecto a las consultas en las cuales se tiene datos como:

- Máximo # de palabras en una consulta
- Promedio de palabras por consulta
- Media de palabras por consulta

Cada una de estas estadísticas dan un panorama de lo diversas que son las colecciones a utilizar para la evaluación de los operadores.

Ahora bien, se puede ver un comparativo de las mismas en las siguientes figuras:

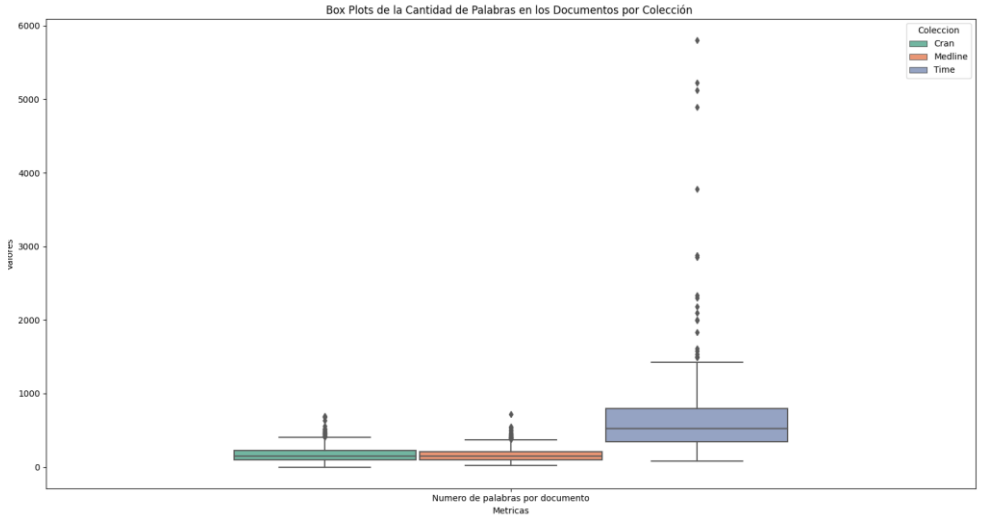


Figura 1 Cantidad de palabras por documento según cada colección. Fuente Elaboración propia.

Como se puede apreciar en la Figura 1 la cantidad de palabras por documento es muy variada y estas no dependen de la cantidad de los documentos de la colección sino del contenido de estos.

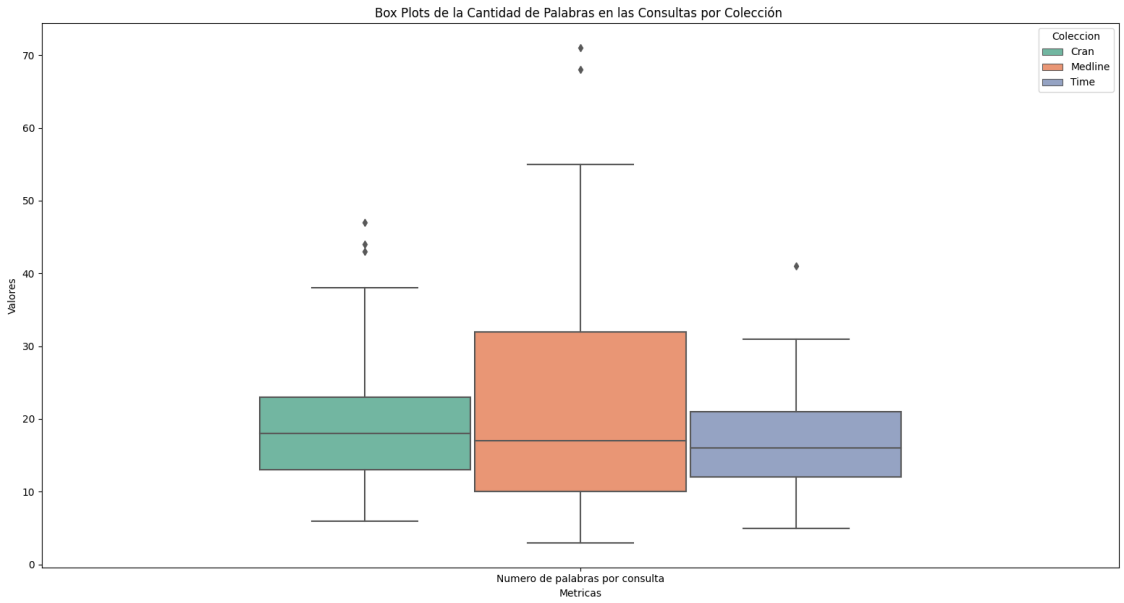


Figura 2 Cantidad de palabras por consulta según cada colección. Fuente Elaboración propia.

Por otro lado, se observa en la Figura 2 la cantidad de palabras utilizadas en las consultas por colección, en la cual se aprecia que algunas de las consultas son muy elaboradas como lo son las de la colección Medline dado el número de palabras por consulta.

A continuación, se detalla las estadísticas de los documentos por colección, las cuales son comprados en la Tabla 29 y seguidos de una serie de figuras las cuales muestran en figuras los números vistos en dicha tabla. Con este comparativo se pretende hacer ver las diferencias de los documentos de una colección a otra.

Tabla 29 Comparativo de las estadísticas de los documentos de las colecciones de Glasgow

Estadísticas de los documentos	Colección Cranfield	Colección Time	Colección Medline
Máxima cantidad de palabras por documento	698 palabras máximas por documento	5801 palabras máximas por documento	718 palabras máximas por documento
Promedio de palabras por documento	170.54 palabras promedio por documento	673 palabras promedio por documento	167.31 palabras promedio por documento
Media de palabras por documento	700.5 palabras media por documento	528 palabras media por documento	517 palabras media por documento

Fuente: Elaboración propia

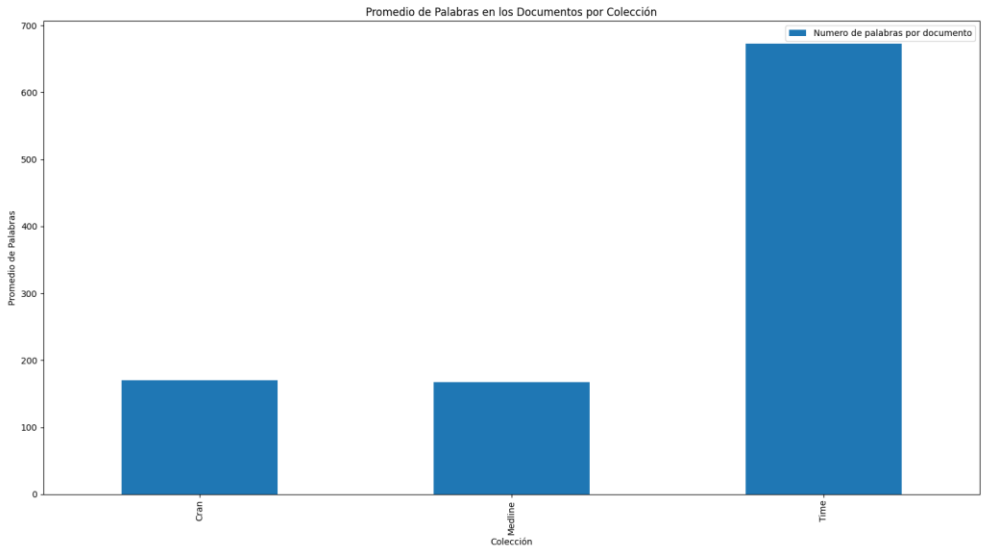


Figura 3 Cantidad de palabras promedio en los documentos por colección. Fuente Elaboración propia.

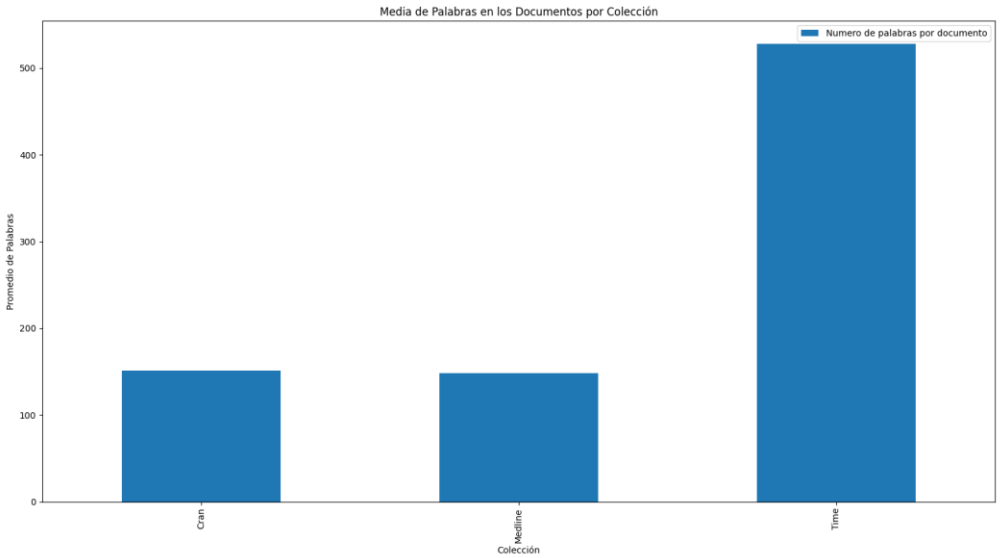


Figura 4 Cantidad de palabras media en los documentos por colección. Fuente Elaboración propia.

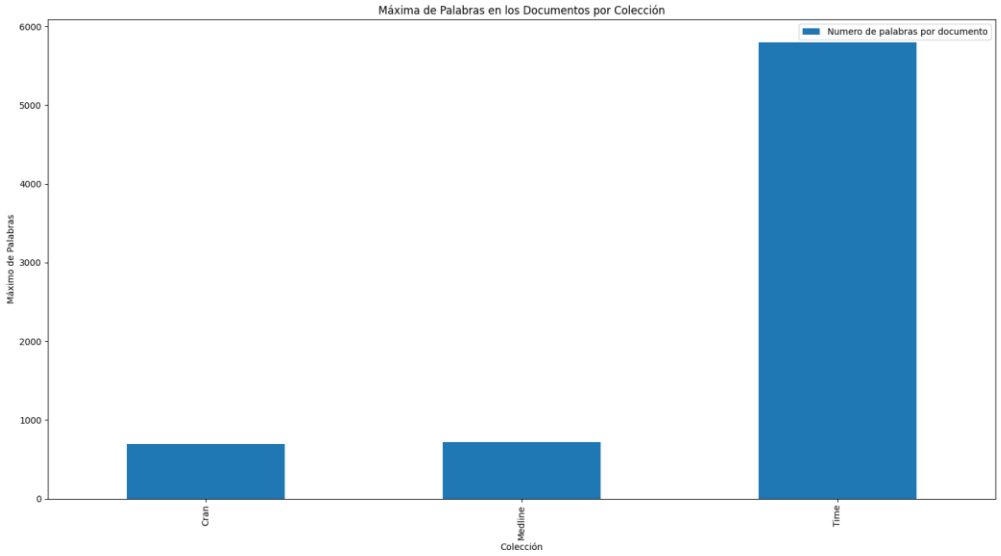


Figura 5 Cantidad de palabras máximas en los documentos por colección. Fuente Elaboración propia.

Ahora bien, se realizó el mismo comparativo para las consultas, ya que estas consultas, tal como se observa en la Figura 2 varían su tamaño en cuanto a palabras y contenido según la colección, por ende, se realizó un comparativo similar a las estadísticas de los documentos, el cual se puede apreciar en la siguiente tabla y figuras:

Tabla 30 Comparativo de las estadísticas de las consultas de las colecciones de Glasgow

Estadísticas de las consultas	**Colección Cranfield**	**Colección Time**	**Colección Medline**
Máxima cantidad de palabras por consulta	47 palabras máximas por consulta	41 palabras máximas por consulta	71 palabras máximas por consulta
Promedio de palabras por consulta	18.35 palabras promedio por consulta	16.69 palabras promedio por consulta	23.75 palabras promedio por consulta
Media de palabras por consulta	18 palabras media por consulta	16 palabras media por consulta	17 palabras media por consulta

Fuente: Elaboración propia

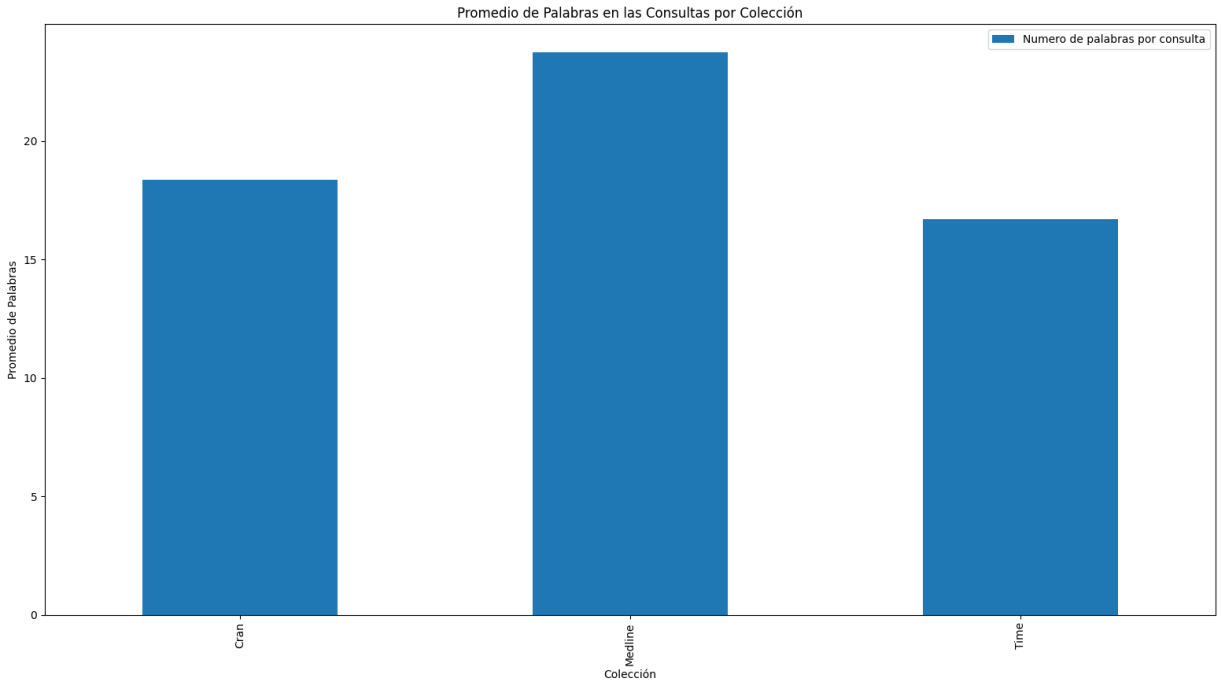


Figura 6 Cantidad de palabras promedio en las consultas por colección. Fuente Elaboración propia.

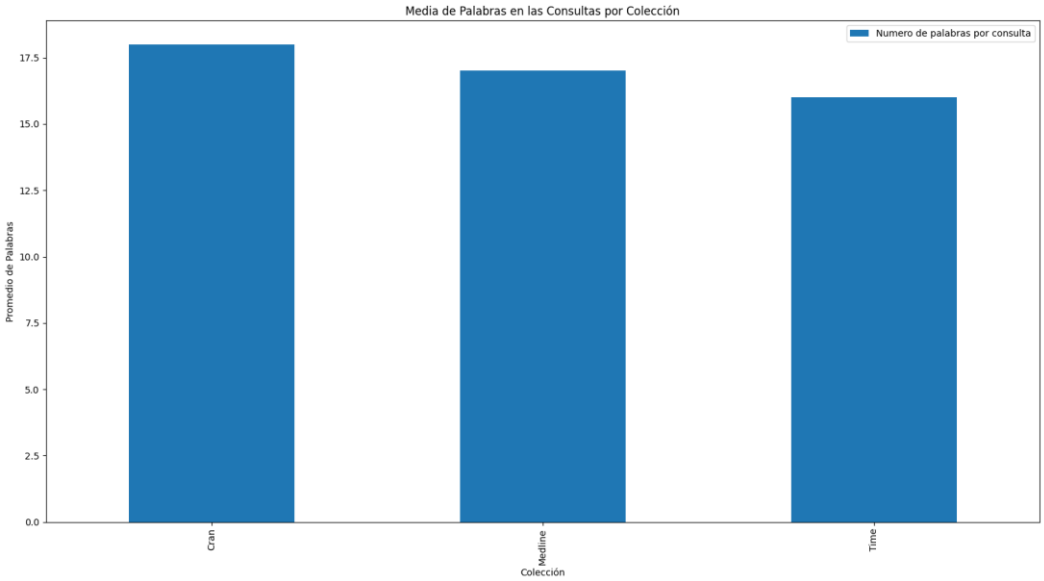


Figura 7 Cantidad de palabras media en las consultas por colección. Fuente Elaboración propia.

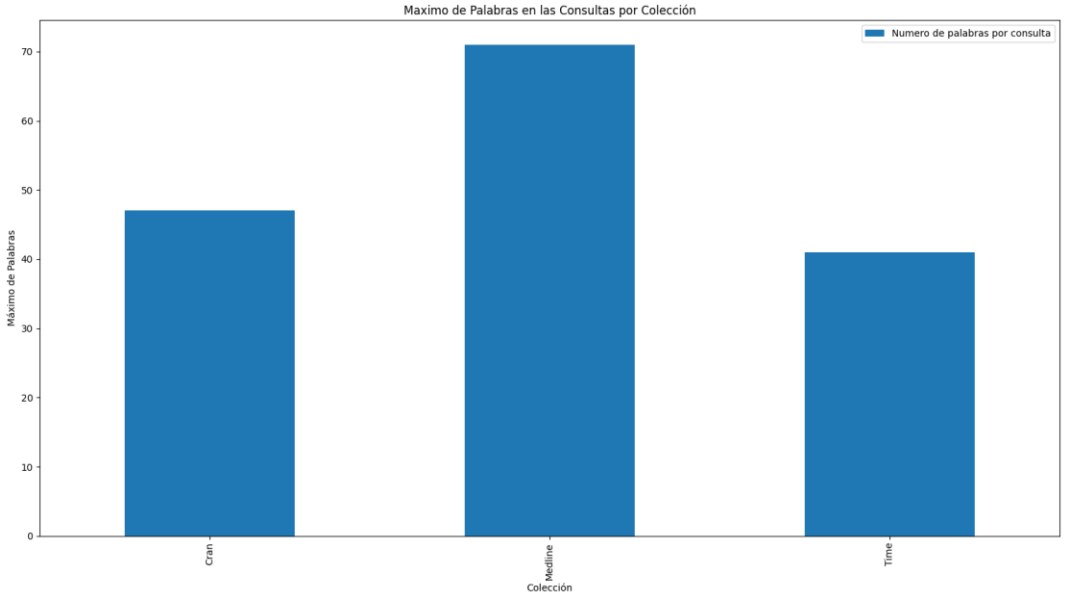


Figura 8 Cantidad de palabras máximas en las consultas por colección. Fuente Elaboración propia.

Como se puede apreciar cada una de las colecciones son diferentes entre sí, esto es lo que se pretende al hacer uso de ellas para evaluar los operadores, ya que dichos operadores se podrán a prueba en colecciones distintas con el fin de poder evidenciar con el operador propuesto una normalización en los resultados independientemente de la colección en la que se utilice.

4.6 Instrumentos de Recolección de Datos

Para la recolección de datos se utilizarán diversos instrumentos o técnicas para poder capturar datos de los resultados de la manera más precisa posible, esto con el fin de poder llegar a demostrar la propuesta planteada en la investigación, así como alcanzar los objetivos propuestos en esta investigación.

Todo este proceso de recolección de datos permite distinguir aquellos datos que son más importantes para la investigación, así como la manera en la que los mismos van a hacer evaluados y obtenidos a lo largo del proyecto.

4.6.1 Análisis Documental

Se utiliza dicha técnica para la identificación y recolección de documentos relacionados al contexto de la investigación, con esto se pretende compartir significados a través de las

elaboraciones escritas de las personas referentes en el tema de la investigación. Con esto se han fijado objetivos con el fin de alcanzar, de manera eficiente, esta recolección de documentos importantes para la investigación, entre estos objetivos se tienen los siguientes:

- Comprender las metodologías de evaluación empleadas para los operadores de relevancia de una máquina de búsqueda.
- Conocer los conjuntos de datos más adecuados a utilizar en la evaluación de los operadores de relevancia para las máquinas de búsqueda.
- Identificar las métricas a utilizar en la evaluación de los operadores de relevancia para una máquina de búsqueda.

Con dichos objetivos se pretende evaluar, de la manera más adecuada, los operadores de relevancia del motor de búsqueda ElasticSearch, así como el operador propuesto. Para esto se analizan las metodologías utilizadas en trabajos similares, así como las fuentes de datos y las métricas utilizadas en dichas evaluaciones, con el fin de llevar a cabo una evaluación del operador de relevancia propuesto de una manera imparcial y eficiente, buscando determinar la eficiencia y los resultados de este, gracias al análisis de estos documentos o trabajos similares.

4.6.2 Obtención de datos de la evaluación

Para la evaluación de la investigación, es necesario procesar la colección de datos y realizar un conjunto de consultas con los dos tipos de operadores de relevancia a la máquina de búsqueda, con el fin de obtener los resultados y poder analizarlos.

Para esto se propone el siguiente flujo de datos:

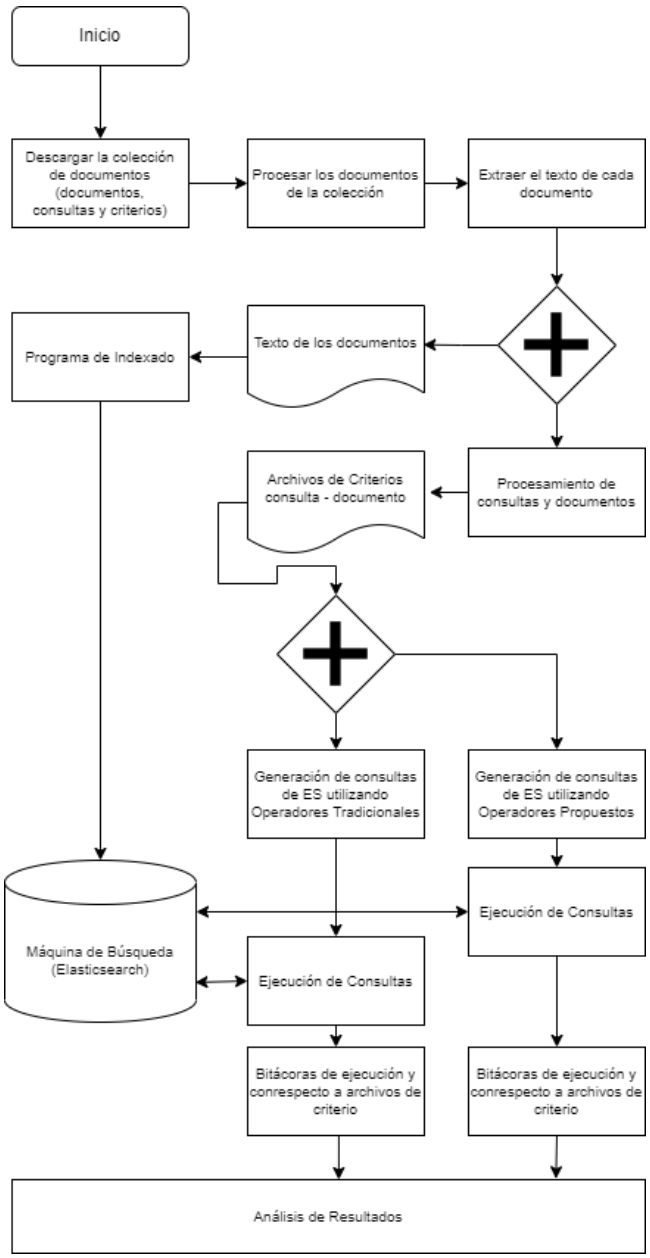


Ilustración 35 Diagrama de flujo de análisis de información.

Fuente: Elaboración propia

En este diagrama se tienen los siguientes pasos:

Tabla 31 Descripción del proceso de análisis de información

Paso	Descripción

Descargar la colección de documentos.	La colección está disponible para su descarga en el sitio (Glasgow Information Retrieval Group, 2023). El primer paso es obtener estas colecciones y ubicarlas en el servidor o estación de trabajo donde se realizará el proceso.
Procesar los documentos de la colección	Durante este paso se desarrollará un programa en el lenguaje de programación Python para interpretar los archivos SGML y separar los diferentes documentos que componen la colección.
Extraer el texto de cada documento	Una vez extraído cada documento, el programa separará la información del texto.
Texto de los documentos	El texto extraído de cada documento es procesado por aparte.
Programa de Indexado	El programa en este paso prepara los documentos para enviarlos a la máquina de búsqueda Elasticsearch y hace el proceso de indexado.
Procesamiento de los documentos	El programa toma los identificadores de los documentos y crea los archivos de criterios consulta-documento.
Archivos de Criterios consulta – documento	Los archivos de criterios consulta-documento contienen una lista de consultas y una lista correspondiente de los documentos relevantes para esa consulta.
Generación de consultas de ES utilizando Operadores Tradicionales	El programa en Python genera, basado en plantilla, consultas basadas en las categorías extraídas de la colección, utilizando operadores tradicionales de Elasticsearch.
Generación de consultas de ES utilizando Operadores Propuestos	El programa en Python genera, basado en plantilla, consultas basadas en las categorías extraídas de la colección, utilizando operadores propuestos en la presente investigación.
Ejecución de Consultas	El programa ejecuta las consultas generadas en los pasos anteriores de forma automatizadas haciendo llamados a la interfaz de aplicación de Elasticsearch.
Bitácoras de ejecución y con respecto a archivos de criterio	La ejecución de las consultas queda registrada en bitácoras, las cuales serán procesadas y comparadas con los archivos de criterios para su análisis posterior.

Análisis de Resultados	Los resultados serán analizados de acuerdo a los criterios establecidos en la sección 3.3.3 Dimensión Axiológica

Fuente: Elaboración propia

Una vez concluida la ejecución y analizados los resultados es posible asignar los puntajes, de acuerdo con la tabla descrita en la sección 4.3.3 para evaluar los operadores desarrollados en la presente investigación en comparación con los operadores incluidos en la máquina de búsqueda.

4.7 Técnicas de Análisis de Información

Con el fin de analizar la información se utilizan las siguientes métricas, tal y como se utilizaron en una evaluación similar en (Marwah & Beel, 2020), en la cual se evalúa un algoritmo similar con las siguientes métricas de evaluación, que son de relevancia en el campo de la recuperación de información. Estas métricas por evaluar son:

- Precisión: en dicha métrica se pretende tomar los mejores 10 resultados, de la búsqueda utilizada con el operador que trae por defecto la máquina de búsqueda y a su vez compararlo con los primeros 10 resultados utilizando el algoritmo propuesto por (P. E. Nelson & David, 2021)

- Cobertura: o bien conocido como *recall*, con dicha métrica se pretende medir la cantidad de documentos recuperados entre el número de documentos relevantes en el índice de dichos documentos esto para obtener los primeros 100 resultados de las consultas. Esto permite determinar la cantidad de resultados relevantes que puede dar el operador, ya sea el propuesto o bien el que trae la máquina de búsqueda.

- F1: esta métrica utiliza los valores previamente conseguidos, como lo son la precisión y la cobertura, ya que se busca sacar un solo valor de estos, esto permite comparar el rendimiento combinado de la precisión y el de la cobertura (Recall) para varias consultas de acuerdo con los operadores utilizados. Dicho valor se consigue aplicando la fórmula vista en la ilustración Ilustración 11.

- Ganancia Acumulada Descontada Normalizada (nDCG): dicha métrica permite ver el ranking de los resultados, es decir, muestra en orden de relevancia los resultados

devueltos por la consulta, para poder obtener dicho valor primero se calcula el DGC regular según se ve en la ilustración Ilustración 12.

Donde rel_i es la puntuación de relevancia del documento en la posición "i". Una vez se obtiene el DCG, se puede calcular nDCG considerando el orden de ranking ideal de DCG a lo largo de los valores dados por el DCG, esto se realiza aplicando la fórmula vista en la ilustración Ilustración 13.

Adicionalmente, se analizan los resultados en otras 3 métricas de carácter más cualitativo, esto para determinar, desde un punto de vista menos subjetivo, la eficiencia de los operadores evaluados, estas métricas cualitativas son:

- Flexibilidad: se pretende con dicha métrica poder evaluar los operadores con diferentes tipos de datos tales como:
    - Texto
    - imágenes
    - Bibliotecas
    - Videos
    - Otros
- Estos tipos de datos se evalúan tanto con el operador propuesto como con el que utiliza el motor de búsqueda ElasticSearch, con el fin de poder determinar qué tan bien se adaptan dichos operadores a la hora de realizar búsquedas de consultas de diversos tipos de datos. Con el fin de poder determinar si uno u otro operador tiene una mayor capacidad respecto a los tipos de datos que maneja un operador u otro.
- Facilidad de uso: con dicha métrica se busca determinar la facilidad que tiene un desarrollador al implementar un operador de relevancia en la máquina de búsqueda ElasticSearch, esto con el fin de poder evaluar la dificultad que conlleva implementar alguno de los operadores evaluados, sin embargo, a dicha métrica no se le da un valor muy alto en la responsabilidad de los resultados, ya que esta métrica afecta al desarrollo

como tal y no directamente a los resultados de satisfacción y eficiencia que un operador de relevancia le pueda brindar al usuario final.

- Explicación de resultados: si bien los resultados de una búsqueda tienden a ser de carácter subjetivo, con base en los deseos del usuario, con esta métrica se busca minimizar esta subjetividad, ya que se pretende evaluar qué tan bien comprende el usuario final que los resultados que recibió de una consulta “q”, están en el ranking que determinó el operador con base en la relevancia que este le dio a los documentos de los resultados, por ende, se pretende evaluar la compresión del usuario de por qué el operador determinó que un documento es más relevante que otro, aun si en su perspectiva el consideraba a otro más relevante que el que definió el operador de relevancia. Todo esto con el fin de evaluar, de una manera imparcial, los resultados, y tratar de que la subjetividad no sea el factor más determinante en los resultados de la consulta o búsqueda.

Todas estas métricas por utilizar permitirán determinar, de cierta manera, el valor numérico y cualitativo, si el resultado de una consulta es el más eficaz y eficiente, aunque se dice de cierta manera, ya que esta se ajusta siempre a una opinión subjetiva, aun así, dichas métricas utilizadas a lo largo del campo de la recuperación de información permiten determinar la eficiencia del resultado de una consulta. Por otro lado, se pretende dotar de cierta responsabilidad a estas métricas por separado en el resultado de una búsqueda con el fin de determinar cuál es la métrica utilizada a la cual se le debe tomar más “importancia" y la que permite determinar la relevancia de un resultado utilizando un operador de relevancia u otro de una manera imparcial. Esta responsabilidad de las métricas en el resultado de la búsqueda se va a repartir de la manera indicada en la sección 4.3.3.

5 Análisis del Diagnóstico

Como se explicó en la sección 3.1.1, las consultas realizadas con el Core de Lucene utilizan al final un método de búsqueda del Index Searcher, el cual utiliza una Query para realizar dicha búsqueda. Estas instancias de query pueden ser creadas de diferentes maneras e igual existen diversos tipos de querys. En esta sección se pretenden ver y analizar algunos de los tipos de Query más comunes utilizadas en el motor de búsqueda ElasticSearch y aquellas que vienen de manera integrada en el Core de Lucene, esto con el fin de entender cómo se crea una Query y bajo qué términos se pueden crear, con el fin de explicar la creación de las querys y cómo estas funcionan dentro de un motor de búsqueda.

5.1 Term Query

La manera más básica de buscar en un índice es a través de un término en específico, ya que el término es la parte indexada más pequeña y está construido por un nombre para el campo y un valor de texto, es decir, el término está compuesto por este par de información. (McCandless et al., 2010)

Este tipo de query devuelve resultados de documentos, los cuales contienen un término exacto en algún campo específico, suele usarse de manera recurrente para búsquedas de documentos de valores precisos, como puede ser el nombre de una persona o bien su ID. A pesar de que este tipo de Query es útil para buscar términos de manera precisa, no es recomendable utilizarlo para realizar búsquedas en campos de texto, debido a que el "Term Query" no analiza el término de la búsqueda, solo busca el término exacto que este proporciona, por ende, puede llegar a ser deficiente al buscar campos de texto.

Ejemplo de cómo se construye un "Term Query":

```
Term t = new Term ("contents", "java");
```

De esta manera se construye un término para la consulta, este está compuesto de dos parámetros importantes.

- Field: hace referencia al campo que desea buscar, en el ejemplo anterior este sería el "contents".

- Value: este contiene el valor o término que se desea encontrar en el campo que se indicó en el primer parámetro, en el caso del ejemplo sería “java”. Es importante que, para que la consulta pueda devolver un documento, este término sea exacto y coincida con el valor del campo, además se debe tener cuidado ya que este parámetro es sensible a mayúsculas y minúsculas, por ende, el término debe ser escrito de la manera más precisa posible.
- Boost: es un parámetro opcional que se puede agregar, el cual permite aumentar o disminuir la relevancia de la consulta, posee un valor predeterminado de 1.0, de manera que un valor entre 0 y 1.0 disminuye su relevancia y un valor mayor a 1.0 la aumenta, esto puede llegar a ser útil para ajustar la relevancia de búsquedas que tienen 1 o más consultas.

Ahora bien, una vez creado el término se realiza la consulta de la siguiente manera:

```
Query query = new TermQuery(t);
```

Dicha consulta devolverá, utilizando el Term Query, todos aquellos documentos que contengan la palabra “java” en el campo de contenido.

5.2 Match Query

Es el tipo de consulta estándar, la cual realiza búsquedas de texto completo, incluidas aquellas con opciones de coincidencias aproximadas, devuelve aquellos documentos que coincidan con el texto, un número, una fecha o un valor booleano. Es importante tomar en cuenta que dicha consulta analiza el texto proporcionado para la búsqueda antes de una coincidencia, esto quiere decir que este proceso de análisis construye una consulta booleana del texto proporcionado, por ello se dice que Match Query es una consulta de tipo booleana.

(Elasticsearch documentación https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-match-query-phrase.html)

Ejemplo de la estructura de Match Query:

```
GET /_search
```

```
{
  "query": {
    "match": {
      "message": {
        "query": "this is a test"
      }
    }
  }
}
```

Ilustración 36 Ejemplo de la estructura Match Query

Fuente: Elaboración propia

Esta estructura está compuesta por diversos parámetros, e incluso puede a llegar a tener más, esto dependerá de que tan compleja se requiere hacer la búsqueda mediante esta consulta. Algunos de estos parámetros son:

(Elasticsearch documentación https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-match-query-phrase.html)

- Query: en este campo se indica el texto, número, fecha o valor booleano que se desee buscar. Se debe recordar que el Match Query analiza los textos antes de realizar la búsqueda, ya que Match Query puede realizar búsquedas también en campos de texto para tokens ya analizados en lugar de solo buscar el término exacto como hace Term Query.
- Analyzer: parámetro opcional, el cual convierte el texto de la Query en un token para el campo, si no posee un analyzer por defecto se le asigna el analyzer del índice.
- Auto_generate_synonyms_phrase_query: parámetro opcional booleano, si el mismo lleva valor true (que es su valor por defecto) crea de manera automática match para sinónimos de varios términos.
- Fuzziness: parámetro opcional que permite indicar la distancia de edición para una coincidencia.
- Max expansions: parámetro opcional que indica el número de términos máximos a los que se expandirá una búsqueda. Por defecto su valor es 50.

- Prefix length: parámetro opcional que permite indicar el número de caracteres iniciales que no se cambiaron para la coincidencia aproximada.
- Fuzzy_transpositions: parámetro opcional booleano que indica si existen transposiciones de dos caracteres (ab -> ba) para las coincidencias aproximadas.
- Fuzzy rewrite: parámetro opcional que permite reescribir la consulta.
- Leniet: parámetro opcional booleano que indica si se pueden ignorar errores basados en formato, es decir, escribir texto en un campo numérico.
- Operator: parámetro opcional que utiliza lógica booleana para interpretar texto en términos de OR o de AND. Ejemplo: si el valor de la Query es “capital de Costa Rica” y se utiliza el operador OR que es el valor por defecto, la búsqueda será interpretada como “capital o Costa Rica” a la hora de realizar la búsqueda. Caso contrario, si se utiliza el operador AND para la misma búsqueda, ya que interpretará “capital y Costa Rica” por lo que será más sencillo y rápido encontrar realmente lo que se está buscando.
- Minimum should match: parámetro opcional que permite indicar el número de cláusulas que deben coincidir para devolver un resultado.
- Zero term Query: parámetro opcional que indica si no se devuelve ningún documento que se eliminen todos los tokens creados por el analyzer, puede tener dos valores:
    - None: su valor por defecto no devuelve ningún documento si el analyzer elimina los tokens.
    - All: devuelve todos los documentos, posee una función similar a la consulta match all.

Cada uno de los parámetros mencionados puede ser utilizado para mejorar o hacer aún más precisa la búsqueda con base en este tipo de consulta, todo esto dependerá de qué tan específica se desea realizar la búsqueda de un documento y tal como se indica en el inicio, este tipo de consulta es estándar, por ende, es una de las más utilizadas en ElasticSearch.

5.2.1 Match Query como un operador booleano

Es importante destacar que internamente el operador match se convierte a un operador booleano, donde se utiliza un operador ya sea “AND” o “OR” para realizar la consulta, tal y como se describe anteriormente. La consulta como tal se reescribe tal y como se describe en la:

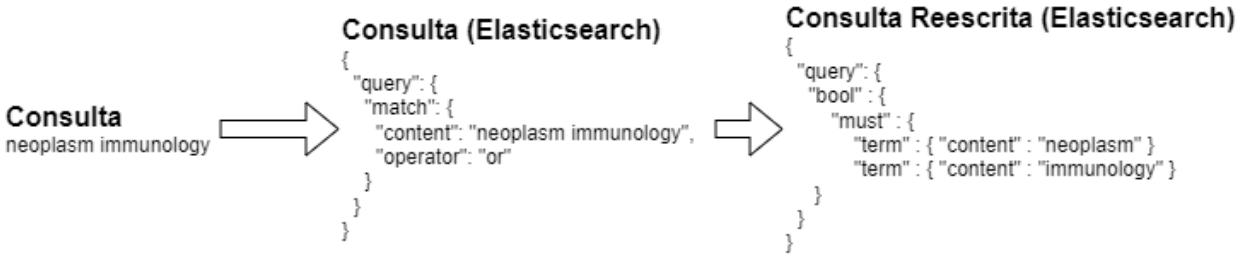


Ilustración 37 Reescritura del operador "match" de Elasticsearch a una consulta booleana (simplificado).

Fuente: Elaboración propia basado en a la documentación de Elasticsearch

Como se puede observar, en el caso de utilizar un operador “OR”, Elasticsearch toma los términos de la consulta y los convierte en consultas de términos individuales unidos por un operador booleano. En este caso es una condición booleana “must”, la cual significa que los términos “deberían” estar en los documentos, y de estar, aportan al puntaje de este.

Este detalle es importante para el presente proyecto dado que se van a comparar los operadores creados contra las búsquedas tradicionales de Elasticsearch. Como se puede observar, es posible replicar el funcionamiento de la búsqueda de los “match query” utilizando una consulta booleana, que realice un “or” entre los términos a buscar.

6 Propuesta de Solución

6.1 Resumen de la solución

La propuesta de la patente de (P. E. Nelson & David, 2021) consiste en crear un puntaje normalizado de 0 a 1 para cada paso de la ejecución de la consulta, incluyendo el puntaje final que recibe cada documento.

El problema que trata de solucionar esta patente es que la mayoría de los motores de búsqueda producen una puntuación numérica no normalizada para los documentos como resultado de una consulta. Estas puntuaciones pueden variar significativamente de una consulta a otra, de un motor de búsqueda a otro, y de un conjunto de datos a otro. Los sistemas existentes no se han preocupado por la comparabilidad o la normalización de estas puntuaciones en diferentes búsquedas, campos o conjuntos de datos.

Con respecto a la facilidad de comparar puntajes, se dan casos en los que se necesitan búsquedas federadas que consisten en combinar los resultados de diferentes máquinas de búsqueda. En estos casos los puntajes asignados a los resultados no son directamente comparables, por lo que es necesario contar con algún mecanismo para poder ordenar los resultados que se presentan a los usuarios – en (Manmatha & Sever, 2002) se propone usar la distribución de los puntajes de los resultados para normalizarlos, por ejemplo. La patente en cuestión hace otra propuesta que realiza la normalización mientras se ejecuta la consulta, y no como un paso de procesamiento posterior de la misma.

La patente describe el proceso de la normalización en la página 1 de (P. E. Nelson & David, 2021):

1. Recibir, desde un dispositivo cliente. una consulta especificada por el usuario
2. Construir la consulta como un árbol de operadores que incluyan cero o más operandos
3. Producir, para cada uno de los operandos, un peso que indique qué tan valiosa o confiable es una determinación de relevancia de cada uno de los operadores para los hermanos en el árbol de consulta, donde el peso se calcula a partir de métricas que son independientes de un documento.

4. Normalizar los pesos para los hermanos de la consulta con un operador padre, aplicando una fórmula de normalización
5. Producir, para cada uno de los operadores. una puntuación, con un valor de cero a uno, así como para el documento, que representa la relevancia del documento para la consulta especificada por el usuario, en la que cada uno de los operadores calcula la puntuación en función de la información disponible de los hijos de cada uno de los operadores y de un índice de la máquina de búsqueda
6. Aplicar el árbol de consulta al documento como un todo, así como a las posiciones dentro del documento
7. Determinar una puntuación normalizada que indique qué tan relevante es el documento para la consulta en función de la aplicación del árbol de consulta al documento
8. Realiza una o más acciones en función de la puntuación.

Dentro de la patente se especifican varios métodos matemáticos y estadísticos para lograr los puntajes normalizados, como se pueden encontrar en la sección 28 de (P. E. Nelson & David, 2021):

- Un modelo de normalización utilizando promedios ponderados
- Un modelo de normalización utilizando magnitud euclideana
- Un modelo de normalización utilizando Softmax
- Un modelo de normalización utilizando máximos
- Un modelo de normalización utilizando la función sigmoide
- Un modelo de normalización utilizando la función logística semitruncada
- Un modelo de normalización utilizando la distribución acumulada en escala logarítmica
- Un modelo de normalización utilizando regresión logística
- Un modelo de normalización utilizando sigmoide truncado

Por otro lado, también se especifican las fórmulas que permiten o bien ayudan a realizar la normalización en el puntaje del resultado del operador, en la sección 28 de (P. E. Nelson & David, 2021). De estas fórmulas se utilizaron 4 de ellas siendo las siguientes:

- Max: esta función permite realizar una normalización de los pesos de un término de una consulta dada, respecto al peso máximo del total de los pesos de los términos, lo cual posibilita comparar dicho peso respecto al máximo de los pesos. Esta función está dada por la siguiente fórmula:

$$W_{norm}(t) = \frac{W(t)}{\max\limits_{i=1\ to\ n} W(t_i)}$$

Ilustración 38 Fórmula de la función Max. Fuente (P. E. Nelson & David, 2021)

- Sum: esta función realiza la normalización de los pesos de los términos, la cual permite tener una visualización porcentual del peso conservando su tamaño relativo. Esta normalización se realiza al dividir el peso de un término entre la suma total de los pesos. Esto permite tener un peso comparable de manera porcentual a los pesos de los demás términos(t). Esto quiere decir, que, si el peso de un término es de valor considerable, este abarcara mayor porcentaje del total de los pesos. Esta función está dada por la siguiente fórmula:

$$W_{norm}(t) = \frac{W(t)}{\sum_{i=1}^{n} W(t_i)}$$

Ilustración 39 Fórmula de la función Sum. Fuente (P. E. Nelson & David, 2021)

- Sigmoide: esta función permite normalizar el valor de los términos (term) con el objetivo de realizar una comparación eficiente de los puntajes de los resultados de los documentos. Dicha función realiza la normalización utilizando diferentes parámetros, entre los que se encuentran:

- t = es el resultado de los términos no normalizados.
- max: el máximo valor esperado de los términos.
- S: es el parámetro el cual funge como una constante de escala que determina la pendiente de curva.

Estos parámetros permiten tener un rango entre dos valores 0, 1 de los términos, lo que permite comparar los resultados, dado que estos estarán entre los dos valores mencionados, siendo los valores más altos los cercanos a 1.

La función está dada por el siguiente criterio:

$$Score(t) = \frac{1}{(1 + e^{s \cdot \left(\frac{-t}{max} + 0.5\right)})}$$

Ilustración 40 Fórmula de la función Sigmoide. Fuente (P. E. Nelson & David, 2021)

La grafica de dicha función es la siguiente con valores de max =2 y s=7:

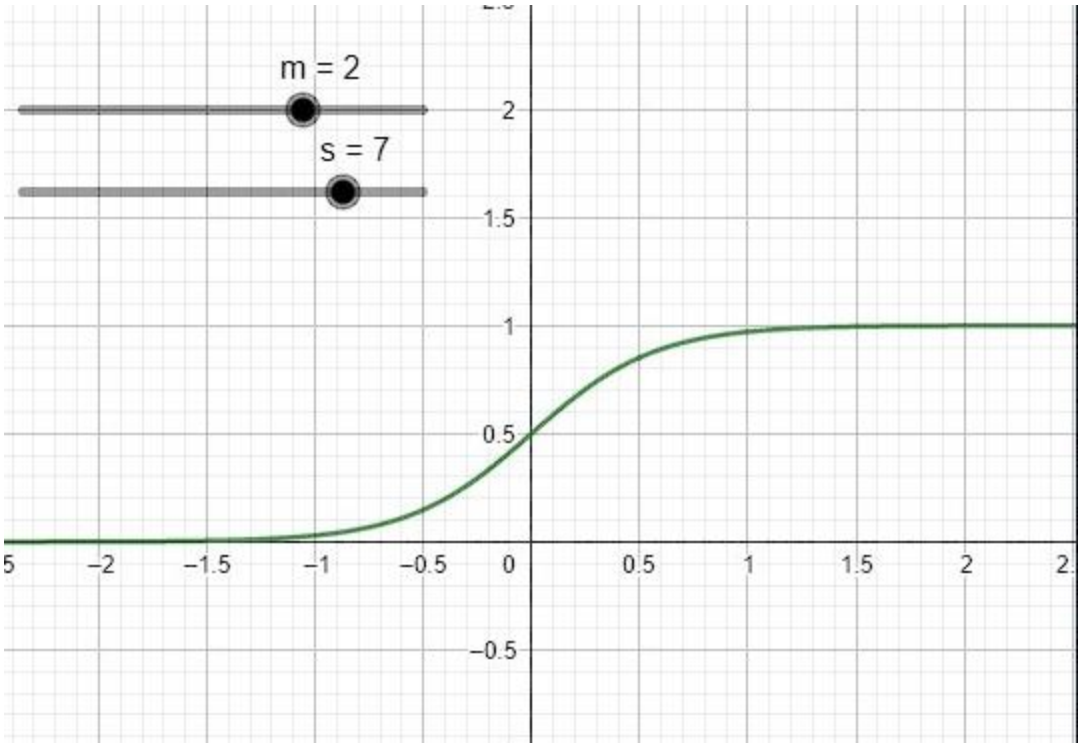


Ilustración 41 Gráfico de la función Sigmoide. Fuente Elaboración propia.

Estos resultados pueden optimizarse al variar los parámetros, donde, al incrementar “s” incrementará la pendiente de la función. Lo cual indica, que mientras mayor es la diferencia de los términos (t) originales mayor será la diferencia de los términos normalizados. Ahora bien, para el parámetro ”max”, al aumentarlo incrementa el rango de valores de entrada de los términos originales, ya que este parámetro determinar el valor máximo esperado de los términos.

- Sigmoide truncado (Half-Sigmoide): al igual que la función sigmoide esta permite normalizar el valor de los términos (t) utilizando los mismos parámetros; sin embargo, permite hacer énfasis en los valores más altos, utilizando la mitad de la función sigmoide de manera que cubra un rango de 0 al máximo, es decir, dejará por fuera valores de términos (t) menores a 0. Por ende, brindará únicamente resultados de los documentos más relevantes. Esta función está dada por el siguiente criterio:

$$Score(t) = \frac{2}{(1 + e^{s \cdot \left(\frac{-t}{max}\right)})} - 1$$

Ilustración 42 Fórmula de la función Sigmoide Truncada. Fuente (P. E. Nelson & David, 2021)

Al igual que la función anterior, los resultados normalizados de esta fórmula pueden ser optimizados con la variación de los parámetros “max” y “s”. Dicha función se visualiza de la siguiente manera con valores de max = 2 y s=7:

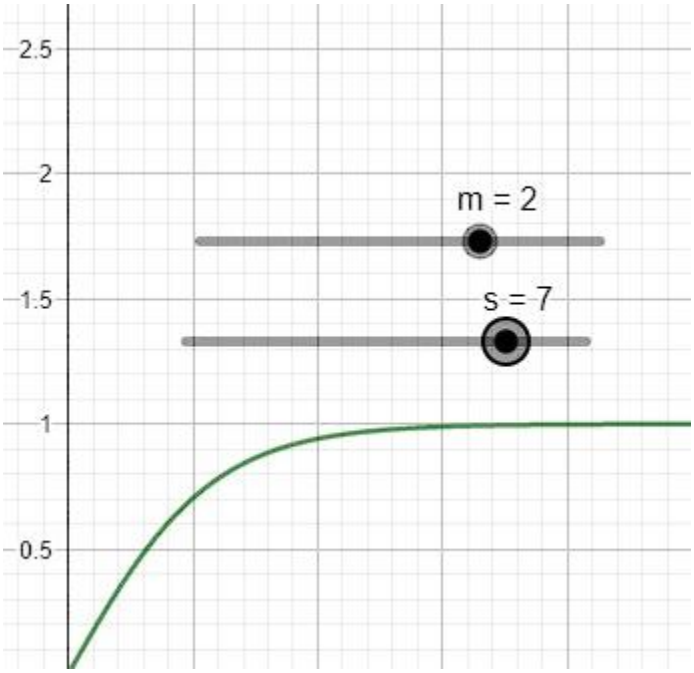


Ilustración 43 Ilustración 39 Gráfico de la función Sigmoide Truncada. Fuente Elaboración propia.

Adicionalmente, con el fin de lograr esta normalización en todos los pasos, en la sección 29 de (P. E. Nelson & David, 2021) se mencionan los siguientes operadores, de los cuales en el presente trabajo solamente se implementó el operador OR tal y como se elabora en la siguiente sección.

- Operador AND: Ejecuta una búsqueda booleana donde todas de las cláusulas debe de encontrarse en el documento para que se considere relevante para la consulta.
- Operador OR: Ejecuta una búsqueda booleana donde alguna de las cláusulas debe de encontrarse en el documento para que se considere relevante para la consulta.
- Operador PHRASE: Permite buscar términos que se encuentre de forma consecutiva en los documentos
- Operador BOOST: Aplica un factor multiplicativo al puntaje de la consulta
- Operador CONSTANT: Retorna un puntaje constante para la consulta
- Operador FILTER: Se ejecuta como parte de la consulta y los resultados deben de satisfacer las condiciones de esta parte de la consulta, pero no aporta al puntaje de relevancia de la misma

6.2 Alcance de la implementación

Debido al amplio alcance de la solución propuesta en (P. E. Nelson & David, 2021) que compete a esta investigación, y resumida en la sección 6.1, solamente se va a evaluar un subconjunto de la solución planteada.

Tal y como se describe en la sección 5.2.1, es posible replicar el funcionamiento de la consulta del operador de Elasticsearch “match query” utilizando operadores que ejecuten la consulta por un término, y juntándolos en un operador booleano que realice un “OR” lógico entre ellos. Debido a esto se limitó la implementación solamente a los siguientes operadores:

- Búsqueda por término: Se desarrolló un operador de Elasticsearch llamado *normTerm* que funciona de forma similar al operador *term* de Elasticsearch, con la diferencia que normaliza las diferentes partes del cálculo del puntaje de acuerdo a la configuración provista para crear un puntaje entre 0 y 1 tal y como se menciona en (P. E. Nelson & David, 2021).
- Operador booleano OR: Se desarrolló un operador de Elasticsearch llamado normOr, que funciona de forma similar al operador “boolean must” de Elasticsearch. Es decir, cada cláusula del operador “debería” estar en los documentos, y de estar, aportan al puntaje de este. Este operador acepta cláusulas que deben ser de tipo *normTerm.*

Adicionalmente, de los 9 modelos de normalización propuestos para los puntajes, se desarrollaron las siguientes 4. Se considera que con estos 4 modelos es posible validar el concepto de los operadores y poder ejecutar las consultas:

- Un modelo de normalización utilizando promedios ponderados
- Un modelo de normalización utilizando máximos
- Un modelo de normalización utilizando la función sigmoide
- Un modelo de normalización utilizando sigmoide truncado

Finalmente, a la hora de realizar la implementación, fue necesario considerar los diferentes parámetros a pasar a los operadores y la forma de enviar las consultas a Elasticsearch, tal y como se elabora a continuación.

6.2.1 Operador normTerm

El Operador normTerm, como se describió anteriormente, es la unidad básica para buscar un término dentro del índice de Elasticsearch. En la siguiente figura se muestra el ejemplo de una consulta de Elasticsearch con este operador.

```
{

  "query":{
    "normTerm":{
      "fieldName": "full_text",
      "term": "obeyed",
      "idfField":"full_text",
      "docData":"BM25TF:k=1;b=0.75",
      "docDataNorm":"HALF-SIGMOID;s=8",
      "max":"5",
      "weight":"BM25IDF",
      "power":"1"
    }
  }
}
```

Ilustración 44 Ejemplo de uso del operador normTerm desarrollado en ElasticSearch. Fuente: Elaboración Propia.

El operador recibe los siguientes parámetros:

1. **weight**: Este es el peso del término de consulta. Puede ser la fórmula IDF "Clásica" estándar de Lucene, la versión BM25 de IDF, o cualquier valor constante de punto flotante.

2. **fieldName**: Este es el campo en el que se va a buscar.

3. **idfField**: Este es el campo utilizado para calcular el IDF del término. Los campos más grandes (como el cuerpo o contenido) proporcionan un IDF más confiable que los campos más pequeños (como el título o el autor).

4. **docData**: Esta es la fórmula para extraer los datos del documento para el token. Puede ser la frecuencia de términos (TF), el componente TF de la fórmula BM25 (BM25TF), o un valor constante.

5. **power**: Este parámetro eleva el valor de docData a una potencia especificada.

6. **docDataNorm**: Esta es la fórmula de normalización aplicada a docData. Puede ser la función sigmoide (SIGMOID) o la función sigmoide truncada (HALF-SIGMOID). En ambos

casos se puede configurar un parámetro “s” que indica la pendiente de la función sigmoide. En caso de no especificarse el valor de “s” será de 8.

7. **max**: Esta es una constante que indica el valor máximo esperado del valor de docData utilizado para escalar la función docDataNorm.

6.2.2 Operador normOr

El operador normOr recibe unos parámetros de configuración y una lista de cláusulas de tipo normTerm entre las cuales realiza un “OR” booleano para determinar si un documento es relevante para la búsqueda o no, y normaliza los puntajes de las sub-cláusulas. En la siguiente figura se muestra el ejemplo de una consulta de Elasticsearch con este operador.

```
{
    "query": {
        "normOr": {
            "weight": 2.0,
            "threshold": 0.5,
            "minClauses": 2,
            "inputWeightNorm": "SUM",
            "function": "AVG",
            "clauses": [
                {
                    "normTerm": {
                        "fieldName": "content",
                        "term": "wing",
                        "idfField": "content",
                        "docData": "TF",
                        "docDataNorm": "SIGMOID",
                        "max": "5",
                        "weight": "IDF",
                        "power": "1"
                    }
                },
                {
                    "normTerm": {
                        "fieldName": "content",
                        "term": "turbulence",
                        "idfField": "content",
                        "docData": "TF",
                        "docDataNorm": "SIGMOID",
                        "max": "5",
                        "weight": "IDF",
                        "power": "1"
                    }
                }
            ]
        }
    }
}
```

Ilustración 45 Ejemplo de uso del operador normOr desarrollado en ElasticSearch. Fuente: Elaboración Propia.

El operador recibe los siguientes parámetros:

1. **weight**: Este es el peso que se aplicará a la salida del operador y que se pasará a su operador padre.
2. **minClauses**: El número mínimo de cláusulas **normTerm** que deben coincidir con el documento (con puntuación no cero) para que sea considerado como relevante para la consulta.
3. **threshold**: Solo los documentos que coincidan por encima del umbral especificado son devueltos (por defecto es 0.0).
4. **inputWeightNorm**: Especifica la función para normalizar los pesos de entrada (los pesos de las subcláusulas). Puede ser SUM o MAX en esta implementación, o un simple número de punto flotante que se multiplicará por cada uno de los pesos.
5. **function**: Especifica la función utilizada para calcular la puntuación final de este operador. Puede ser AVG, MAX o SUM.
6. **clauses**: Consiste en una serie de operadores NormTerm que serán evaluados dentro del NormOr

6.3 Implementación

Los operadores de Lucene en Elasticsearch consisten en las siguientes partes:

- Clase *Query*: Es la clase que representa el "punto de entrada" del operador. Hereda de la clase org. apache. lucene.search.Query y r representa la consulta del usuario, independientemente del contenido del índice de Lucene
- Clase *Weight*: Clase que calcula y almacena los valores del índice que son constantes para cada documento, como por ejemplo el IDF. Hereda de la clase org.apache.lucene.search.Weight
- Clase *Scorer*: La clase Scorer es la que retorna los documentos y los puntajes, o *scores*, de cada documento. Hace los cálculos a nivel de documento. Hereda de la clase org.apache.lucene.search.Scorer.

- Clase *Similarity*: Si bien normalmente no es necesario realizar una implementación de una clase de similaridad, para la implementación pertinente es necesario. La similaridad es la que finalmente determina el puntaje de un documento con respecto a una consulta, realizando las operaciones matemáticas pertinentes. Lucene cuenta con diversas clases de similaridad, como una que utiliza TF/IDF clásico (*org.apache.lucene.search.similarities.ClassicSimilarity*), otra que utiliza Okapi BM5 (*org.apache.lucene.search.similarities.BM25Similarity*) y otras. En nuestro caso dependiendo de cómo se configure el operador, es necesario calcular la similaridad de forma diferente, por lo que fue necesario crear una clase de similaridad que encapsula instancias de las dos similaridades anteriormente mencionadas y las utiliza de acuerdo con cómo se requiera por la configuración dada.

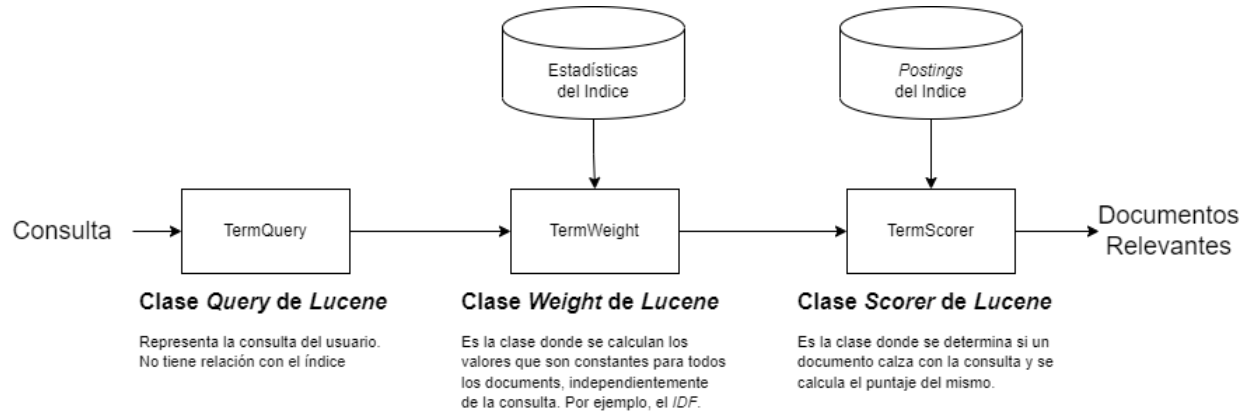


Ilustración 46 Estructura de las clases de Lucene involucradas en la ejecución de la consulta.Fuente: adaptado de (P. Nelson, 2019)

Para la implementación se crearon dos operadores más las clases necesarias para su operación. Los operadores se implementan en las clases **NormTermQuery** y **NormOrQuery**.

6.3.1 NormTermQuery

La clase **NormTermQuery** extiende la clase **Query** de Lucene. La clase implementa una lógica de puntuación y normalización para las búsquedas de términos en Elasticsearch. Es la clase central que realiza la normalización para cada término de la consulta, que permite que luego estos

puntajes sean combinados y normalizados en otros operadores como se describe más adelante en el documento.

El siguiente diagrama muestra las clases principales involucradas en la implementación de esta funcionalidad.

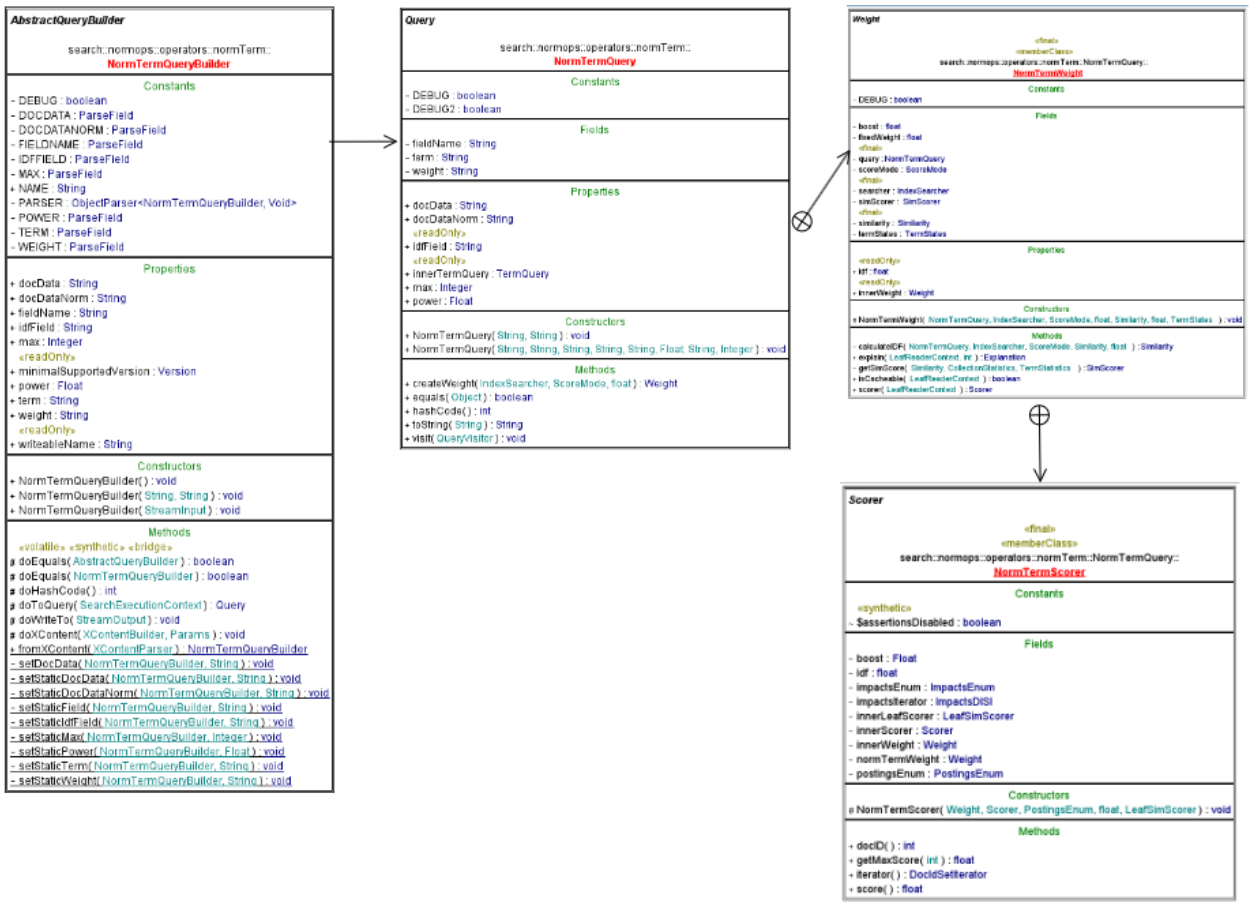


Ilustración 47 Diagrama de clases para la clase NormTermQuery. Mostrando las clases implementadas para su funcionalidad. Fuente: Elaboración Propia.

Las clases implementadas tienen la siguiente funcionalidad:

search.normops.operators.normTerm.NormTermQueryBuilder: esta clase es el punto de entrada para poder utilizar el operador en Elasticsearch. Extiende la clase de Elasticsearch ***org.elasticsearch.index.query.AbstractQueryBuilder***. Las funciones que representan la funcionalidad principal de la clase son las siguientes:

- Método **fromXContent**() es llamada desde los componentes de Elasticsearch que realizan el análisis del texto ingresado en la consulta. Para esto, Elasticsearch toma la entrada en texto dada por el usuario y la convierte en un formato inspirado en JSON llamado **XContent**. (Elasticsearch, 2023c). Esta función toma el contenido del **XContent** y usa la información para asignar los valores a las diferentes propiedades que eventualmente serán usadas para crear el objeto de la consulta.
- Método **doToQuery**() toma la información guardada y procesada en la clase y crea una instancia del operador de la consulta **NormTermQuery** con estos parámetros. Algo importante a destacar es que este método recibe el SearchExecutionContext, el cual contiene la información del índice de Elasticsearch, para que el operador pueda obtener información sobre los campos a procesar, los analizadores a usar con los términos, etc. En esta implementación en particular se utiliza un analizador que realiza una lematización, llamado *cran_stemmed*.

search.normops.operators.normTerm.NormTermQuery: Es la clase principal en donde se procesa la consulta y normaliza la puntuación de los documentos en las búsquedas de un término, utilizando otras clases que se describen en esta sección. En la sección se detalla la estructura de estos tipos de clases, con un **Query**, un **Weight** y un **Scorer**. Algunos componentes relevantes de esta clase son los siguientes:

- Propiedades: Se definen varias propiedades (**weight, idfField, fieldName, docData, power, docDataNorm, max, term, innerTermQuery**) que contienen información sobre el término de búsqueda y cómo se debe calcular la puntuación, tal y como se explican en la sección 6.2 Alcance de la implementación. A estas propiedades se les asignan los valores provistos en la consulta por medio de los métodos *get* y *set* correspondiente.
- Propiedad **innerTermQuery**: Esta propiedad en particular es importante para la lógica del operador. Es de tipo **org.apache.lucene.search.TermQuery**, y es usado por el operador para realizar la funcionalidad de búsqueda como tal. Sin embargo, en el método **createWeight()** se crea un peso donde se normalizan los puntajes dependiendo de la configuración, por lo que no se utilizan los puntajes que retorna el **TermQuery**.

- Método **createWeight()**: Este método contiene parte importante de la lógica de normalización. Es responsable de crear un objeto tipo **Weight** que se utiliza para calcular las puntuaciones de los documentos en la búsqueda. Este método considera diferentes modelos de similitud (Similarity) como IDF clásico, BM25IDF o un valor flotante personalizado, dependiendo de la configuración provista por el usuario, utilizando la clase **NormOpsSimilarity** que se describe más adelante. El método devuelve un nuevo objeto de tipo **NormTermWeight**, que es que tiene en cuenta la similitud seleccionada, el término de búsqueda y otros factores relevantes para poder calcular el puntaje de los documentos.

search.normops.operators.normTerm.NormTermQuery.NormTermWeight: es una subclase estática de **org.apache.lucene.search.Weight** que implementa la lógica para calcular puntuación normalizada de componentes de las fórmulas como el IDF. Algunos componentes relevantes de esta clase son los siguientes:

- Método **calculateIDF**(): Este método calcula el IDF basado en la configuración que se le pasa al **NormTermQuery** en los campos **idfField**, **docData**, y **docDataNorm**. Este IDF una vez calculado para una consulta es un valor fijo para todos los documente – las subclases de **Weight** realizan los cálculos de valores que son constantes para toda la colección, como se explica al inicio de la presente sección 6.3. Para los cálculos de este IDF se utiliza la clase **NormOpsSimilarity** de tal forma que ya se tiene el IDF y no lo calcula de nueva como sucedería con las clases de similaridad que incluye Lucene.
- Método **scorer**(): Este método crea un nuevo **NormTermScorer**, que es una subclase de **Scorer** e implementa la lógica de la puntuación a asignar a cada documento de la consulta. A esta clase se le pasa la similaridad y toda la configuración pertinente para llevar a cabo la normalización.

search.normops.operators.normTerm.NormTermQuery.NormTermScorer: es una subclase estática de **org.apache.lucene.search.Scorer** que implementa la lógica de asignar la puntuación normalizada a documentos individuales. Algunos componentes relevantes de esta clase son los siguientes:

- Constructor: Algo importante de esta implementación es que en el constructor se le asigna un **LeafScorer**, que es la clase de Lucene que se encarga de asignar los puntajes. Pero en el método **scorer()** de la clase **NormTermWeight**, se crea un **LeafScorer** con el scorer de la similaridad **NormOpsSimilarity.NormOpsSimScorer**, (descrita más adelante). Al delegarle la creación de puntaje no es necesario implementar los detalles de bajo nivel de Lucene, sino que el mismo Lucene se encarga de esto, pero utilizando la similaridad que se creo en la clase Weight.
- Método **score**(): es el método principal de la clase, y es el método que asigna el puntaje a un documento. En este caso utiliza una propiedad llamada **innerLeafScore,** descrita anteriormente para asignar los puntajes a los documentos.

6.3.2 NormOrQuery

La clase NormOrQuery extiende la clase Query de Lucene. La clase implementa una lógica de puntuación y normalización para las búsquedas en Elasticsearch que combinan múltiples NormTermQueries.

Ilustración 48 Diagrama de clases para la clase NormOrQuery. Mostrando las clases implementadas para su funcionalidad. Fuente: Elaboración Propia.

Las clases implementadas tienen la siguiente funcionalidad:

search.normops.operators.normOr.NormOrQueryBuilder: como se mencionó anteriormente en el operador **NormTermQuery**, esta clase es el punto de entrada para poder utilizar el operador en Elasticsearch. Extiende la clase de Elasticsearch ***org.elasticsearch.index.query.AbstractQueryBuilder***. Las funciones que representan la funcionalidad principal de la clase son las siguientes:

- Constructor: El constructor recibe una instancia de la clase **org.elasticsearch.common.io.stream.StreamInput**. En estos constructores de los operadores siempre es caso, y no hay nada que resaltar. Sin embargo, en este caso estamos recibiendo una lista de operadores **NormTermQuery**, los cuales no conoce Elasticsearch al ser una extensión propietaria. Eso significa que es necesario especificar explícitamente que la lista de cláusulas del operador **NormOrQuery** son de tipo

NormTermQuery para que puedan ser interpretadas correctamente. Esto se realiza en el constructor.

- Método **fromXContent**(): Al igual que con el operador NormTermQuery, este método es llamado desde los componentes de Elasticsearch que realizan el análisis del texto ingresado en la consulta. Aquí se interpretan las propiedades ingresadas en formato XContent y se almacenan para asignar los valores a las diferentes propiedades posteriormente.
- Método **doToQuery**(): Una vez más, al igual que con los operadores anterior, toma la información guardada y procesada en la clase y crea una instancia del operador de la consulta **NormOrQuery** con estos parámetros, así como los **NormTermQuery** de las cláusulas.

search.normops.operators.normOr.NormOrQuery: Es la clase principal en donde se procesa la consulta y normaliza la puntuación de las cláusulas de tipo **NormTermQuery** de las cuales consiste este operador. Al igual que con todos los operadores de Lucene, existe una clase **Query**, un **Weight** y un **Scorer**. Algunos componentes relevantes de esta clase son los siguientes:

- Propiedades: Algunas propiedades importantes de la clase son las siguientes:
    - **clauses**: Lista de consultas de tipo **NormTermQuery** que forman parte de la consulta principal.
    - **threshold**, **minClauses**, **weight**: Parámetros utilizados para la búsqueda.
    - **inputWeightNorm** y **outputNorm**: Representan la normalización a aplicar a la consulta, tomando en cuenta los puntajes generados por las cláusulas **NormTermQuery**.
    - **function**: Define la función de normalización que se va a usar.
    - **innerQuery**: Una **BooleanQuery** construida a partir de las cláusulas de la consulta. Este es una de las propiedades más importantes. La lógica necesaria para poder determinar si un documento satisface la consulta o no es altamente compleja, por lo que en esta implementación se utilizar una instancia de **org.apache.lucene.search.BooleanQuery**. Esta instancia se configura de tal

manera que las cláusulas booleanas son de tipo **org.apache.lucene.search.BooleanClause.Occur.SHOULD**, lo cual es equivalente a un operador lógico "**OR**". Y adicionalmente se crean las cláusulas que utiliza el **BooleanQuery**, que son de tipo **org.apache.lucene.search.BooleanClause,** y recibe un **Query** como parámetro – en este caso se le pasa una instancia del **NormTermQuery** que representa cada cláusula de esta forma se utiliza la lógica ya existente y probada de Lucene para realizar este tipo de consultas booleanas.

- Método **createWeight()**: Este método crea una lista de objetos de tipo **Weight** para cada consulta en la lista de cláusulas del OR – en su efecto llamando al método **createWeight**() de **NormTermQuery** con los mismos parámetros proporcionados. Luego, el método crea un **Weight** para **innerQuery**, es decir, le delega al **BooleanQuery** el cálculo de los pesos de toda la consulta. Con estas propiedades se retorna un objeto **NormOrWeight**.

search.normops.operators.normOr.NormOrQuery.NormOrWeight: es una subclase estática de **org.apache.lucene.search.Weight** que implementa la lógica para el IDF y proveer el **Scorer** para la clase **NormOrQuery**. El principal método relevante de esta clase es el método **scorer()**:

- Método **scorer**(): Este método crea un nuevo **NormOrScorer**, que es una subclase de **Scorer** e implementa la lógica de la puntuación a asignar a cada documento de la consulta. Algo importante de destacar de la implementación es que a la clase del **NormOrScorer** es necesario pasarle el **Scorer** creado por la consulta de tipo **BooleanQuery** a la cual se le delega la funcionalidad de realizar la puntuación, y también es necesario pasarle los **Scorer** de cada una de las cláusulas, para poder extraer los puntajes intermedios y normalizarlos de acuerdo a la configuración provista.

search.normops.operators.normOr.NormOrQuery.NormOrScorer: es una subclase de **org.apache.lucene.search.Scorer** que implementa la lógica de asignar la puntuación normalizada a documentos individuales. Algunos componentes relevantes de esta clase son los siguientes:

- Constructor: En el constructor se asignan las propiedades de la clase. Algo importante de destacar es que se utiliza una implementación propia del **DocIdSetIterator** para poder

avanzar los documentos que el **BooleanQuery** al que se le delega la funcionalidad de determinar si un documento cumple con la consulta o no. Esta implementación es necesaria porque Cada **NormTermQuery** que es parte de las cláusulas del **BooleanQuery** avanza entre los documentos a los que considera relevantes de forma separada, por lo que se implementó la clase **NormOpsDocIfSetIterator** para resolver este problema y que cuando el operador solicite el siguiente documento relevante retorne uno que satisfaga la condición correctamente.

- Método **score**(): El método **score()**calcula una puntuación de relevancia para los documentos utilizando los puntajes de los **NormTermQuery** de las cláusulas, y utiliza la operación que se pasa en el parámetro **function**. Esta puede ser "SUM", "MAX" o "AVG", que corresponde a la suma de los puntajes de las cláusulas, el máximo, o el promedio de las mismas.

6.3.3 NormOpsSimilarity

La clase NormOpsSimilarity extiende la clase org.apache.lucene.search.similarities.Similarity de Lucene para implementar la funcionalidad de calcular puntajes de relevancia normalizados cuando se determina si un documento es relevante para una consulta o no.

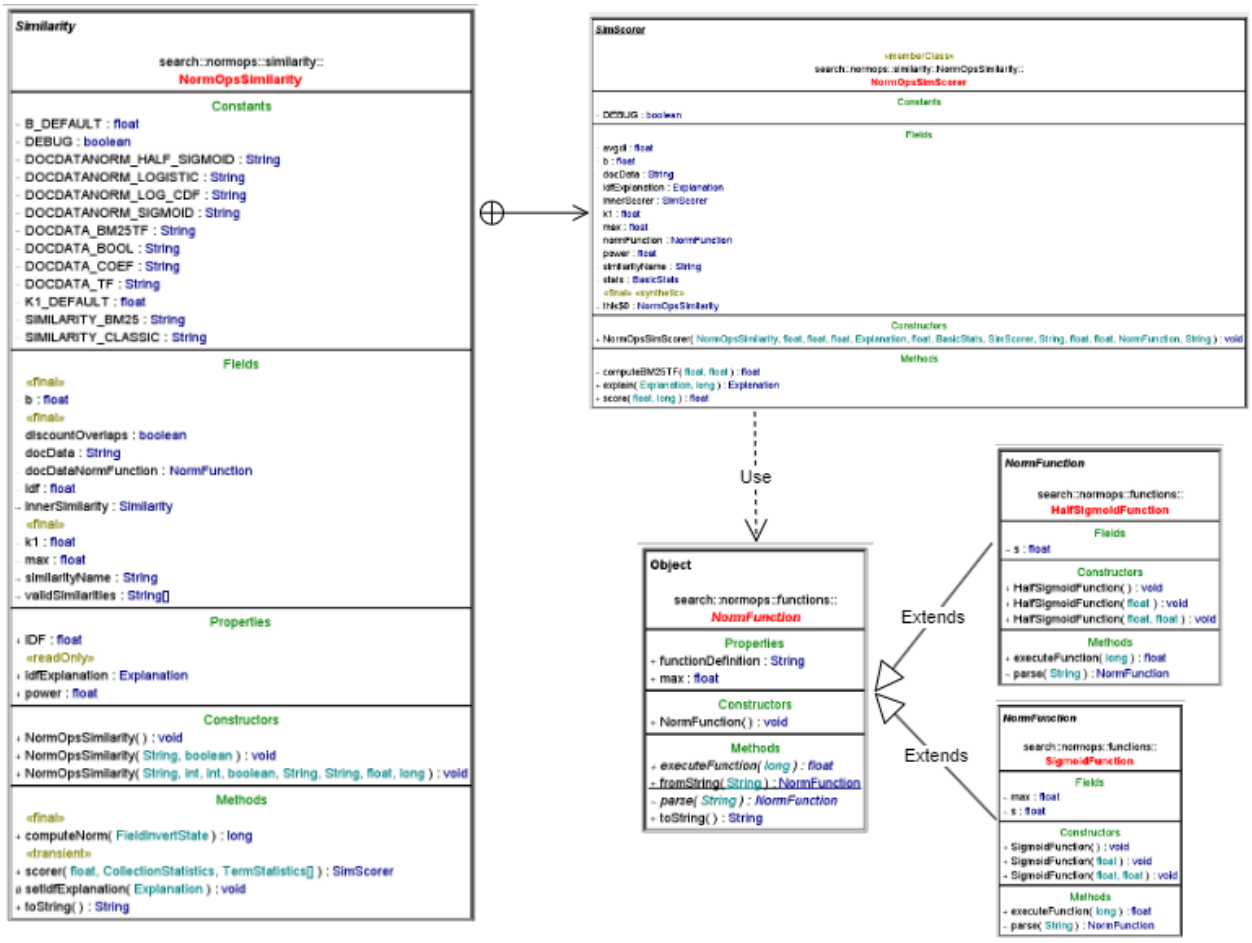


Ilustración 49 Diagrama de clases para la clase NormOpsSimilarity. Mostrando las clases implementadas para su funcionalidad. Fuente: Elaboración Propia.

Las clases implementadas tienen la siguiente funcionalidad:

search.normops.similarity.NormOpsSimilarity: como se mencionó anteriormente en el operador **NormTermQuery**, esta clase es el punto de entrada para poder utilizar el operador en Elasticsearch. Extiende la clase de Elasticsearch ***org.elasticsearch.index.query.AbstractQueryBuilder***. Las funciones que representan la funcionalidad principal de la clase son las siguientes:

search.normops.similarity.NormOpsSimilarity.NormOpsSimScorer: Extiende la clase de Lucene ***org.apache.lucene.search.similarities.SimScorer***. Las funciones que representan la funcionalidad principal de la clase son las siguientes:

- Constructor: El constructor de **NormOpsSimScorer** recibe varios parámetros para configurar el cálculo de la puntuación, incluyendo un **SimScorer** interno, el nombre de la similaridad a utilizar, y una función de normalización, entre otros.
- Método **score()**: El método **score** calcula la puntuación utilizando la frecuencia del término y la norma del documento, y aplica una de las dos fórmulas de frecuencia de término (TF), TF estándar o BM25TF, en función del valor configurado de **docData**. Posteriormente, se aplica la función de normalización a la frecuencia del término si está definida, utilizando las clases basadas en **NormFunction** (se explica más adelante).

search.normops.functions.NormFunction: Esta es la clase base que se utilizó para poder implementar las diferentes funciones de normalización de los datos de una forma flexible. Esta clase es abstracta, pero tiene dos funcionalidades principales: Por un lado, puede instanciar las subclases, y por otro define la interfaz para poder ejecutar la función de normalización sobre la frecuencia del término. Algunos de sus métodos más importantes son:

- fromString(): Método que realiza el análisis de la función que se configura para un operador, como por ejemplo “HALF-SIGMOID;s=8”. En este caso determina que la función a utilizar es la función sigmoide truncada con un parámetro “s” con valor de 8, por lo que instancia la subclase correspondiente y le pasa esa configuración.
- executeFunction(): Este método recibe un número y de acuerdo a la configuración proporcionada ejecuta la función. En la clase base es un método abstracto que debe ser implementado por las subclases.
- Las funciones implementadas en la presente investigación son las siguientes:
    - **search.normops.functions.SigmoidFunction**: Implementa la función sigmoide tal y como se describe en la sección 6.1

    - **search.normops.functions.HalfSigmoidFunction**: Implementa la función sigmoide tal y como se describe en la sección 6.1
- La intención es que en futuro sea posible agregar nuevas funciones fácilmente por medio de este mecanismo de extensión

6.3.4 NormOpsPlugin

La clase NormOpsPlugin esa utilizada para registrar los operadores desarrollados como una extensión en Elasticsearch.

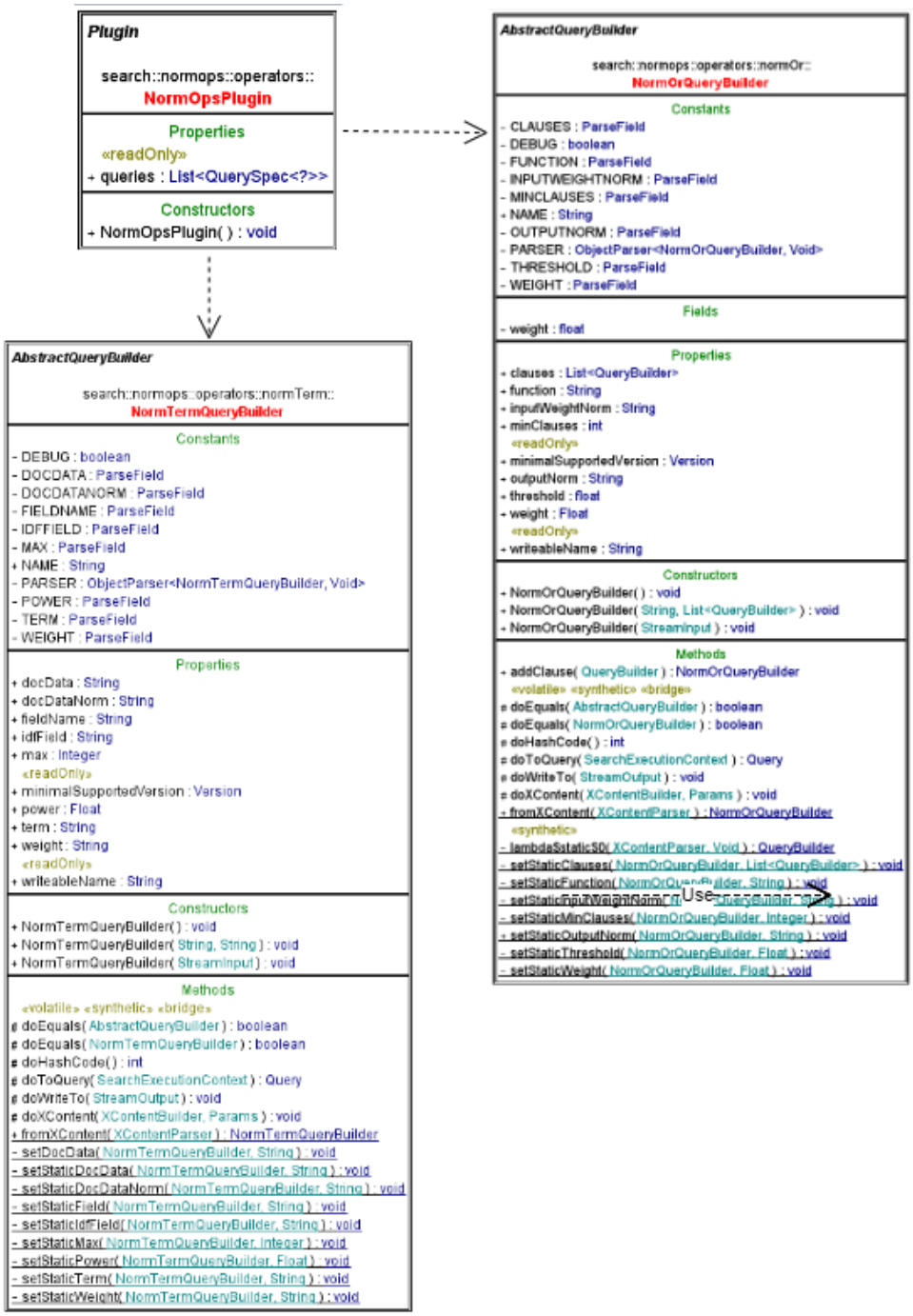


Ilustración 50 Diagrama de clases para la clase NormOpsPlugin. Mostrando las clases implementadas para su funcionalidad.

Fuente: Elaboración Propia.

Esta clase tiene dos características principales:

- Extiende la clase **Plugin** e implementa la clase SearchPlugin de Elasticsearch, que es la forma como el mecanismo de extensiones de Elasticsearch permite que se puedan registrar extensiones en el sistema, tal y como se documenta en (Elasticsearch, 2023a).
- Sobreescribe el método **getQueries**, que se utiliza para registrar los **QueryBuilders** con Elasticsearch de tal forma que se puedan utilizar. En este caso, registra

```
@Override
public List<QuerySpec<?>> getQueries() {
 return asList(
   new QuerySpec<>(NormTermQueryBuilder.NAME, NormTermQueryBuilder::new,
                   NormTermQueryBuilder::fromXContent)
   new QuerySpec<>(NormOrQueryBuilder.NAME, NormOrQueryBuilder::new, NormOrQueryBuilder::fromXContent)
         ):
```

Ilustración 51 Implementación del método getQueries() utilizado para registrar los operadores desarrollados en Elasticsearch

Fuente: Elaboración Propia.

NormTermQueryBuilder y **NormOrQueryBuilder**

Si bien esta clase no interviene en la lógica de los cálculos de las puntuaciones, es necesaria para poder utilizar los operadores en Elasticsearch.

6.4 Evaluación de la solución

Para la evaluación de la solución se utilizó la metodología descrita en la sección 4.7. En cuanto a la implementación de esta, se crearon programas en el lenguaje de programación Python para procesar los archivos de las colecciones de prueba a utilizar, indexarlos, y posteriormente ejecutar las pruebas y evaluar los resultados. Estas colecciones de prueba se encuentran disponibles en el repositorio de las colecciones de prueba de la Universidad de Glasgow (Glasgow Information Retrieval Group, 2023).

El código de la implementación de la evaluación se encuentra disponible en el archivo comprimido entregado junto al presente documento y en la plataforma Github[1]. La estructura de los directorios es la siguiente:

[1] Disponible en https://github.com/joaguilar/dl_and_index

- Data: directorio que contiene los archivos de las colecciones. Estos archivos deben bajarse de forma independiente y colocarse en los directorios correspondientes:
    - Cran: los archivos de la colección de Cranfield
    - Medline: los archivos de la colección de Medline
    - Time: los archivos de la colección de time
- indexers: El directorio indexers contiene las clases de Python que toman los archivos de las colecciones, los transforma, y los indexa en el índice de Elasticsearch. Para esto utiliza una plantilla de índice y una plantilla de documento de Elasticsearch que se encuentran en el directorio index_profiles. Dentro de este directorio existe una clase llamada Baseindexer, la cual es la que ejecuta todo el proceso de indexación. Adicionalmente existen 3 subclases, una para cada colección (CranIndexer, MedlineIndexer, TimeIndexer) que heredan de la clase BaseIndexer y la configuran de forma correcta para poder indexar los documentos específicos de cada colección.
- Index_profiles: Este directorio contiene las plantillas de Elasticsearch de los índices a utilizar para cada colección, así como plantilla de documentos que utilizan las clases del directorio indexers para crear los documentos que envían a Elasticsearch para su indexado.
- Processors: Finalmente el directorio processors contiene dos clases para cada colección, cuyo fin es realizar el análisis sintactico de los archivos de cada colección y extraer la información necesaria. En este directorio se encuentra la clase BaseProcessor que es la clase principal de la cual heredan los demás procesadores. Para cada colección existe una clase llamada “<nombre de colección>Processor”, como por ejemplo TimeProcessor, que extrae los documentos a indexar de la colección respectiva (Time en este caso). Además, existe una clase llamada “<nombre de la colección>QueryProcessor” como por ejemplo TimeQueryProcessor, que realiza el análisis tanto de las consultas como de los archivos de juicio de cada colección.

Aparte de estos subdirectorios, existen varios programas que utilizan estas clases descritas anteriormente para realizar el indexado de las colecciones y la ejecución de las evaluaciones:

- dl_and_index.py: Programa principal que toma los archivos obtenidos de cada colección, y utilizando los processors los indexa en Elasticsearch
- get_search_metrics.py: Programa de análisis de datos. Ejecuta el flujo de tomar las consultas, ejecutarlas en el índice de Elasticsearch creado por el programa dl_and_index.py, y comprar los resultados obtenidos contra los resultados esperados de acuerdo a los archivos de juicios de cada colección. Basado en estos resultados este programa calcula las métricas usadas en la evaluación, a saber la precisión, la cobertura, el F1, y el nDCG, tal y como se explica en la sección Técnicas de Análisis de Información. Este programa produce una serie de archivos con valores separados por comas (*.csv) con los resultados.
- analyze_results.py: Programa que toma la salida del programa get_search_metrics.py, los archivos *.csv, y los procesa para producir gráficos que analicen los resultados.

El siguiente diagrama de secuencia muestra la funcionalidad de la carga de los documentos a índice de Elasticsearch:

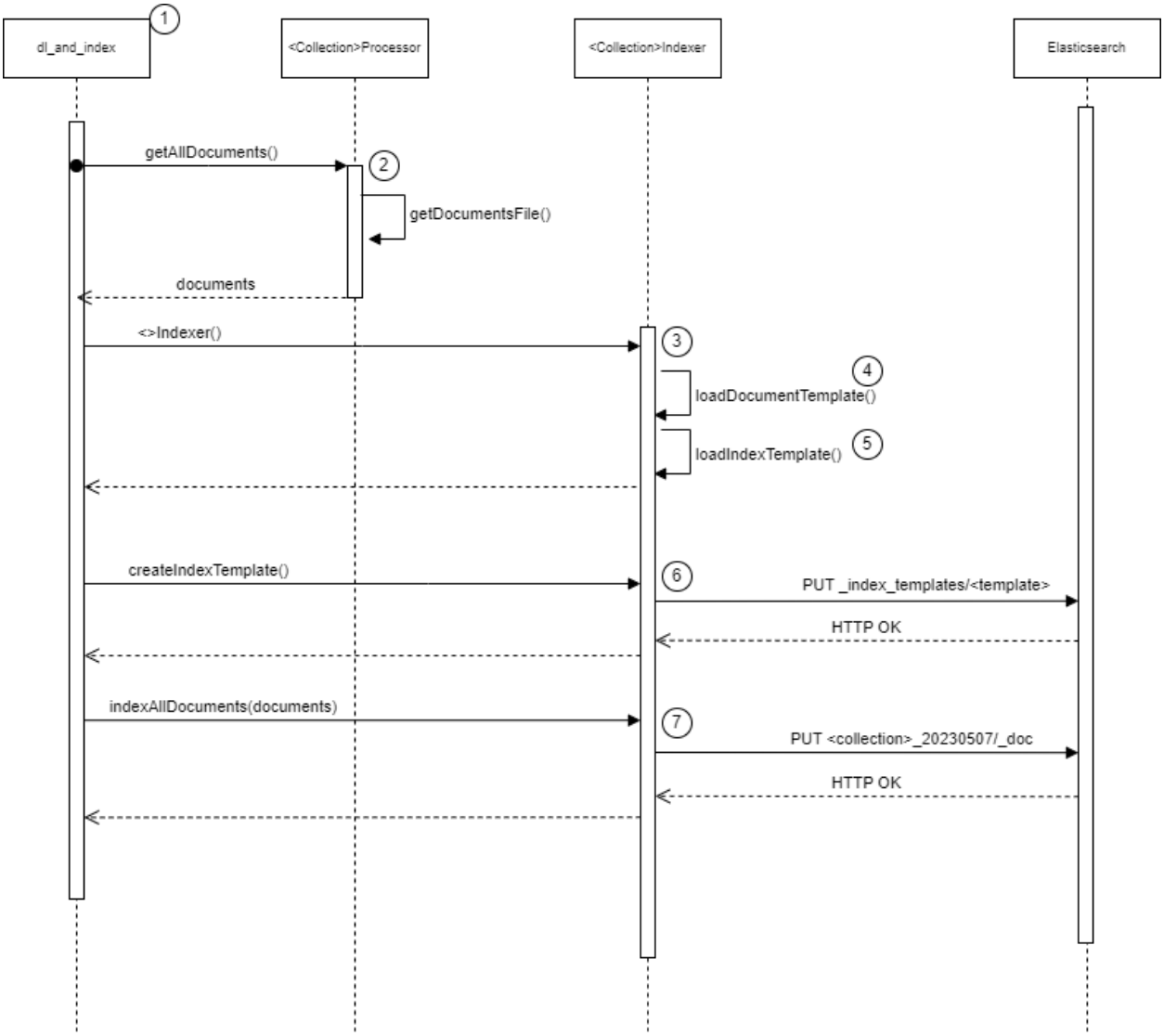


Ilustración 52 Secuencia de pasos para la carga de los documentos de las colecciones a Elasticsearch para su indexado. Fuente: Elaboración Propia

En este diagrama se observan los siguientes pasos:

1. Para iniciar el proceso de carga de los datos, se ejecuta el programa de Python "dl_and_index.py". Este programa ejecuta la secuencia de pasos del 2 al 7 para cada una de las colecciones en consideración.
2. El primer paso es la carga de documentos. Para cada colección existe una clase de Python con el nombre de la colección más un sufijo "Processor" (CranProcessor.py, TimeProcessor.py, y MedlineProcessor.py). Estas clases heredan de la clase

BaseProcessor.py, la cual tiene la lógica de cargar y retornar los documentos de la colección. Las subclases de lo que se encargan es del análisis del formato especifico de los archivos de cada colección para extraer los documentos. Dentro de la clase el método getDocumentFile() carga el o los archivos de documentos, y la clase Processor crea los objetos de documentos para retornarlos al programa principal.

3. Adicionalmente, para cada colección existe una clase de Python con el nombre de la colección más un sufijo "Indexer" (CranIndexer.py, TimeIndexer.py, y MedlineIndexer.py). Estas clases heredan de la clase BaseIndexer.py, la cual tiene la lógica de leer documentos y cargarlos en el índice de Elasticsearch.
4. Cada subclase carga una plantilla de documento de un archivo, el cual es un archivo tipo json donde se tienen marcadores que son reemplazados por los valores reales de cada documento. El método loadDocumentTemplate() carga esta plantilla y la guarda en memoria para su uso posterior.
5. Adicionalmente para cada colección existe una plantilla de índice de Elasticsearch, que es la que indica cómo serán indexados los documentos (Elasticsearch, 2023b). El método loadIndexTemplate() carga esta plantilla de un archivo json y la guarda en memoria.
6. El programa principal ejecuta la función createIndexTemplate() del Indexer de cada colección para crear la plantilla del índice en Elasticsearch.
7. Finalmente, con la plantilla creada, se recorre la lista de documentos cargados de los archivos y se ejecuta uno por uno la carga en Elasticsearch, rellenando los marcadores de la plantilla de documentos.

Una vez cargados los documentos en Elasticsearch, el siguiente paso es realizar la evaluación de las consultas. Para esto es necesario cargar las consultas, los archivos de juicio, ejecutar las consultas, y generar las métricas respectivas que serán usadas para la evaluación. El siguiente diagrama de secuencia muestra este proceso:

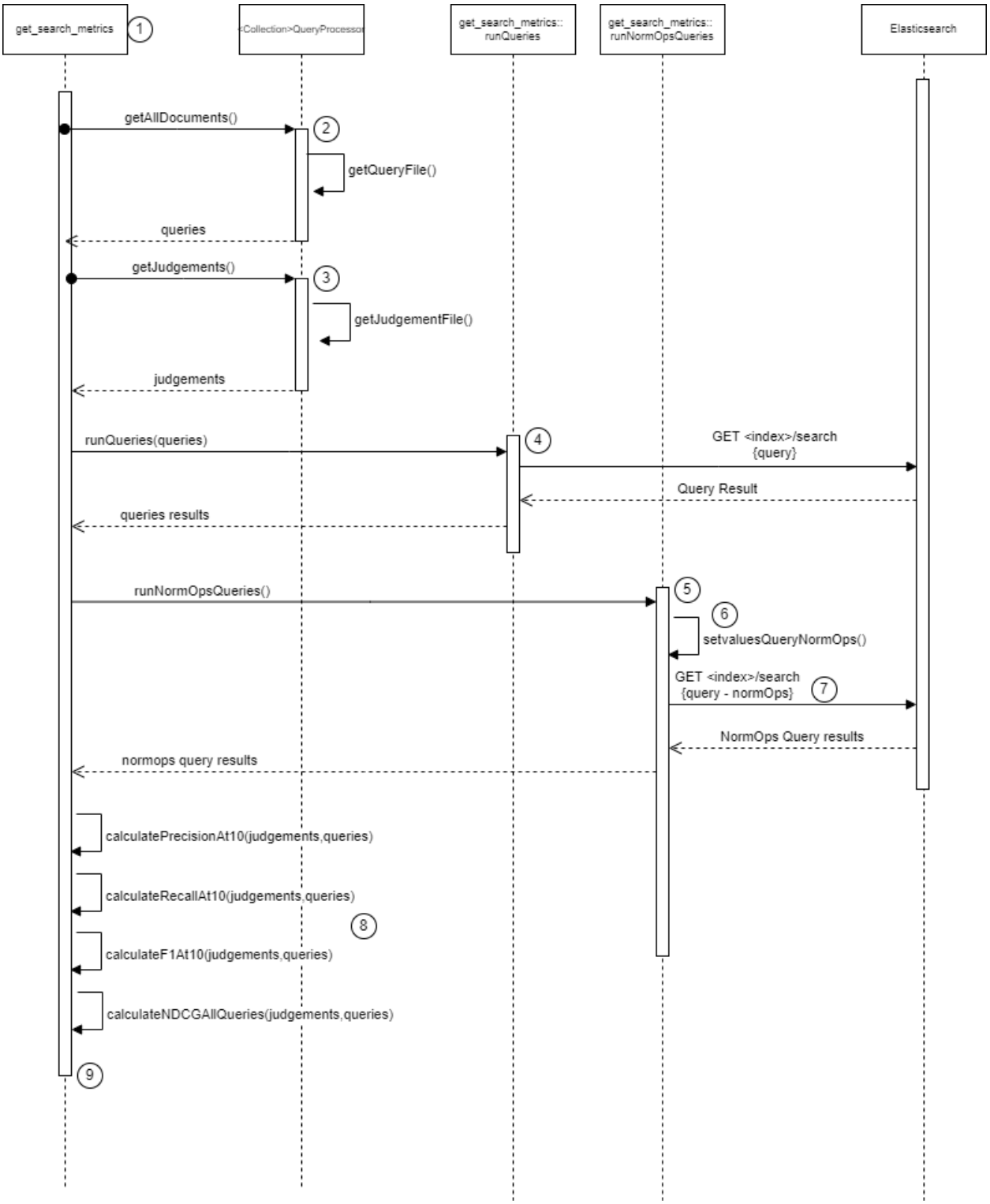


Ilustración 53 Secuencia de pasos para la carga de las consultas y de los archivos de juicios de las colecciones, así como la ejecución de las consultas en Elasticsearch y el cálculo de las métricas de las consultas. Fuente: Elaboración Propia

En este diagrama se observan los siguientes pasos:

1. El programa principal para ejecutar la evaluación de la solución se encuentra en el archivo de Python “get_search_metrics.py”.
2. De forma análoga a como se realiza con la carga de datos, para cada colección existe una clase de Python con el nombre de la colección más un sufijo “QueryProcessor” (CranQueryProcessor.py, TimeQueryProcessor.py, y MedlineQueryProcessor.py). Estas clases heredan de la clase BaseQueryProcessor.py, la cual tiene la lógica de cargar y retornar tanto las consultas como los juicios de la colección. Las subclases de lo que se encargan es del análisis del formato especifico de los archivos de cada colección para extraer esta información.

   El método getAllDocuments() de estas clases se encargan realizar el análisis de los archivos con formato especifico de cada colección, carga las consultas y las retorna al programa principal en un formato que se estandarizo a través de todas las colecciones.
3. El método getJudgements() de estas clases se encargan realizar el análisis de los archivos con formato especifico de cada colección, carga los juicios y las retorna al programa principal en un formato que se estandarizo a través de todas las colecciones.
4. Una vez cargadas las consultas, el programa principal llama a la función runQueries(). Esta función toma las consultas cargadas anteriormente, y crea consultas en el formato requerido de Elasticsearch utilizando el operador *match* que se especificó en la sección 5.2 Match Query. Posteriormente ejecuta cada consulta contra el índice de Elasticsearch y almacena los resultados para su uso posterior.
5. Seguidamente el programa principal llama a la función runNormOpsQueries ().
6. Dentro de la función runNormOpsQueries () se ejecuta otra función llamada setvaluesQueryNormOps(). Esta función toma las mismas consultas usadas en el punto anterior, y crea consultas en el formato requerido de Elasticsearch utilizando los

operadores normalizados desarrollados como parte del proyecto, tal y como se explican en la sección 6.2 Alcance de la implementación.

7. Una vez creadas estas consultas, se ejecuta cada una contra el índice de Elasticsearch y almacena los resultados para su uso posterior.
8. Concluida la ejecución, para cada conjunto de consultas, se calculan varias métricas de rendimiento, específicamente la precisión, la cobertura, el F1, y NDCG, llamando a las funciones respectivas mostradas en el diagrama.
9. Finalmente, los resultados se guardan en archivos CSV para su posterior análisis.

6.5 Resultados de la evaluación

Para poder realizar esta evaluación de resultados, se realizó el proceso descrito en la sección 4.6.2, el cual se hizo en varias iteraciones, ya que se buscó el cambiar parámetros en los operadores sobre todo en el operador propuesto en la investigación al cual para efecto de analizar los resultados nombramos “NormOps” y seguiremos refiriéndonos el mismo para dicha evaluación con ese nombre. Además de esto también se realizó dicho proceso con las colecciones propuestas en la sección 4.5, ya que se busca evaluar el comportamiento de los operadores con diferentes colecciones de datos, por otro lado también se pretende con esto el poder evaluar la normalización de los “score” de los resultados de las búsquedas de prueba realizadas con las consultas de prueba de cada una de las colecciones y poder determinar las métricas de evaluación propuestas en la sección 4.3.3 en base al archivo de los resultados previstos según la relevancia para cada consulta estipulada en cada una de las colecciones. Para cada una de las pruebas de los operadores en las colecciones se estará generando una bitácora en formato CSV en el cual se guardará la información como el conjunto de datos que se está utilizando, el operador que se utilizó, así como la parametrización de los operadores y los resultados de estos en los criterios ya mencionados.

6.5.1 Parametrización del operador NormOps

Para poder realizar las pruebas y posterior evaluación se buscó obtener los mejores resultados posibles del operador propuesto en la investigación esto con el fin de realizar la comparación con los mejores números posibles. Ahora bien, para esto se realizaron ciertos cambios en ciertos parámetros, los cuales pretendían buscar mejorar las métricas del operador, sobre todo la métrica del NDCG, la cual es una de las más relevantes, siendo esta la que da un valor numérico y que también realiza el posicionamiento de los documentos del operador. Esta parametrización se hizo con varias iteraciones y combinaciones de parámetros siendo los parámetros de "docData_clauses", "docDataNorm_clauses" y "weigth_clauses" los que más se cambiaron buscando la combinación que diera los resultados óptimos. Este cambio de parámetro se realizó para cada una de las colecciones y con diferentes combinatorias las cuales se pueden apreciar en las siguientes tablas:

Tabla 32 Pruebas de los parámetros utilizados en la colección Cran.

Colección	Parametrización	Promedio del NDCG
Cran	docData_clauses = "BM25TF" docDataNorm_clauses = "HALF-SIGMOIDE" weigth_clauses= "BM25IDF"	0.0054
Cran	docData_clauses = "BM25TF" docDataNorm_clauses = "SIGMOIDE" weigth_clauses= "BM25IDF"	0.0052
Cran	docData_clauses = "BM25TF" docDataNorm_clauses = "HALF-SIGMOIDE" weigth_clauses= "IDF"	0.0056
Cran	docData_clauses = "TF"	0.0054

	docDataNorm_clauses = “HALF-SIGMOIDE” weigth_clauses= “BM25IDF”	
Cran	docData_clauses = “BM25TF” docDataNorm_clauses = “SIGMOIDE” weigth_clauses= “IDF”	0.0053
Cran	docData_clauses = “TF” docDataNorm_clauses = “SIGMOIDE” weigth_clauses= “IDF”	0.0053

Fuente: Elaboracion propia.

Tabla 33 Pruebas de los parámetros utilizados en la colección Medline

Colección	Parametrización	Promedio del NDCG
Medline	docData_clauses = “BM25TF” docDataNorm_clauses = “HALF-SIGMOIDE” weigth_clauses= “BM25IDF”	0.145
Medline	docData_clauses = “BM25TF” docDataNorm_clauses = “SIGMOIDE” weigth_clauses= “BM25IDF”	0.157
Medline	docData_clauses = “BM25TF” docDataNorm_clauses = “HALF-SIGMOIDE” weigth_clauses= “IDF”	0.144
Medline	docData_clauses = “TF” docDataNorm_clauses = “HALF-SIGMOIDE” weigth_clauses= “BM25IDF”	0.145

Medline	docData_clauses = “BM25TF” docDataNorm_clauses = “SIGMOIDE” weigth_clauses= “IDF”	0.156
Medline	docData_clauses = “TF” docDataNorm_clauses = “SIGMOIDE” weigth_clauses= “IDF”	0.156

Fuente: Elaboracion propia.

Tabla 34 Pruebas de los parámetros utilizados en la colección Time

Colección	Parametrización	Promedio del NDCG
Time	docData_clauses = “BM25TF” docDataNorm_clauses = “HALF-SIGMOIDE” weigth_clauses= “BM25IDF”	0.0076
Time	docData_clauses = “BM25TF” docDataNorm_clauses = “SIGMOIDE” weigth_clauses= “BM25IDF”	0.0097
Time	docData_clauses = “BM25TF” docDataNorm_clauses = “HALF-SIGMOIDE” weigth_clauses= “IDF”	0.0074
Time	docData_clauses = “TF” docDataNorm_clauses = “HALF-SIGMOIDE” weigth_clauses= “BM25IDF”	0.0076
Time	docData_clauses = “BM25TF” docDataNorm_clauses = “SIGMOIDE” weigth_clauses= “IDF”	0.0100

Time	docData_clauses = “TF” docDataNorm_clauses = “SIGMOIDE” weigth_clauses= “IDF”	0.0100

Fuente: Elaboracion propia.

Ahora, según como se puede apreciar en las figuras Figura , Figura , Figura las cuales resumen los resultados obtenidos en cada operador utilizado, siendo el que trae el motor de ElasticSearch por defecto y el operador propuesto NormOps en esta investigación, además de las diferentes colecciones de datos utilizadas. En dichas figuras, se pueden comparar los 4 aspectos numéricos (Precisión, Cobertura, F1 y NDCG) propuestos en la sección 4.3.3 para evaluar el desempeño de los operadores utilizados.

6.5.2 Evaluación colección Medline

En el caso de la colección de Medline, se puede apreciar como el operador de ElasticSearch muestra un mejor desempeño en los 4 aspectos en comparación al operador NormOps, esto, aunque se realizó la prueba con distintos parámetros en NormOps y distintas veces, aun así los resultados no variaron y el operador de ElasticSearch tuvo con esta colección mejores resultados.

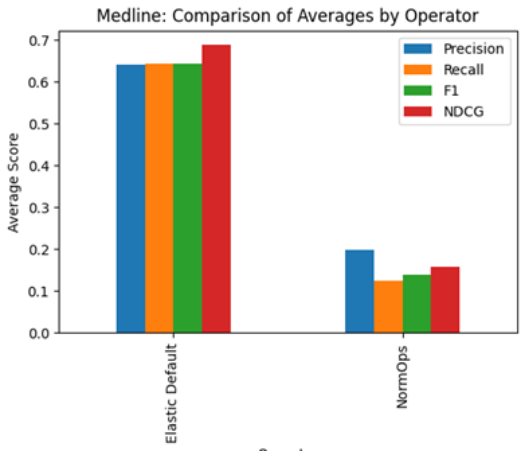


Figura 9 Resultados de las pruebas realizadas a la colección Medline

Fuente: Elaboracion propia.

6.5.3 Evaluación colección Cran

Por otro lado, al evaluar los operadores con la colección CRAN se tienen mejores resultados del operador propuesto, destacando que el operador NormOps es altamente superior en la métrica del NDCG, la cual es de suma importancia ya que es la que permite determinar qué tan bien ordena los documentos relevantes, es decir, que el operador propuesto ordena de una manera más eficiente los documentos relevantes, dándole las posiciones más altas a los documentos con mayor relevancia. También es importante destacar que en métricas como la Cobertura se mantienen muy similares al operador por defecto de ElasticSearch, sin embargo, se puede ver como en métricas como la Precisión y el F1 se quedan un poco por detrás, a pesar de esto el rendimiento del operador NormOps es destacable ya que incluso logra superar al operador por defecto del motor de búsqueda en una de las métricas de mayor importancia para la evaluación.

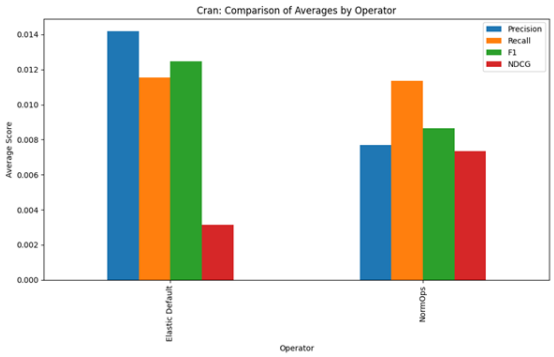


Figura 10 Resultados de las pruebas realizadas a la colección Cran

Fuente: Elaboracion propia.

6.5.4 Evaluación colección Time

Por último, se tienen los resultados de los operadores en la colección Time, en los cuales, como se puede ver en la Figura los números de las métricas con las cuales se evalúa el rendimiento y el desempeño de los operadores son mejores para el operador NormOps, el cual destaca en métricas como la Precisión, el F1 y el NDCG, siendo estas unas de las métricas que superan en

gran medida al operador de ElasticSearch, además se destaca por tener un NDCG mayor al del operador de ElasticSearch y como bien se ha visto esta es una de las métricas de mayor peso a la hora de evaluar el rendimiento de un operador, al igual que con las demás colecciones, para dicha colección también se realizaron pruebas con diferentes parámetros en los operadores buscando el mejor resultado.

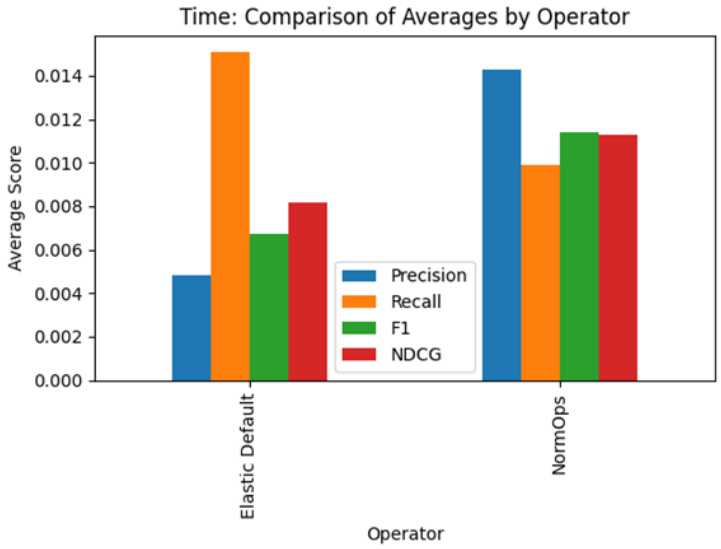


Figura 11 Resultados de las pruebas realizadas a la colección Time

Fuente: Elaboración propia

6.5.5 Evaluación de la normalización de los Score

Otro de los aspectos que cabe recalcar en los resultados es la normalización de los score de los resultados del operador NormOps, tal como se puede apreciar en las figuras más adelante, las cuales muestran los resultados promedio de los "score" de los operadores tanto de ElasticSearch como del NormOps, en los cuales se puede evidenciar que el operador propuesto NormOps mantiene siempre un "score" con un valor entre 0 y 1, independientemente de la colección de documentos en la que se hay utilizado, gracias a estos números se puede corroborar que la intención de normalizar los resultados y que estos no varíen tanto, de una colección a otra, como pasa actualmente con el operador de ElasticSearch, que como bien se puede apreciar en las

siguientes figuras, estas muestran “score” de los resultados cambiantes según la colección que se utilice.

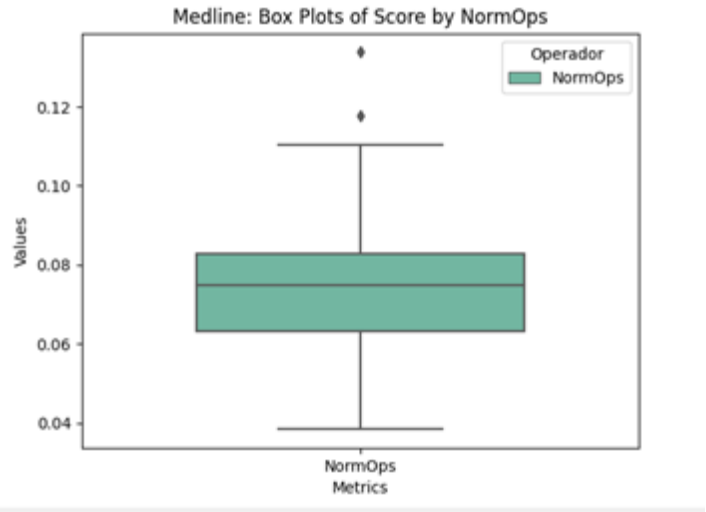


Figura 12 Score del Operador NormOps en la colección Medline

Fuente: Elaboración propia

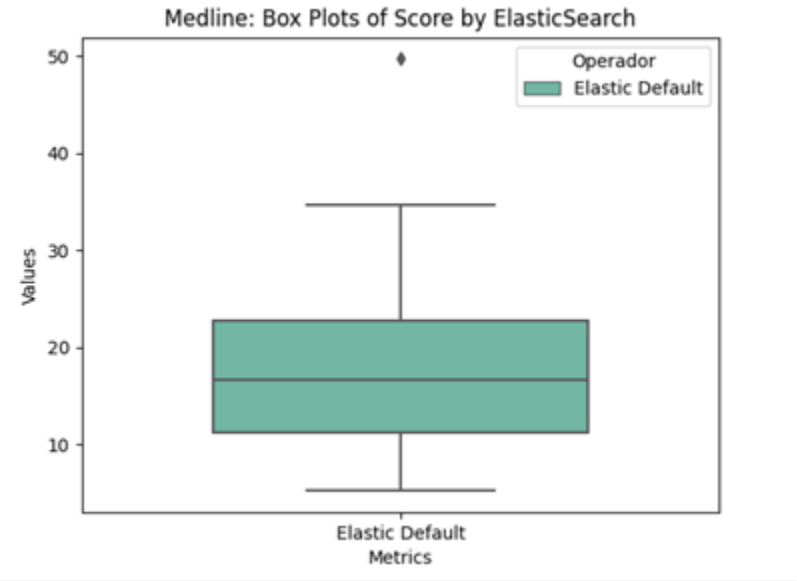


Figura 13 Score del Operador de ElasticSearch en la colección Medline

Fuente: Elaboración propia

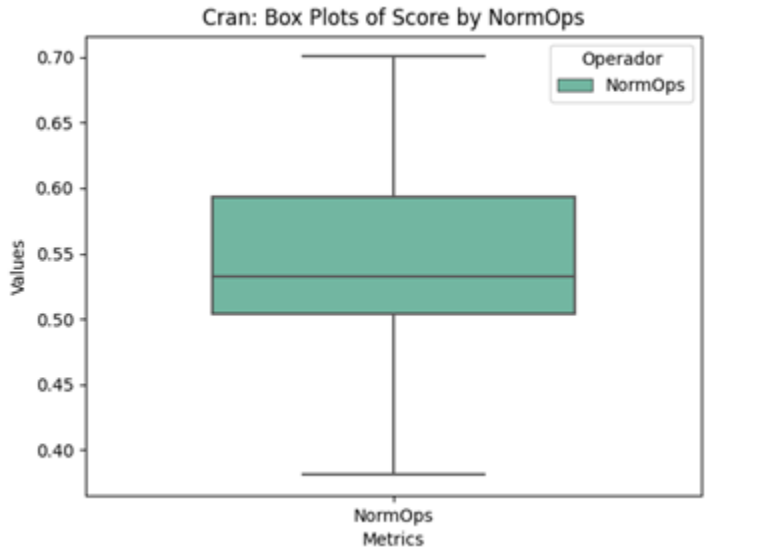


Figura 14 Score del Operador NormOps en la colección Cran

Fuente: Elaboración propia

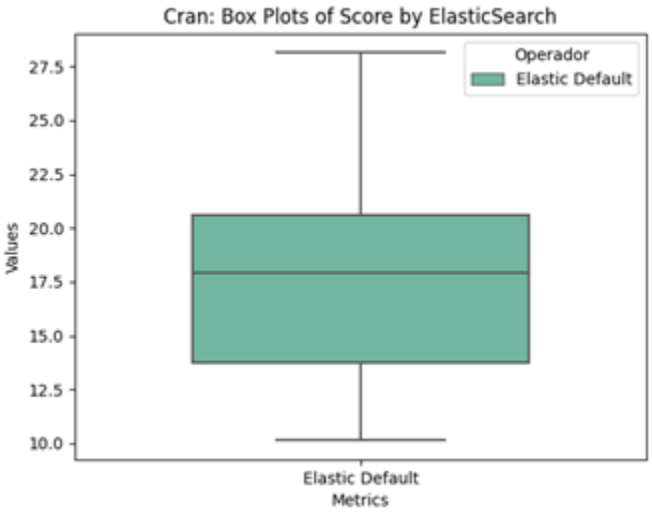


Figura 15 Score del Operador de Elasticsearch en la colección Cran

Fuente: Elaboración propia

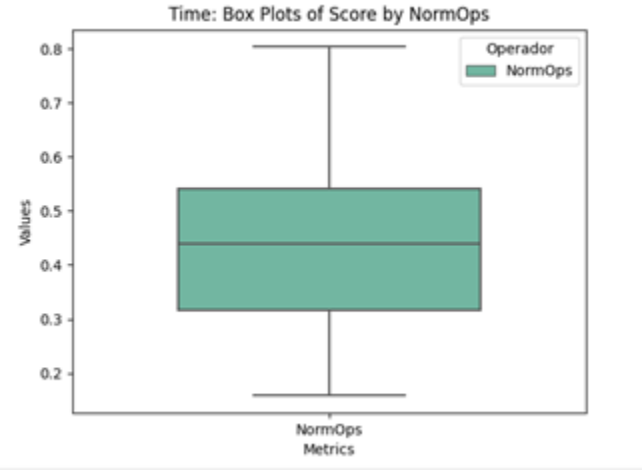


Figura 16 Score del Operador NormOps en la colección Time

Fuente: Elaboración propia

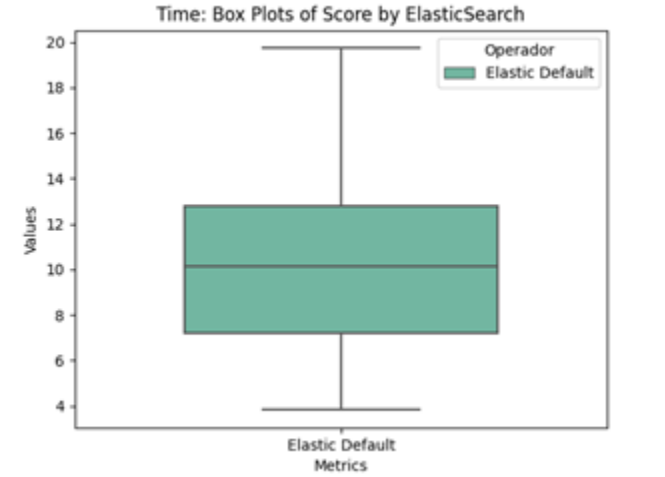


Figura 17 Score del Operador de Elasticsearch en la colección Time

Fuente: Elaboración propia

6.5.6 Evaluación final

Finalmente, basado en los criterios de evaluación que se establecieron en la sección 4.3.3, actualmente los resultados se consideran aceptables, ya que si bien no hubo resultados sobresalientes, el operador NormOps logró destacar en métricas de la evaluación importantes, donde tiene resultados comparables e incluso mejores en algunos aspectos, en dos de las 3

colecciones, sobre todo en métricas relevantes como la Precisión y el NDCG. Además de superar en una de las colecciones al operador de ElasticSearch en 3 métricas, lo cual es algo destacable. También hacer mención de que dicho operador logró el cometido de crear una normalización en los resultados del "score" de los documentos.

Tabla 35 Resultados de la Evaluación de los Operadores

Rubro	Operador NormOps	Operador ElasticSearch	Valor	Comentarios
	Evaluación de métricas de recuperación de información			
Precisión	10%	15%	20%	Precision@10
Cobertura	10%	5%	10%	Complicada de evaluar al existir la posibilidad que no se conozca la totalidad de los documentos relevantes
F1	10%	5%	10%	Al combinar la precisión y cobertura se considera que debe tener una importancia menor por las dificultades de la cobertura mencionadas
nDCG	15%	15%	20%	Los documentos relevantes deben aparecer en posiciones altas en los resultados, de lo contrario se penaliza conforme aparezcan en posiciones más bajas.
	Evaluación de factores de uso de los operadores			
Flexibilidad	20%	15%	20%	Se pretende analizar la capacidad del operador al realizar consultas con diferentes conjuntos de datos, así como analizar cómo se comportan con varios tipos de datos.

Facilidad de uso	7%	9%	10%	Da un vistazo de la dificultad de poder implementar un operador u otro en la máquina de búsqueda por parte del desarrollador
Explicación de resultados	5%	20%	20%	Permite conocer que tan fácil le es al usuario determinar por qué el operador considero un resultado más relevante que otro.
Total	77%	84%	100%	

Fuente: Elaboración propia

La evaluación de los factores de uso de los operadores, fueron puntuadas de acuerdo con el criterio experto de los investigadores, y se consideran los siguientes factores:

Flexibilidad: Si bien tanto los operadores de Elasticsearch como los operadores de *NormOps* permiten una amplia configuración de parámetros, se puede mencionar que los *NormOps,* al permitir cambiar de forma significativa como se calculan los puntajes finales, se considera que son más flexibles en cuanto a la relevancia de los documentos. Sin embargo, los operadores de Elasticsearch son más fáciles de usar, sin complicaciones sobre las fórmulas matemáticas de relevancia, y permiten una configuración important.

Facilidad de Uso: Ambos tipos de operadores requieren un conocimiento exhaustivo de las diferentes opciones requeridas. Sin embargo, los operadores de Elasticsearch se pueden configurar más fácilmente y es posible combinarlos con cualquier otro operador de Elasticsearch. Los de *NormOps* solamente pueden funcionar dentro de la misma familia de operadores.

Explicación de los resultados: La facilidad que tienen los operadores de Elasticsearch para mostrar cómo se llegó al resultado es superior que la de los operadores *NormOps*. Sin embargo, los operadores *NormOps,* al dar un puntaje entre 0 y 1 a cada resultado, le puede facilitar a los

usuarios determinar qué tan relevante es el resultado con respecto a su consulta basado en ese puntaje.

7 Conclusiones y Recomendaciones

7.1 Conclusiones

7.1.1 Conclusiones del Objetivo 1

"Describir los diferentes métodos de relevancia en máquinas de búsqueda de alto uso para obtener una perspectiva de los métodos más utilizados."

Este objetivo fue alcanzado y se concluye:

- Las máquinas de búsqueda más utilizadas actualmente están basadas en la biblioteca Apache Lucene.
- Los métodos más utilizados por Apache Lucene fueron descritos en la sección 3 Marco Conceptual
- En esta sección se mencionan y describen los métodos TF/IDF y BM25 que son los más utilizados por Apache Lucene, y por su implementación en Elasticsearch.

7.1.2 Conclusiones del Objetivo 2

"Enumerar los métodos de evaluación de relevancia de máquinas de búsqueda con el fin de escoger un método de evaluación a utilizar para comparar los diferentes enfoques."

Este objetivo fue alcanzado y se concluye:

- Existen múltiples métodos de evaluación en recuperación de información, entre los cuales se pueden mencionar DCG, nDCG, precisión, cobertura y F1, que fueron enumerados en la sección 4.3.3 Dimensión Axiológica
- Los más utilizados son el *Normalized Discounted Cumulative Gain*, Precisión, Cobertura y F1, los cuales fueron utilizados en el presente estudio para la evaluación cuantitativa de los resultados presentados en la sección 6.5 Resultados de la evaluación.

7.1.3 Conclusiones del Objetivo 3

"Comprender los operadores de relevancia expuestos en (P. E. Nelson & David, 2021) para que generen puntajes normalizados para la máquina de búsqueda de alto uso."

Este objetivo fue alcanzado y se concluye:

- Se revisó el documento de la patente (P. E. Nelson & David, 2021) y referencias adicionales con el fin de comprender el alcance de la misma y los objetivos de la misma.
- Se concluye que es necesario, aparte de tener un amplio conocimiento del tema de recuperación de información, conocer ampliamente los métodos y fórmulas matemáticas utilizadas para alcanzar la normalización de los diferentes componentes de las fórmulas de relevancia, con el fin de poder implementarlos.
- Se presenta un resumen de la patente en la sección 6.1 Resumen de la solución

7.1.4 Conclusiones del Objetivo 4

"Construir un subconjunto de los operadores de relevancia planteados en la patente expuesta en (P. E. Nelson & David, 2021) para la máquina de búsqueda de alto uso para poder compararlos con los métodos utilizados actualmente."

Este objetivo fue alcanzado y se concluye:

- Con el fin de comparar el rendimiento de los operadores, se implementaron solamente aquellos que fueron necesarios para poder compararlos con una consulta en Elasticsearch, utilizando el tipo de consulta "match", el cual es el utilizado para consultas de texto, y sin mayor configuración.
- Los operadores implementados, fueron el *normOr*, el cual calcula puntajes de relevancia normalizados entre 0 y 1 con base en cláusulas que consisten en el otro operador implementado, el *normTerm*, que calcula el puntaje de relevancia normalizado entre 0 y 1 de un término de la consulta, tal y como se explica en la sección 6 Propuesta de Solución
- Este tipo de implementación se puede considerar como una prueba de concepto de los operadores de la patente. En ellos se pueden evaluar los resultados utilizando lo mínimo

necesario, con el fin de determinar si es viable la implementación de toda la familia de operadores descrita en 6.1 Resumen de la solución.

- La implementación de los operadores en Elasticsearch no es una tarea trivial. Es necesario un amplio conocimiento de las estructuras de datos de Apache Lucene, de cómo interactúan las diferentes clases involucradas en la ejecución de una consulta, y de cómo escribir un *plugin* de Elasticsearch. En esto se utilizó la mayor parte del tiempo de los investigadores y fue un factor que influyó en la decisión de crear los operadores mínimos para poder evaluar la viabilidad de esta implementación.

7.1.5 Conclusiones del Objetivo 5

"Comparar los nuevos operadores en contraste con los incluidos en la máquina de búsqueda de alto uso para determinar su efectividad o sus beneficios."

Este objetivo fue alcanzado y se concluye:

- Se realizaron las pruebas calculando las métricas predefinidas, utilizando el tipo de consulta *match* de Elasticsearch y los operadores *normOr* y *normTerm* implementados por los investigadores.
- Con el fin de validar los operadores en diferentes colecciones, se utilizaron 3 colecciones que, si bien tienen documentos relativamente cortos todas ellas, tanto los documentos como las consultas son diferentes. Esto permitió ver cómo se comportan los operadores en diferentes escenarios.
- Es posible normalizar los resultados de las búsquedas de 0 a 1, lo cual puede facilitar la interoperabilidad y la comparación entre varias consultas. Esto se observa en los resultados de la ejecución de los experimentos.
- De los resultados observados, es posible concluir que, en dos de las colecciones, Cranfield y Time, estos operadores dieron un resultado de relevancia bastante aceptable. En la colección Medline, sin embargo, la diferencia con el operador *match* de Elasticsearch fue significativa, lo cual indica que este tipo de normalización puede estar afectando la relevancia en el tipo de documentos que forman parte de esta colección.

- Si bien se realizó la implementación de las fórmulas de normalización mencionadas en el documento, existen otras fórmulas que se pueden evaluar y que eventualmente pueden dar mejores resultados. Esto queda para un trabajo futuro.
- El marco de trabajo creado para la evaluación de las consultas con los diferentes operadores fue algo realmente valioso. Este marco de trabajo puede ser reutilizado fácilmente para comparaciones de relevancia de forma automática, en escenarios donde se quiere evaluar el impacto de cambios en parámetros o diferentes fórmulas de relevancia. La arquitectura de este permite que se agreguen nuevas colecciones o nuevos tipos de consulta muy fácilmente.

7.1.6 Conclusión Objetivo General

"Evaluar un nuevo enfoque de operadores de relevancia normalizados en máquinas de búsqueda por medio de su implementación y comparación con los operadores existentes para determinar su efectividad."

Es posible concluir que se pudo realizar la evaluación del enfoque de los operadores de relevancia normalizados por medio de la implementación de dos de ellos, *normOr*, y *normTerm*, lo cual permitió compararlos con el operador de Elasticsearch que tiene la funcionalidad de poder realizar búsquedas de texto.

Sin embargo, es necesario destacar que esta implementación y comparación de estos operadores lo que permitió fue evaluar el enfoque y el impacto que estos tienen en la relevancia de los documentos. No es posible concluir completamente sobre la superioridad de un enfoque sobre otro. Sin embargo, los resultados fueron lo suficientemente cercanos en dos de las tres colecciones usadas para la evaluación, por lo que se considera que es viable continuar la implementación de los demás operadores y pueden convertirse en una alternativa a la forma tradicional en que se calcula la relevancia dentro de Lucene.

Vale la pena destacar que un gran aporte de la investigación fue la comparación de los operadores como tal. El hecho de tener un marco de trabajo que permite rápidamente ejecutar este tipo de comparaciones no solo facilitó la evaluación, sino que es posible utilizarlo fuera de

este contexto con modificaciones mínimas, para comparar diferentes formas de calcular la relevancia de documentos y obtener una medición objetiva de la misma.

En esta evaluación del enfoque solamente se consideraron 4 formas de normalizar los puntajes dentro de los operadores. Es posible que al utilizar otros modelos de normalización se obtengan mejores resultados.

7.2 Recomendaciones

A continuación, se mencionan las recomendaciones basadas en la experiencia de la presente investigación, considerando tanto cuestiones técnicas como procedimentales:

- Los operadores, tanto el operador *match* de Elasticsearch como los operadores implementados *normTerm* y *normOr*, reciben diversos parámetros que permiten ajustar su funcionamiento de diversas formas. Es posible que se puedan mejorar los puntajes de las métricas utilizadas al variar estos parámetros de diferentes formas. Este tipo de evaluación puede quedar para un trabajo posterior.
- En el punto de vista técnico y de evaluación, es posible mencionar que solamente se implementaron algunos modelos de normalización. Para ciertos tipos de documentos y búsquedas es posible que se puedan mejorar las métricas de los operadores implementados utilizando alguno de los modelos adicionales.
- Dada la complejidad de la implementación realizada, es recomendable contar con expertos en las áreas de matemática y con amplio conocimiento de la estructura, funcionamiento e implementación de Apache Lucene.
- La implementación se realizó utilizando una mezcla de Python y Java como lenguajes de programación, y se utilizó GitHub para el control de versiones. Esta es una recomendación general para el desarrollo de software, pero en este caso fue invaluable para poder colaborar en cuanto a cambios en el código y para hacer revisiones de pares del código escrito.
- Es recomendable utilizar una sola herramienta de colaboración para la generación de los documentos compartidos, como en el caso de este documento.

8 Reflexiones Finales

El presente trabajo surgió de la curiosidad de saber el impacto de un enfoque que varía en cierto grado las fórmulas de cálculo de puntajes de relevancia más tradicionales, que se utilizan desde hace mucho tiempo en el campo de la recuperación de información. El ejercicio presentado en esta investigación, si bien no alcanzó para implementar y evaluar todos los aspectos presentados en la patente mencionada en el documento, sí dio una oportunidad de comparar diferentes métodos, e inclusive crear una metodología para hacerlo. Asimismo, se considera que, sí se pudo evaluar el enfoque como tal, de crear puntajes realmente normalizados, que van más allá de la normalización realizada por algoritmos como BM25, y se observó cómo se comportan en colecciones de evaluación conocidas.

Una enseñanza importante fue la profundización necesaria en cuanto a conceptos de recuperación de información como campo, y el aprendizaje de cómo están implementados los algoritmos más comunes ya en forma práctica, en la biblioteca de recuperación de información de código abierto Apache Lucene.

Finalmente, la creación de un marco de trabajo para evaluar dos fórmulas de relevancia es un aporte importante que ya está siendo utilizado en otros aspectos, y está disponible de forma abierta en línea.

9 Trabajos a Futuro

Se consideran los siguientes trabajos a futuro para continuar con la investigación presentada en el presente documento:

- Para realizar una evaluación más completa del enfoque, se recomienda completar la implementación de todos los operadores y los modelos de normalización presentados en la patente en cuestión. Esto permitiría utilizar diferentes operadores en diferentes circunstancias y poder evaluar de forma más profunda el impacto que estos puedan tener.
- Utilizar colecciones de evaluación con un volumen de documentos y consultas que sean de un orden de magnitud más grande.
- Realizar una refactorización de los operadores implementados con el fin de optimizar su rendimiento y poder eventualmente liberarlos como código fuente abierto para que sean aprovechados por la comunidad de recuperación de información.

10 Bibliografía

Apache Software Foundation. (2022a). *Lucene—Core—ASF JIRA*. https://issues.apache.org/jira/projects/LUCENE/summary/

Apache Software Foundation. (2022b). *Org.apache.lucene.search.similarities (Lucene 9.2.0 core API)*. https://lucene.apache.org/core/9_2_0/core/org/apache/lucene/search/similarities/package-summary.html#package.description

Apache Software Foundation. (2022c). *Overview (Lucene 9.2.0 core API)*. https://lucene.apache.org/core/9_2_0/core/index.html

Apache Software Foundation. (2022d). *TFIDFSimilarity (Lucene 9.2.0 core API)*. https://lucene.apache.org/core/9_2_0/core/org/apache/lucene/search/similarities/TFIDFSimilarity.html

Beel, J., Gipp, B., Langer, S., & Breitinger, C. (2016). Research-paper recommender systems: A literature survey. *International Journal on Digital Libraries*, *17*(4), 305–338. https://doi.org/10.1007/s00799-015-0156-0

David D. Lewis & Steve Finch. (1997). *UCI Machine Learning Repository: Reuters-21578 Text Categorization Collection Data Set*. https://archive.ics.uci.edu/ml/datasets/reuters-21578+text+categorization+collection

Ekstrand, M. D., McDonald, G., Raj, A., & Johnson, I. (2022). Overview of the TREC 2021 Fair Ranking Track. *The Thirtieth Text REtrieval Conference (TREC 2021) Proceedings*, *NIST Special Publication 500-335*. https://fair-trec.github.io/index

Emmott, S., Mullen, A., Pidsley, D., & Nelms, T. (2022). *Magic Quadrant for Insight Engines* (Núm. G00748852). Gartner, Inc. Solicitado por medio de https://www.elastic.co/explore/improving-digital-customer-experiences/gartner-magic-quadrant-for-insight-engines-report

Harman, D. (2019). Information Retrieval: The Early Years. *Foundations and Trends® in Information Retrieval*, *13*(5), 425–577. https://doi.org/10.1561/1500000065

Ingersoll, G. S., Morton, T. S., & Farris, A. L. (2013). *Taming Text*. Manning Publications Co. https://www.manning.com/books/taming-text

Lewis, D. D., & Ringuette, M. (1994). A Comparison of Two Learning Algorithms for Text Categorization. *In Third Annual Symposium on Document Analysis and Information Retrieval*, 81–93.

Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press. https://nlp.stanford.edu/IR-book/information-retrieval-book.html

Naranjo-Zeledón, L. (2020). Investigación en Informática: El enfoque alternativo. *Technology Inside by CPIC*, *5*(2), 1–15.

Nelson, P. E., & David, M. (2021). *United States Patent: 11204920 - Utilizing search engine relevancy ranking models to generate normalized and comparable search engine scores* (Patent Núm. 11204920). https://image-ppubs.uspto.gov/dirsearch-public/print/downloadPdf/11204920

Post, W. (2017). *TREC Washington Post Corpus*. TREC Washington Post Corpus. https://trec.nist.gov/data/wapost/

ROBERTSON, S. E. (1977). THE PROBABILITY RANKING PRINCIPLE IN IR. *Journal of Documentation*, *33*(4), 294–304. https://doi.org/10.1108/eb026647

Robertson, S., Walker, S., Jones, S., Hancock-Beaulieu, M. M., & Gatford, M. (1995). *Okapi at TREC-3*. 109–126. https://www.microsoft.com/en-us/research/publication/okapi-at-trec-3/

Rowe, B. R., Wood, D. W., Link, A. N., & Simoni, D. A. (2010). *Economic Impact Assessment of NIST's Text REtrieval Conference (TREC) Program* (Final Report RTI Project Number 0211875; p. 121). https://trec.nist.gov/pubs/2010.economic.impact.pdf

Russell-Rose, T., & MacFarlane, A. (2020, julio 30). *Towards Explainability in Professional Search*. The 3rd International Workshop on ExplainAble Recommendation and Search (EARS 2020). https://ears2020.github.io/

Russell-Rose, T., & Shokraneh, F. (2020). Designing the Structured Search Experience: Rethinking the Query-Builder Paradigm. *Weave: Journal of Library User Experience*, *3*. https://doi.org/10.3998/weave.12535642.0003.102

Salton, G., Wong, A., & Yang, C. S. (1975). A vector space model for automatic indexing. *Communications of the ACM*, *18*(11), 613–620. https://doi.org/10.1145/361219.361220

Samimi, P., & Ravana, S. D. (2014). Creation of Reliable Relevance Judgments in Information Retrieval Systems Evaluation Experimentation through Crowdsourcing: A Review. *The Scientific World Journal*, *2014*, 135641. https://doi.org/10.1155/2014/135641

Sanderson, M. (1994, enero 1). *The Reuters collection*.

Scells, H., & Zuccon, G. (2018). searchrefiner: A Query Visualisation and Understanding Tool for Systematic Reviews. *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, 1939–1942. https://doi.org/10.1145/3269206.3269215

Singhal, A., & Google, I. (2001). Modern Information Retrieval: A Brief Overview. *IEEE Data Engineering Bulletin*, *24*.

VandenBerg, C., Fisher, D., Brooks, G., & The Lemur Project. (s/f). *RankLib File Format*. Recuperado el 9 de agosto de 2022, de https://sourceforge.net/p/lemur/wiki/RankLib%20File%20Format/

What is Elasticsearch? | *Elasticsearch Guide [8.3]* | *Elastic*. (s/f). [Learn/Docs/Elasticsearch/Reference/8.3]. Recuperado el 2 de agosto de 2022, de https://www.elastic.co/guide/en/elasticsearch/reference/current/elasticsearch-intro.html

Wikipedia Contributors. (2022). Apache Lucene. En *Wikipedia*. https://en.wikipedia.org/w/index.php?title=Apache_Lucene&oldid=1092904744

Wong, W. T. (2019). *Advanced Elasticsearch 7.0* (1st ed.). Packt Publishing. https://www.packtpub.com/product/advanced-elasticsearch-7-0/9781789957754