

UNIVERSIDAD CENTRAL DE VENEZUELA
FACULTAD DE CIENCIAS
POSTGRADO EN CIENCIAS DE LA COMPUTACIÓN



RECUPERACIÓN, EXTRACCIÓN Y CLASIFICACIÓN DE INFORMACIÓN DE SABER UCV

Trabajo de grado de Maestría presentado ante la
ilustre Universidad Central de Venezuela por el
Econ. José Miguel Avendaño Infante para optar al título de
Magister Scientiarum en Ciencias de la Computación

Tutor: Dr. Andres Sanoja

Caracas - Venezuela
Octubre 2023

Resumen:

Se presenta la investigación *Recuperación, Extracción y Clasificación de Información de Saber UCV* donde se ejecutan procesos de clasificación, almacenamiento y recuperación de información sobre las tesis y trabajos de grado que se encuentran publicados en el repositorio institucional Saber UCV.

Se implementa un Sistema que permite clasificar los documentos mencionados, según el área académica donde cursó estudios el autor de la investigación.

Se extraen los resúmenes de los trabajos y con las clasificaciones obtenidas se conforma un corpus sobre el cual se genera un índice invertido. Al corpus se le aplican técnicas de Procesamiento de Lenguaje Natural, de Minería de Datos y con modelos preentrenados de inteligencia artificial se crean *embeddings* de textos. Con toda la información procesada se alimenta una base de datos indexada.

El Sistema adicionalmente cuenta con una aplicación web para hacer procesos de Recuperación de Información donde el usuario puede hacer la exploración del corpus, incluida la búsqueda semántica, indicando el: texto a buscar, rango de fechas, área en la cual se generó la investigación, el nivel académico; posteriormente se recuperan los trabajos de mayor relevancia, enriqueciendo la experiencia con la presentación de los resultados en tablas interactivas, Mapas de Conocimiento y recomendaciones de documentos que puedan ser de interés.

La aplicación se implementa bajo un sistema distribuido con la arquitectura cliente-servidor y se soporta en el uso de contenedores orquestados.

Palabras Clave:: Recuperación de Información, Corpus, Procesamiento del Lenguaje Natural, Relevancia, Inteligencia Artificial, Embeddings, Búsqueda Semántica, Mapas de Conocimiento.

Abstract:

The research *Recovery, Extraction and Classification of Information from Saber UCV* is presented, where processes of classification, storage and retrieval of information on theses and degree works published in the institutional repository Saber UCV are executed.

A system is implemented that allows classifying the mentioned documents, according to the academic area where the author of the research studied.

The abstracts of the works are extracted and with the obtained classifications a corpus is formed on which an inverted index is generated. Natural Language Processing and Data Mining techniques are applied to the corpus, and with pre-trained artificial intelligence models, text *embeddings* are created. With all the processed information an indexed database is fed.

The system also has a web application for Information Retrieval processes where the user can explore the corpus, including the semantic search, indicating the: text to search, date range, area in which the research was generated, academic level; subsequently the most relevant works are retrieved, enriching the experience with the presentation of the results in interactive tables, Knowledge Maps and recommendations of documents that may be of interest.

The application is implemented under a distributed system with client-server architecture and is supported by the use of orchestrated containers.

Keywords: Information Retrieval, Corpus, Natural Language Processing, Relevance, Artificial Intelligence, Embeddings, Semantic Search, Knowledge Maps.

Dedicatoria:

A mi hijo Cassiel y

mi esposa Waleska.

De ustedes es esta investigación.

Agradecimientos:

- A mi madre, obvio, sino no habría ni una sola palabra acá.
- A mi padre Fernando por negarme el Atari e insistir en el Oddyssey 2.
- A mi tía Mercedes Infante y mi tío Teófilo García †, por "lo que fue y será".
- A mi hermano David por su aguante y soporte.
- A mi primo Cesar Alejandro García por todo el soporte.
- Dr. Andres Sanoja primero por aceptar la tutoría y enseñarme qué es la investigación dentro de una comunidad científica.
- Dr. José Mirabal por siempre andar con alguna idea a desarrollar y el tiempo dedicado a escuchar las propuestas y complementar esta Investigación.
- Dra. Concettina Di Vasta por las imponentes sesiones de 2 horas 15 minutos llenas de coherencia y conocimiento.
- Dra. Haydemar Nuñez por la rigurosidad al impartir los conocimientos.
- Dra. Vanessa Leguizamo por tomarse el tiempo de revisar la solicitud de estudio de un oxidado economista y por ser mi Prof.^a.
- Dra. Mairene Colina por la ayuda y sugerencias a lo largo de la investigación.
- Mauricio Sáez Toro del equipo Saber UCV por mantener activo el Sistema Saber UCV y tener el tiempo de haber colaborado con esta investigación.
- A Alexandra Asanovna Elbakyan por ayudar a liberar el conocimiento.

- A todo el personal del Postgrado: sus buenos días, por tener a mano la llave, por ayudar a mantener viva la Academia.
- A toda la comunidad de creadores de software libre y open source, en especial a los #useRs por motivarme a adentrarme al mundo de las ciencias de la computación.

Como todos los hombres de la Biblioteca,
he viajado en mi juventud; he peregrinado
en busca de un libro, acaso del *catálogo de
catálogos*; ahora que mis ojos caso no pueden
descifrar lo que escribo, me preparo a morir a
unas pocas leguas del hexágono en que nací.

— Jorge Luis Borges, *La Bibioloteca de Babel*,
Ficciones

Every important aspect of programming arises
somewhere in the context of sorting or
searching.

— Donald Knuth, *The Art of Computer
Programming*, Volume 3

Índice

1	Introducción:	1
1.1	Estructura:	2
2	El Problema:	4
2.1	Descripción del Problema:	4
2.2	Delimitación del Problema:	5
2.3	Descripción de la Solución:	5
2.4	Justificación e Importancia:	6
2.4.1	Objetivo General:	7
2.4.2	Objetivos Específicos:	7
2.5	Aportes:	8
3	Marco Teórico-Referencial:	10
3.1	Reseña histórica:	11
3.2	Recuperación de Información:	13
3.2.1	Sistemas de Recuperación de Información (SRI) :	15
3.2.2	Ejemplos de IRS:	15
3.2.3	Modelos de Recuperación de Información:	16
3.2.3.1	Recuperación booleana:	16
3.2.3.2	Índices Invertidos:	17
3.2.4	Relevancia:	19

3.2.5	Re Ordenamiento (re-ranking):	19
3.2.5.1	Learning to Rank (LTR):	19
3.2.5.2	BM25:	20
3.2.6	Medidas y Métodos de Evaluación:	20
3.3	Procesamientos a los textos:	22
3.3.1	Procesamiento del Lenguaje Natural (Natural Language Processing- NLP):	23
3.3.1.1	Tokenizador:	23
3.3.1.2	Etiquetado de Partes del Discurso (<i>Part of speech tagging-POS</i>):	24
3.3.1.3	Stemming:	24
3.3.1.4	Lematización:	25
3.3.2	Minería de Texto:	25
3.3.2.1	Term-Document Matrix:	26
3.3.2.2	Coocurrencia de Palabras:	26
3.3.2.2.1	Mapas de Conocimiento:	27
3.3.3	Similitud de documentos:	28
3.4	Sistemas Distribuidos:	29
3.4.1	Contenedores:	29
3.4.2	Orquestador:	30
3.5	Estado del Arte:	30
3.5.1	Embeddings	31
3.5.2	Arquitectura de Redes Neuronales <i>Transformers</i> :	37
3.5.3	Largos Modelos de Lenguaje:	38
3.5.4	Integración:	41
3.5.5	Búsqueda Semántica:	41

4	Capítulo Marco Metodológico:	44
4.1	Metodología de Trabajo Kanban:	44
4.2	Desarrollo Adaptable de Software:	45
4.2.1	Características:	45
4.2.2	Ciclos:	46
4.2.2.1	Especulación:	47
4.2.2.2	Colaboración:	47
4.2.2.3	Aprendizaje:	47
5	Desarrollo de la Solución:	48
5.1	Descripción General de la Solución:	48
5.2	Arquitectura de la Solución:	49
5.2.1	Modelo:	49
5.2.2	Vista:	51
5.2.3	Controlador:	52
5.3	Ciclos de Desarrollo:	52
5.3.1	Ciclo - Conformación del Conjunto de Datos:	54
5.3.1.1	Iteración- “Extracción de Datos web Saber UCV”: 54	
5.3.1.1.1	Especulación:	54
5.3.1.1.2	Colaboración:	55
5.3.1.1.3	Aprender:	56
5.3.1.2	Iteración- Levantamiento de Categorías:	57
5.3.1.2.1	Especulación:	57
5.3.1.2.2	Colaboración:	58
5.3.1.2.3	Aprender:	59
5.3.1.3	Iteración- Extracción y Clasificación de las Investigaciones:	60
5.3.1.3.1	Especulación:	60

5.3.1.3.2	Colaboración:	60
5.3.1.3.3	Aprender:	63
5.3.1.4	Objetivos alcanzados:	66
5.3.2	Ciclo-Prototipo del SCSU:	67
5.3.2.1	Iteración- Preparación del Corpus:	67
5.3.2.1.1	Especulación:	67
5.3.2.1.2	Colaboración:	67
5.3.2.1.3	Aprender:	71
5.3.2.2	Iteración - Recomendación Documentos:	73
5.3.2.2.1	Especulación:	73
5.3.2.2.2	Colaboración:	73
5.3.2.2.3	Aprendizaje:	74
5.3.2.3	Iteración- Implementación Prototipo:	74
5.3.2.3.1	Especulación:	74
5.3.2.3.2	Colaboración:	79
5.3.2.3.3	Aprender:	86
5.3.2.4	Objetivos Alcanzados:	87
5.3.3	Ciclo Integración de Componentes del Software:	88
5.3.3.0.1	Especulación:	88
5.3.3.0.2	Colaboración:	111
5.3.3.0.3	Aprender:	118
5.3.4	Ciclo Incorporación de otras investigaciones:	120
5.3.4.0.1	Especulación:	120
5.3.4.0.2	Colaboración:	120
5.3.4.0.3	Aprender:	121
5.3.5	Ciclo Buscador Semántico:	122
5.3.5.0.1	Especulación:	122

ÍNDICE

5.3.5.0.2	Colaboración:	124
5.3.5.0.3	Aprender:	125
5.4	Pruebas de aceptación:	127
5.4.1	Funcionales:	127
5.4.2	Rendimiento:	129
5.4.3	Relevancia:	130
6	Conclusiones:	132
6.1	Contribución:	132
6.2	Trabajos Futuros:	133
7	Apéndice: Librerías usadas y créditos	134

Índice de figuras

2.1	Arquitectura del Sistema-Este diagrama se va a traducir y adaptar al SCSU. Se coloca ya que es muy similar a la Solución Implementada	7
3.1	Gráfico que acompaña resultados de búsqueda de un término en la biblioteca digital de la Association for Computing Machinery (https://dl.acm.org/)	16
3.2	Coocurrencia de Palabras	27
3.3	Representación de palabras en un plano	33
3.4	Representación de palabra "king" mediante el modelo GloVe	35
3.5	Representación de palabras mediante el modelo GloVe	35
3.6	Evolución en la Cantidad de parámetros en los LLM	39
4.1	Representación de un tablero según la metodología Kanban	45
4.2	Ciclo ASD	46
5.1	Modelo de Arquitectura MVC	50
5.2	Etiquetas nodos url's	55
5.3	Listado de Maestrías	58
5.4	Total pregrados y postgrados por Facultad-Centro	59
5.5	Cantidades de categorías por facultad y nivel académico	62
5.6	Cantidades de investigaciones clasificadas por Facultad y por año de publicación	62
5.7	Histógrama Nombres Tutores Extraídos	63

ÍNDICE DE FIGURAS

5.8	Ejemplo de nombres simplificados	66
5.9	Arquitectura General pipeline Spacy	68
5.10	Detalle anotación del Corpus	69
5.11	Cantidad de palabras por función gramatical	69
5.12	Palabras Claves en investigaciones de la Escuela de Computación	71
5.13	Texto "Resumen" de Trabajo de Grado de Maestría en Química. Autora: Margarita González	72
5.14	Resultado de recomendaciones al documento: "Diseño de un sistema para la desinfección de aguas de consumo humano y de uso industrial empleando un material inorgánico antibacterial" . .	74
5.15	Caso de Uso 1 - Prototipo SCSU - Nivel 1	76
5.16	Caso de Uso 1.1 - Prototipo SCSU - Nivel 2	76
5.17	Mock-Up de campos de entrada en la Interfaz del Prototipo	78
5.18	Representación Mapas de Conocimiento	78
5.19	Modelo Entidad-Relación	79
5.20	Esquema General del Prototipo de la Aplicación	80
5.21	Estructura de datos Índice Invertido	81
5.22	Estructura de datos Índice Invertido	84
5.23	SCSU: Diagrama de Casos de Uso 1, Nivel 1.	95
5.24	SCSU: Diagrama de Casos de Uso 1, Nivel 1.	95
5.25	SCSU: Diagrama de Casos de Uso 2.	95
5.26	SCSU: Diagrama de Casos de Uso 3, Nivel 1	96
5.27	SCSU: Diagrama de Casos de Uso 3, Nivel 2.	96
5.28	SCSU: Diagrama de Casos de Uso 4.	96
5.29	SCSU: Diagrama Entidad Relación	104
5.30	SCSU: Diagrama General del Sistema en Contenedores	105
5.31	Esquema Extracción y Clasificación de Datos	106
5.32	Interfaz de Usuario - Sidebar	107

5.33	Interfaz de Usuario - Pestañas	107
5.34	Interfaz de Usuario - Tabla Resultados de Búsqueda	108
5.35	Interfaz de Usuario - Mapas de Conocimiento	108
5.36	Interfaz de Usuario - Tooltips	109
5.37	Interfaz de Usuario -Inspección Documento	109
5.38	Interfaz de Usuario- Inspección Mapas de Conocimiento	110
5.39	Diagrama Aplicación Web Shiny Versión Modular	110
5.40	Diagrama API- Registrar Embedding	123
5.41	Diagrama API- Consultar Embedding	123
5.42	Resultados de Búsqueda Semántica	125

Índice de cuadros

3.1	Embedding bidimensional para representar palabras	33
5.1	Cantidades de Trabajos por Categoría	54
5.2	Cantidades de Trabajos Duplicados	57
5.3	Cantidades de Postgrados por Categoría	58
5.4	Ratio de Cantidad Única de lemas Vs. Cantidad total de lemas . .	70
5.5	Prototipo SCSU UC.1	77
5.6	SCSU UC.1.	97
5.7	SCSU UC.1, Nivel 2.	98
5.8	SCSU UC. 2.	99
5.9	SCSU UC. 3.	100
5.10	SCSU UC. 3. Nivel 2.1.	101
5.11	SCSU UC. 3. Nivel 2.2.	102
5.12	SCSU UC. 4.	103
5.13	Pruebas de Caja Negra: Casos de Uso	128
5.14	Pruebas de Caja Negra: Requerimientos Funcionales	129

Capítulo 1

Introducción:

Es conocida la gran disponibilidad de información en distintos formatos que tienen a su disposición los investigadores. Ejemplo de esto son libros en las bibliotecas o la variedad de documentos que se encuentran accesibles en formatos digitales como artículos publicados en revistas arbitradas o investigaciones alojadas en repositorios digitales. Es en esta abundancia donde recaen varios de los problemas a los cuales se enfrenta la labor investigativa, en particular, cuando se realiza la fase exploratoria de selección de aquellos textos que puedan resultar de mayor importancia.

Una de las herramientas que tienen a su disposición los investigadores, para la búsqueda de información que se encuentra en formato digital, son los Sistemas de Recuperación de Información. En ellos, ante un texto que representa una necesidad, son recuperados los documentos que cumplen ciertos criterios y en la representación del resultado se deben jerarquizar los que poseen mayor relevancia (Manning et al., 2008).

En el caso de la Universidad Central de Venezuela existe un repositorio digital de documentos académicos denominado Saber UCV al que se puede acceder desde la dirección web saber.ucv.ve. Dentro de él se cuenta con un sistema que permite realizar búsquedas sobre las investigaciones generadas en la comunidad universitaria.

En este Trabajo de Grado se presenta la propuesta para la **Recuperación, Extracción y Clasificación de Información de Saber UCV**, mediante el desarrollo de un sistema informático que se denomina **Sistema Complementario Saber UCV (SCSU)**.

La implementación del SCSU es de utilidad para los investigadores que necesitan

1.1. ESTRUCTURA:

encontrar información sobre el conjunto de: tesis doctorales, trabajos de grado de maestría, trabajos especiales de grado, trabajos de ascenso y trabajos de grado; al ofrecer una mayor cantidad de parametros que permiten refinar la búsqueda.

Las funcionalidades que incorpora el SCSU se sustentan en procesos de extracción y clasificación de datos que originalmente no se encuentran estructurados en el repositorio oficial, como el nombre de la carrera de pregrado o el postgrado donde fue generada la investigación y el nombre del tutor correspondiente.

Igualmente se aplican métodos para el Procesamiento del Lenguaje Natural y Minería de Texto para representar “Mapas de Conocimiento” (Dueñas et al., 2011) y los resultados de la búsqueda son enriquecidos con recomendaciones de documentos similares.

Otra funcionalidad disponible en el SCSU es la “búsqueda semántica” la cual permite recuperar información relevante basada en el significado contextual y la relación semántica de los términos.

1.1 Estructura:

En el Capítulo 2 **El Problema** se hace una “Descripción del Problema” enfrentado, mientras que en 2.2 se hace la “Delimitación del Problema”. En 2.4 se expone la “Justificación e Importancia” de realizar este estudio y en 2.3 “Descripción de la Solución” se presenta la Solución junto con el 2.4.1 “Objetivo General” y los 2.4.2 “Objetivos Específicos”. En 2.5 se enumeran los “Aportes” que hace esta investigación.

En el Capítulo 3 **Marco Teórico** se hace una “Reseña Histórica” 3.1 sobre el problema computacional que representa la “búsqueda”. En la sección 3.2 “Recuperación de Información” se mencionan los conceptos claves para comprender los procesos de búsqueda de texto dentro de un Corpus y se introducen a los “Sistemas de Recuperación de Información” que permiten ejecutar dicha tarea. Luego en 3.2.3 se describen algunos “Modelos de Recuperación de Información” que se usan en la implementación del SCSU. Para representar los resultados de las búsquedas, el Sistema implementado necesita que los textos sean procesados mediante diversos métodos que son descritos en 3.3 “Procesamiento a los Textos”. Seguidamente en 3.4 se definen a los “Sistemas Distribuidos”, motivado a que la implementación realizada se hace mediante este tipo de

sistemas. Al finalizar este Capítulo en 3.5 se presentan el “Estado del Arte” en lo relativo a los procesos de Recuperación de Información.

En el Capítulo 4 **Marco Metodológico** se presenta la “Metodología de Trabajo Kanban” 4.1 que sirvió para realizar la planificación de la investigación y el “Desarrollo Adaptable de Software –Adaptive Software Development (ASD)”- 4.2 que fue la metodología adoptada para realizar los Ciclos de Desarrollo del SCSU.

En el Capítulo 5 **Desarrollo de la Solución** se presenta la 5.1 “Descripción General de la Solución”. En 5.2 se presenta la “Arquitectura” mientras que 5.3 se muestran los “Ciclos de Desarrollo” y sus correspondientes iteraciones.

Para concluir esta investigación, en el Capítulo 6 **Conclusiones** se exponen en 6.1 las “Contribuciones” y en 6.2 los “Trabajos Futuros” que se pueden derivar del proceso expuesto a lo largo del contenido de los capítulos anteriormente citados.

Capítulo 2

El Problema:

En el presente Capítulo en 2.1 se hace la **Descripción del Problema**. En 2.2 se hace la **Delimitación del Problema**. En 2.4 **Justificación e Importancia** se mencionan las razones principales que motivan haber llevado a cabo esta investigación. En 2.3 **Descripción de la Solución** se hace una aproximación a la propuesta general.

En 2.4.1 se menciona el **Objetivo Principal** que se trazó, mientras que en 2.4.2 se enumeran los **Objetivos Específicos** y en 2.5 se listan los **Aportes** efectuados.

2.1 Descripción del Problema:

La Universidad Central de Venezuela cuenta con un repositorio digital de documentos que se llama Saber UCV donde se alojan distintas investigaciones realizadas por la comunidad de la Universidad. Ingresando a la dirección web <http://saber.ucv.ve/> se permite “el acceso libre a la producción intelectual, materiales y recursos académicos elaborados en las áreas de docencia, investigación y difusión de la UCV”. ¿Qué es Saber UCV?. (2023). Saber UCV. <http://saber.ucv.ve>.

Los usuarios interesados en hacer búsquedas sobre la información académica alojada en el repositorio, pueden efectuarlas introduciendo un texto y aplicando filtros sobre distintas categorías como:

- Seleccionar “comunidades”: artículos de investigación, tesis (doctorales, maestrías, otras y pregrado), guías de estudio, revistas entre otras categorías.
- Seleccionar la “fecha de inicio” referente al año en que se efectuó la publicación.

Igualmente pueden realizar la búsqueda sobre atributos específicos como el título del documento, el nombre del autor o el texto del resumen.

Dadas estas funcionalidades, en algunos casos el investigador que acuda a este repositorio puede necesitar filtrar información por el criterio del área académica donde fue realizada la investigación, no siendo esto posible al no contar los criterios “nombre de facultad” y “nombre de escuela-postgrado”.

2.2 Delimitación del Problema:

El corpus con el que se trabajará es el subconjunto de los trabajos de grado de pregrado, trabajos especiales de grado de especialización, trabajos de grado de maestría, tesis doctorales y trabajos de ascenso, no abarcando otro tipo de documentos que se encuentran registrados en el repositorio.

No obstante en la implementación del ciclo de desarrollo 5.3.3 se incluyen dos revistas publicadas por la Universidad Central de Venezuela que son Gestión I+D y Episteme y una publicación del repositorio Saber U.L.A. de la Universidad de los Andes, para mostrar posibles ampliaciones en la ingesta de distintos tipos de publicaciones y fuentes dentro del SCSU.

Estas posibles incorporaciones implican procesos de revisión en las estructuras de los datos, ya que por los momentos el Sistema no está diseñado para automatizar tales procesos, sin la modificación de los códigos con los que se extrae la información desde los sitios web donde se alojan los documentos.

2.3 Descripción de la Solución:

La Solución que se propone es la implementación de un Sistema de Recuperación de Información con diversos componentes. Inicialmente de los trabajos mencionados en 2.2, mediante técnicas de extracción de nodos de archivos HTML's, se obtienen de Saber UCV los siguientes datos de cada investigación: el título, el nombre del autor, las palabras claves, la fecha de publicación, el resumen y el enlace de descarga del documento que soporta el trabajo de grado.

Posteriormente el Sistema descarga el documento anexo a cada investigación, da lectura y clasifica la información del nombre de la facultad, la escuela o postgrado

2.4. JUSTIFICACIÓN E IMPORTANCIA:

donde fue realizado el trabajo e igualmente extrae el nombre del tutor.

Todos los datos obtenidos son sometidos a técnicas del estado del arte en el Procesamiento del Lenguaje Natural y la Minería de Texto para conformar un corpus anotado, un índice invertido y se generan los vectores de *embeddings* de texto. Posteriormente las tablas obtenidas se almacenan en una base de datos indexada.

El Sistema cuenta con una aplicación web que permite a los usuarios desde un navegador, explorar extensivamente el corpus anotado, realizando consultas de texto y aplicando varios filtros como la selección de la jerarquía, el área académica y el rango de fechas. La relevancia de los resultados recuperados se determina mediante una función de ponderación y los documentos se presentan de manera priorizada para mejorar la experiencia del usuario.

Adicionalmente, el Sistema ofrece recomendaciones de documentos que presentan similitud con aquellos que fueron recuperados en el proceso anterior. También representa los “Mapas de Conocimiento”, según una adaptación *ad-hoc* hecha para la implementación del SCSU basada en el trabajo de (Dueñas et al., 2011), en una representación gráfica interactiva.

La solución implementada cuenta con procesos automatizados de actualización para incorporar las nuevas investigaciones que sean añadidas al repositorio Saber UCV y se soporta en un sistema distribuido conformado por contenedores que son gestionados por un orquestador con la arquitectura “modelo-vista-controlador”.

En la figura 2.1 se muestra un diagrama con la arquitectura del SCSU. (falta adaptarlo-pequeños cambios).

2.4 Justificación e Importancia:

Con esta Investigación se implementa un método que permite subsanar la falta de clasificaciones por áreas académica que tiene el repositorio Saber UCV y mediante la aplicación web se amplían los criterios de búsqueda disponibles para investigadores que necesiten realizar consultas sobre los documentos disponibles en el repositorio.

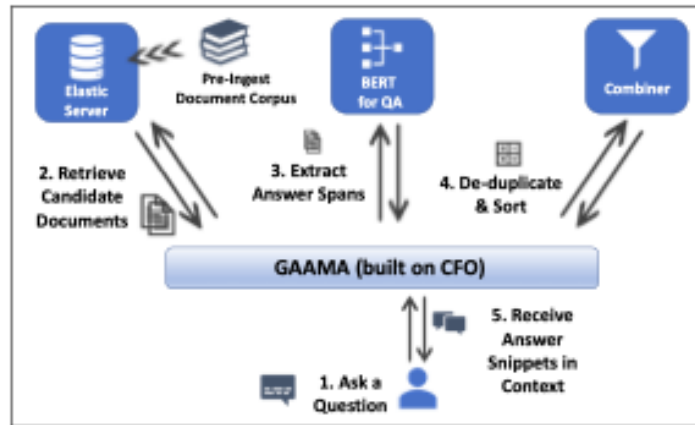


Figura 2.1: Arquitectura del Sistema-Este diagrama se va a traducir y adaptar al SCSU. Se coloca ya que es muy similar a la Solución Implementada

2.4.1 Objetivo General:

Implementar un Sistema en el que se puedan realizar procesos de recuperación de información (*information retrieval*), extracción y clasificación de los textos académicos alojados en el Repositorio Saber UCV, empleando técnicas de procesamiento de lenguaje natural, de minería de texto y usando modelos de lenguaje preentrenados de inteligencia artificial.

2.4.2 Objetivos Específicos:

1. Conformar un corpus con los resúmenes, títulos, palabras claves y nombres de autor con todos los documentos de tesis doctorales y trabajos de grado de pregrado y maestría alojados en Saber UCV.
2. Clasificar los trabajos de grado alojados en Saber UCV por área académica donde se haya realizado la investigación y extraer el nombre completo del tutor. (nota para el Prof. Mirabal: en este punto se puede separar, o colocar un nombre genérico para el objetivo como: creación de metadata a partir del texto contenido en los documentos de cada trabajo de grado).
3. Crear una aplicación web con una interface de navegación de corpus que facilite realizar “búsquedas de texto completo” o “búsqueda semántica” con acceso a información de relevancia que permita generar “Mapas de Conocimiento”.

2.5. APORTES:

4. Generar recomendaciones de investigaciones que presenten similitud con los documentos recuperados por el Sistema.

2.5 Aportes:

Algunos de los aportes que se generaron al realizar esta investigación son los siguientes:

- Se mejora y flexibilizan los criterios de búsqueda en comparación a los que tiene el repositorio Saber UCV al clasificar por área académica un total de 9.585 investigaciones de 9.982 potenciales documentos a categorizar.
- Se crea un listado con 425 categorías de carreras de pregrado, especializaciones, maestrías y doctorados que se imparten en la Universidad.
- Se obtiene un listado de potenciales tutores de las investigaciones realizadas con 8.217 nombres y una cantidad de 3.766 nombres únicos.
- Se crean visualizaciones y representaciones de “Mapas de Conocimiento”.
- Se implementa la búsqueda de texto completo con una función de relevancia distinta a la de Saber UCV permitiendo obtener mejores métricas de desempeño en la tarea de “recuperación de información”
- Se implementa un “sistema de recomendación” de documentos que presenten similitudes con los textos recuperados.
- La experiencia del usuario se enriquece mediante gráficos que muestran la evolución de aparición de los términos buscados en el período establecido para la recuperación de información.
- Todos los componentes del Sistema son de código abierto, libres de algún pago de licencia.
- La implementación se hace mediante un orquestador de contenedores, teniendo asociados los archivos “docker compose” y “dockerfiles”, que facilitan que pueda estar en producción el Sistema con un mínimo de configuraciones y se garantiza la reproducibilidad de los códigos que conforman la Solución.

- La propuesta es escalable y adaptable a la demanda de accesos.
- El Sistema está diseñado para poder actualizar los datos registrados con la periodicidad que se defina.
- El Sistema puede servir de prototipo para integrar publicaciones de otros repositorios de documentos que pertenecen a instituciones nacionales de investigación.
- Se realiza la implementación de “búsquedas semánticas”.

Capítulo 3

Marco Teórico-Referencial:

En este capítulo se exponen los fundamentos teóricos en que se sustentan los procesos y métodos aplicados en la investigación **Recuperación, Extracción y Clasificación de Información de SABER UCV**.

En 3.1 se hace una **Reseña histórica** sobre los procesos de búsqueda, en 3.2 se examina qué es la **Recuperación de Información (RI)**, en 3.2.1 se indica qué son los **Sistemas de Recuperación de Información (SRI)** y en 3.2.2 se muestran un par de ejemplos de SRI. Adicionalmente en 3.2.3 **Modelos de Recuperación de Información** se exploran modelos como el 3.2.3.1 **Boleano** y el 3.2.3.2 de **Índices Invertidos**. En la sección 3.2.4 se introduce el concepto de **Relevancia** que es clave para comprender cuál es uno de los principales objetivos que tiene cualquier SRI. En 3.2.5 **Re Ordenamiento** se exponen dos algoritmos usados para jerarquizar los documentos que sean más relevantes en un proceso de RI y en 3.2.6 **Métodos de Evaluación** se muestran algunas técnicas usadas para medir la precisión que se obtiene en los procesos de recuperación de información.

Para indicar los métodos con los que es tratado el Corpus de esta Investigación en 3.3 **Procesamiento de Texto** se muestran las técnicas del 3.3.1 **Procesamiento de Lenguaje Natural** y de 3.3.2 **Minería de Texto**. En 3.3.3 **Similitud de Documentos** se indica cómo se puede comparar la similitud de los textos y su aplicación.

Adicionalmente, al formar parte de esta Investigación, el desarrollo e implementación del software denominado **Sistema Complementario Saber UCV (SCSU)**, se exponen definiciones asociadas a los 3.4 **Sistemas Distribuidos**, sentando las bases para el desarrollo e implementación del SCSU.

Se finaliza este Capítulo haciendo una inspección en 3.5 a lo que constituye el

Estado del Arte en la materia, haciendo énfasis en la representación de textos mediante 3.5.1 **Embeddings**, en 3.5.2 se introduce la **Arquitectura de Redes Neuronales Transformers** con que se logran generar los embeddings de mayor calidad para el IR y brevemente se muestra en 3.5.3 los **Largos Modelos de Lenguaje (LLM's)**, mientras en 3.5.4 **Integración** se comentan los distintos avances y tendencias que se presentan en los Sistemas de Recuperación de Información.

3.1 Reseña histórica:

El profesor Donald Knuth señala, dentro del campo de las ciencias de la computación, que la **búsqueda** *es el proceso de recolectar información que se encuentra en la memoria del computador de la forma más rápida posible, esto cuando tenemos una cantidad N de registros y nuestro problema es encontrar el registro apropiado de acuerdo a un criterio de búsqueda* (Knuth, 1997, p.392). Iniciamos con esta cita porque la recuperación de información gira en torno a un problema central de las ciencias de la computación que es la **búsqueda**.

En la década de 1940 cuando aparecieron las computadoras, las búsquedas no representaban mayor problema debido a que estas máquinas disponían de poca memoria *RAM* pudiendo almacenar sólo moderadas cantidades de datos.

No obstante con el desarrollo e incremento del almacenamiento en memoria *RAM* o en dispositivos de almacenamiento permanentemente, ya en la década de 1950 empezaron a aparecer los problemas de **búsqueda** y los primeras investigaciones para afrontarla.

En la década de 1960 se adoptan estrategias basadas en arboles. Los primeros algoritmos que sirvieron para localizar la aparición de una frase dentro de un texto, o expresado de forma más abstracta, como la detección de una subcadena P dentro de otra cadena T , fueron los algoritmos de “*Pattern-Matching*” (Goodrich et al., 2013).

Así nos encontramos en la literatura con el algoritmo “Fuerza Bruta” donde dado un texto T y una subcadena P , se va recorriendo cada elemento de la cadena T para detectar la aparición de la subcadena P . Si bien este algoritmo no presentaba el mejor desempeño, creó una forma válida de enfrentar el problema de la búsqueda de subcadenas de texto.

3.1. RESEÑA HISTÓRICA:

El algoritmo “*Knuth-Morris-Pratt*” que se introdujo en 1976 tenía como novedad que se agregó una función que iba almacenando “previas coincidencias parciales” en lo que eran fallos previos y así al realizar un desplazamiento tomaba en cuenta cuántos caracteres se podían reusar. De esta forma se logró considerablemente mejorar el rendimiento en los tiempos de ejecución de $O^{\{(a+b)\}}$ que son asintóticamente óptimos.

Posteriormente en 1977 el problema se enfrenta con un nuevo algoritmo que es el de *Boyer-Moore* en el cual se implementan dos heurísticas (*looking-glass* y *character-Jump*) que permiten ir realizando algunos saltos en la búsqueda, ante la no coincidencia de la subcadena con la cadena y adicionalmente, el orden en el que se va realizando la comparación se invierte. Estas modificaciones permitieron obtener un mejor desempeño.

Sobre una modificación al algoritmo *Boyer-Moore* se sustenta la utilidad *grep* de la línea de comandos UNIX que también da soporte a diversos lenguajes que la usan para ejecutar búsquedas de texto, con un proceso que comúnmente es conocido como *grepping*. Esta utilidad fue ampliamente usada para resolver parcialmente lo que se expondrá en 5.3.1.

Otra de las estrategias que surgió para enfrentar las búsquedas de texto, fue el uso de la programación lineal, donde bajo la premisa “*divida et impera*”, los problemas que requieren tiempo exponencial para ser resueltos son descompuestos en polinomios y por lo tanto se disminuye la complejidad en tiempo para encontrar la solución.

Entre este tipo de algoritmos se puede mencionar los de *alineación de cadenas del ADN* de forma parcial o total dentro de una cadena mayor, siendo una versión de estos el algoritmo ***Smith-Waterman*** (Smith and Waterman, 1981). Posteriormente se identificó que este tipo de solución era extrapolable a las subcadenas de texto. Un algoritmo que se usó para resolver el problema de hacer coincidir una etiqueta de clasificación con los textos del *Corpus* en 5.3.1 fue el algoritmo precitado.

Un gran paso para aproximarnos a la aparición de los Sistemas de Recuperación de Información 3.2.1 lo representó el enfoque que presentan los algoritmos *Tries*. Este nombre proviene del proceso de *Information Retrieval* y principalmente se basa en hacer una serie de preprocesamientos a los textos para que al momento de ejecutar la búsqueda de texto, es decir, de la subcadena dentro de la cadena, ya tengamos una parte del trabajo realizado previamente y no tener que ejecutarlo todo “*on the fly*”, es decir, sobre la marcha.

Un *Trie* (Fredkin, 1960) es una estructura de datos que se crea para almacenar textos para así poder ejecutar más rápido la coincidencia en la búsqueda. En la propuesta del SCSU todos los textos van siendo preprocesados con distintas técnicas a medida que son insertados en la base de datos.

3.2 Recuperación de Información:

El eje central sobre el cual gira el proceso de recuperación de información (RI) es satisfacer las necesidades de información relevante que sean expresadas por un usuario mediante una consulta de texto, lo que también se denomina el *query*. El investigador Charu Aggarwal en su libro sobre Minería de Texto (Aggarwal and Zhai, 2012) menciona que el objetivo del proceso de RI es conectar la información correcta, con los usuarios correctos en el momento correcto, mientras que otro de los autores con mayor dominio sobre el tema, Christopher Manning en su libro *Information Retrieval* indica que “es el proceso de encontrar materiales (generalmente documentos) de una naturaleza no estructurada (generalmente texto) que satisface una necesidad de información dentro de grandes colecciones (normalmente almacenada en computadores)” (Manning et al., 2008).

A los efectos de delimitar el espacio de búsqueda sobre el que se realiza la acción de la consulta, se define al **Corpus** como el conjunto cerrado de documentos codificados electrónicamente que se encuentra integrado en un sistema de almacenamiento (Martín de Santa Olalla Sánchez, 1994) o entendido desde otra perspectiva, es el conjunto de datos sobre el cuál se hará la búsqueda generando el proceso de recuperación de información.

Satisfacer una necesidad de recuperación de información no sólo se circunscribe a un problema búsqueda de un texto dentro de un *corpus*. En la mayoría de los casos se deberá cumplir con ciertos criterios, o restricciones, como por ejemplo que el *query* esté dentro de un período de fechas, o que se encuentre limitado a ciertas restricciones, siendo esto a lo que se le denomina “búsqueda múltiple atributo”.

La información que se recolecte en una búsqueda y el orden en que sea presentado el resultado al usuario, dependerá de varios factores como: la aparición, parcial o total, de las palabras del *query* en el documento; se puede dar un mayor peso a la aparición de la frase del *query* dentro del título o palabras claves; la proximidad (la cercanía entre dos palabras) dentro del documento con las que se introducen

3.2. RECUPERACIÓN DE INFORMACIÓN:

en el *query*; o por la frecuencia de aparición de una palabra, o varias, dentro del documento. En 3.2.4 **Relevancia** se darán más detalles sobre este particular.

Igual puede aportar un peso mayor a la recuperación de un documento, las referencias (citas) que contengan otros documentos a ese determinado escrito, similar a la propuesta del algoritmo *PageRank* (Brin and Page, 1998), siendo el fin último, la extracción de los documentos que resulten de mayor relevancia para el usuario. Esta aproximación también puede usarse para la detección de comunidades dentro del *Corpus* (Heydari and Teimourpour, 2020).

Incluso es válido incorporar documentos, en los resultados que arroje la búsqueda, que propiamente no coincidan exactamente con los términos buscados sino que contengan palabras que sean sinónimos o que presenten alguna similitud con el texto del *query*. Lo que acabamos de mencionar incorporará formalmente dentro del proceso de extracción de información algo de imprecisión con la intención última de enriquecer el proceso de *Information Retrieval* (Kraft and Colvin, 2017). En 3.3.3 **Similitud de Documentos** y en 3.5.1 **Embeddings**, se mencionan y especifican algunas de las técnicas con las cuales se incorporan este lote de documentos en los resultados de una búsqueda.

Evaluando el proceso con cierto nivel de abstracción se tiene que el proceso de recuperación de información está compuesto principalmente:

- por un *query*
- por un corpus y
- por una función de *ranking* 3.2.5 para ordenar los documentos recuperados de mayor importancia a menor.

El desarrollo de los algoritmos expuestos en 3.1, sumado a la necesidad de resolver los problemas asociados a la búsqueda de un texto dentro de un *corpus* con múltiples atributos, en tiempos aceptables y el crecimiento exponencial de datos disponibles en formato digital (wor, 2016), potenciada por el uso generalizado de los computadores desde la década de 1980, abonó las condiciones para la creación de los **Sistemas de Recuperación de Información**.

3.2.1 Sistemas de Recuperación de Información (SRI) :

Los Sistemas de Recuperación de Información (*Information Retrieval Systems-IRS*) son los dispositivos (software y/o hardware) que median entre un potencial usuario que requiere información y la colección de documentos que puede contener la información solicitada (Kraft and Colvin, 2017) 1. El SRI se encargará de la representación, el almacenamiento y el acceso a los datos que estén estructurados y se tendrá presente que las búsquedas que sobre él recaigan tendrán distintos costos, siendo uno de estos el tiempo que tarde en efectuarse.

Es de nuestro conocimiento que generalmente los datos estructurados son gestionados mediante un sistema de base de datos, pero en el caso de los textos, estos se gestionan por medio de un motor de búsqueda, motivado a que los textos en un estado crudo carecen de estructura (Aggarwal and Zhai, 2012) . Son los motores de búsqueda (*search engines*) los que permiten que un usuario pueda encontrar fácilmente la información que resulte de utilidad mediante un *query*.

El SCSU está diseñado como un SRI donde se pueden ejecutar queries, que son procesados y los resultados que se obtienen, son sometidos a una función de *ranking* que será expuesta en una fase posterior del desarrollo de esta investigación.

3.2.2 Ejemplos de IRS:

Profundizando en el tema de esta Investigación se mencionan un par de páginas de internet que funcionan como IRS sobre corpus de investigaciones científicas.

1. Arxiv alojado en <https://arxiv.org/>, que es un repositorio de trabajos de investigación. Al momento del usuario hacer un requerimiento de información, adicional al texto de la búsqueda, se pueden indicar distintos filtros a aplicar como puede ser el área del conocimiento (física, matemática, computación, etc.), si se quiere buscar sólo dentro del título de una investigación, o el nombre autor, en el *abstract*, o en las referencias.
2. Portal de la *Association Computery Machine* (ACM) alojado en <https://dl.acm.org> incorpora un motor de búsqueda con particulares características ya que los resultados son acompañado por distintas representaciones gráficas que le dan un valor agregado. En la figura 3.1 se ve una de estas representaciones que incluye la frecuencia de aparición del *query* en el tiempo.

3.2. RECUPERACIÓN DE INFORMACIÓN:

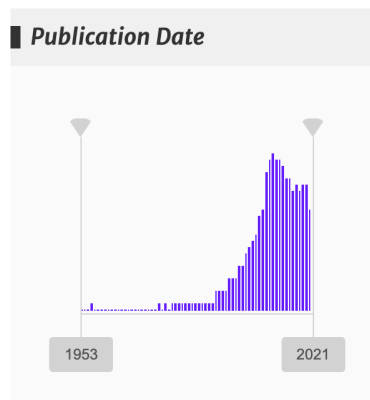


Figura 3.1: Gráfico que acompaña resultados de búsqueda de un término en la biblioteca digital de la Association for Computing Machinery (<https://dl.acm.org/>)

3.2.3 Modelos de Recuperación de Información:

3.2.3.1 Recuperación booleana:

Ante una búsqueda de información se recorre linealmente todo el documento para retornar un valor booleano indicando la presencia o no del término buscado. Es uno de los primeros modelos que se usó y está asociado a técnicas de *grepping* (Manning et al., 2008). El desarrollo de este modelo apareció entre 1960 y 1970.

El usuario final obtendrá como respuesta a su *query* sólo aquellos textos que contengan el término. Es un modelo muy cercano a los típicos *queries* de bases de datos con el uso de operadores “AND”, “OR” y “NOT”. En el procesamiento de los textos se genera una matriz de incidencia binaria término-documento, donde cada término que conforma el vocabulario, ocupa una fila i de la matriz mientras que cada columna j se asocia a un documento. La presencia de el término i en el documento j se denotará con un valor verdadero o un “1”.

La recuperación booleana si bien representa una buena aproximación a la generación de *queries* más rápidos, presenta una gran desventaja y es que al crecer la cantidad de documentos y el vocabulario (palabras únicas contenidas dentro del Corpus), se obtiene una matriz dispersa de una alta dimensionalidad que hace poco efectiva su implementación.

Los documentos y los *queries* son vistos como conjuntos de términos indexados, que luego fueron llamados “bolsa de términos” (*bag of terms*). Las deficiencias de este modelo recaen en que los resultados, no tienen ningún ranking. Si por ejemplo el término sobre el cual se realiza el *query* aparece 100 veces en un documento y en

otro aparece sólo una vez, en la presentación de los resultados ambos documentos aparecerán al mismo nivel, no pudiendo mostrar preferencia del uno sobre el otro.

Otra de las desventajas es que no se registra el contexto semántico de las palabras e incluso se pierde el orden en que aparecen las palabras en cada texto.

Este modelo se presume que es el cual se basa la implementación de Saber UCV y por eso es que en general, se termina presentando el problema de que al usar el operador OR en las búsquedas exactas, se obtiene un gran *recall*¹ en los resultados.

Con la propuesta del 5.3.2 Prototipo de Buscador del SCSU se obtiene una versión de recuperación de información que aplica métodos de mayor eficiencia y genera una mayor “*precision*” con un menor “*recall*”, mejorando la relevancia de los resultados. En 3.2.6 **Evaluación** se indicarán qué son estas métricas y algunos métodos para medirlas.

3.2.3.2 Índices Invertidos:

Se denominan índices invertidos porque en vez de guardar los documentos con las palabras que en ellos aparecen, en estos se procede a guardar cada palabra y se indica los documentos en los cuales se encuentra y adicionalmente se puede registrar la posición en que aparece cada palabra con distintas granularidades, pudiendo ser estas: dentro del documento, del capítulo, del párrafo o de la oración. También pueden contener la frecuencia con que se presenta determinada cada palabra. Toda esta información nos permite mejorar los tiempos de búsqueda pero con ciertos costos.

El primero es el espacio en disco que implica guardar estos datos adicionales, que puede oscilar del 5% al 100% del valor inicial de almacenamiento, mientras que el segundo costo lo representa el esfuerzo computacional de actualizarlos una vez que se incorporan nuevos documentos (Mahapatra and Biswas, 2011).

Existen diversos tipos de **Índices Invertidos** y constantemente se están realizando investigaciones que permitan mejorar su desempeño motivado en que sobre ellos recae gran parte de la efectividad que podamos obtener ejecutando los *queries*. Algunos ejemplos de estos índices son el *Generalized Inverted Index*

¹Precision: la fracción, o porcentaje, de los documentos recuperados que son relevantes en la búsqueda efectuada. Recall, también denominado en español como “exhaustividad” mide la fracción de documentos relevantes que fueron recuperados con respecto a la totalidad de los documentos relevantes presentes en la base de datos.

3.2. RECUPERACIÓN DE INFORMACIÓN:

(GIN), también está el RUM ² o el VODKA ³ que es otra implementación con menos literatura sobre posibles usos pero con métodos disponibles para su uso en manejadores de base de datos como PostgreSQL que es el que soporta al SCSU.

El espacio que ocupa la implementación de estos índices se puede ver reducido, por un lado mediante el preprocesamiento que hagamos a las palabras buscando su raíz con el *stemming* 3.3.1.3 o removiendo las *stop words* (las palabras que no generan mayor valor semántico como: la, el, tu, son, etc.).

Por otra parte el peso total se puede incrementar a medida que decidamos tener una granularidad más fina en el registro de las palabras y su ubicación dentro de los documentos. En el transcurso del desarrollo de nuestra investigación se indicará en cuánto se incremento el espacio de almacenamiento en disco con la aplicación de este índice y la granularidad que se adoptó.

Continuando con los índices inversos, existen estrategias que significan la adopción de generar dos índices inversos para un sistema, conteniendo uno de estos la lista de documentos y la frecuencia de la palabra, mientras que el otro registra la lista con las posiciones de la palabra.

El uso de los índices invertidos permite la denominada “búsqueda de texto completa” (*full text search*) que es uno de los pilares que sustenta a los motores de búsqueda y se entiende por este tipo de búsqueda aquella que permite encontrar documentos que contienen las palabras clave o frases determinadas en el texto del *query*. Adicionalmente se puede introducir el criterio de búsqueda de texto aproximado (*approximate text searching*), donde se flexibiliza la coincidencia entre el texto requerido y el resultado.

En la Solución que se propone, la optimización en la generación de este índice quedará bajo la administración del propio manejador de base de datos que es *postgreSQL*.

Cuando la base de datos que registra el índice invertido crece y no es viable almacenarla en un único computador, es necesario acudir al uso de técnicas que permitan distribuir la base de datos con el uso de tecnologías como *Spark*, *Hadoop*, *Apache Storm* entre otras. En el trabajo de (Mahapatra and Biswas, 2011) se encuentran detalles adicionales sobre este tipo de índices.

²En el vínculo <https://github.com/postgrespro/rum> se tiene acceso a la explicación e implementación de este índice para PostgreSQL.

³este índice fue presentado en la Postgres Conference en el año 2014 https://www.pgcon.org/2014/schedule/attachments/318_pgcon-2014-vodka.pdf

En 3.5 se muestra el **Estado del Arte** en los Sistemas de Recuperación de Información al incorporar representaciones de **Embeddings** (Reimers and Gurevych, 2019a) para los textos y su uso como un Modelo de Recuperación de Información.

3.2.4 Relevancia:

Refiere la medida en que un documento o recurso recuperado satisface las necesidades de información del usuario. En otras palabras, un documento es relevante si contiene información que es útil y está relacionada con el *query* realizado por el usuario (Büttcher et al., 2010a). La relevancia no es una propiedad intrínseca del documento, sino que depende del contexto y de las necesidades de información del usuario en un momento específico.

3.2.5 Re Ordenamiento (re-ranking):

Es una técnica utilizada para mejorar la precisión y lograr extraer los documentos que tengan mayor relevancia 3.2.4 en los resultados de una búsqueda. Cuando los usuarios realizan el *query* a menudo se encuentran con una gran cantidad de documentos que coinciden con sus consultas. Sin embargo, no todos estos documentos son igualmente relevantes para el usuario. Por lo tanto, el re-ranking implica reorganizar los resultados de búsqueda originales para que los documentos más relevantes aparezcan en las primeras posiciones, mejorando así la experiencia del usuario.

3.2.5.1 Learning to Rank (LTR):

Los algoritmos de aprendizaje para la clasificación (LTR, por sus siglas en inglés) son comúnmente utilizados para el re-ranking. En ellos se utilizan técnicas de aprendizaje automático para modelar la relevancia de los documentos basándose en características específicas (Büttcher et al., 2010b). Los atributos pueden incluir la frecuencia de palabras clave, la proximidad de términos en el documento y otros factores que indican la relevancia. Los modelos LTR pueden ser entrenados con conjuntos de datos que contienen consultas y documentos etiquetados con su relevancia, y luego aplicados para re-ordenar los resultados de búsqueda en función de las características aprendidas.

3.2. RECUPERACIÓN DE INFORMACIÓN:

3.2.5.2 BM25:

Es un algoritmo que apareció a mediados de la década de 1990 el cual contiene una función de puntuación basada en un modelo probabilístico que es utilizada para calcular la relevancia de un documento con respecto a una consulta (Robertson and Zaragoza, 2009) y ha demostrado ser efectivo en la práctica para clasificar documentos según su relevancia con las consultas de los usuarios, llegando en su momento a compararse a que obtenía un rendimiento similar al humano, al hacer los procesos de ranking sobre las colecciones (Trotman et al., 2014) de documentos TREC (Voorhees et al., 2005). Se basa en la frecuencia de los términos de búsqueda y la longitud del documento. A diferencia de los modelos clásicos como TF-IDF, BM25 ajusta la importancia de la frecuencia del término y la longitud del documento mediante una fórmula matemática compleja (Zhai and Massung, 2016), lo que lo hace más eficaz para lidiar con variaciones en la longitud del documento y mejorar la precisión en los resultados de búsqueda.

3.2.6 Medidas y Métodos de Evaluación:

Las siguientes métricas son usadas en el campo de la recuperación de información:

1. **Precision (Precisión)** : Es la proporción de documentos relevantes recuperados por el sistema con respecto a todos los documentos recuperados. Cuanto mayor es la precisión, menos documentos irrelevantes se recuperan.
2. **Recall (Recuperación)**: Es la proporción de documentos relevantes recuperados por el sistema con respecto a todos los documentos relevantes presentes en la base de datos. Un alto “recall” indica que el sistema encuentra la mayoría de los documentos relevantes.
3. **F1 Score**: Es la media armónica de precisión y recall. Proporciona un equilibrio entre precisión y recall. Un F1 Score alto indica un buen equilibrio entre la precisión y la capacidad para encontrar todos los documentos relevantes.

Una vez enunciado el concepto de estas tres medidas es necesario determinar el proceso con que se puede determinar la “relevancia” de los documentos recuperados. Para explicar esto se expone el origen de este método.

Posterior a la segunda guerra mundial se incrementó considerablemente la publicación de investigaciones en el ámbito científico y se hizo necesario contar con sistemas analógicos que fuesen eficientes para la indexación de los documentos. En el estudio denominado “Cranfield Tests”(Harman, 2011), que fue conducido por Cyril Cleverdon, a partir de 1958 se empezaron a definir los estándares para evaluar la efectividad de los índices disponibles para aquel momento.

Desde ese entonces quedó definido el concepto de “relevancia” ante los resultados obtenidos en un proceso de búsqueda documental, siendo bastante similar al que actualmente se denomina “precisión” .

Una de las estrategias que se llevó a cabo en el proyecto fue usar el “*known-item searching*” (búsqueda del elemento conocido), que consistía en encontrar un documento que garantizara ser relevante ante una determinada pregunta. Para obtener la dupla “pregunta-nombre documento” más relevante, acudieron a los autores de 1500 documentos y les pidieron que formularan una pregunta que satisfactoriamente iba a ser respondida en el documento de su autoría (Harman, 2011).

Hoy en día se han construido distintos conjuntos de datos constituidos por la dupla “pregunta - identificación de documento con respuesta correcta”, para evaluar mediante las métricas “precisión” y “recall”, la efectividad que tiene un sistema de recuperación de información. Este tipo de conjuntos de datos se denominan las “Standard Test Collections”, que es un conjunto de juicios de relevancia por parte de expertos, dados como una expresión binaria: “relevante” o “no relevante” para la dupla “query- documento”.

Con esto se puede conformar una aproximación a una “gold standard” o “ground truth judgment of relevance” (juicio de pertinencia basado en la verdad).

Desde inicios de la década de los 90, con las reuniones periódicas de la denominada “Text REtrieval Evaluation Conference-TREC” se crean distintos conjuntos de datos con diversos documentos agrupados por temas, donde expertos anotan juicios de relevancia para indicar cuáles son los documentos más relevantes para cada uno de los temas.

Así se introdujo una de las metodologías que es usada para evaluar la “relevancia” obtenida, al comparar la que ejecuta el sistema de recuperación de información con la que fue realizada por expertos.

3.3. PROCESAMIENTOS A LOS TEXTOS:

El problema que presenta este enfoque es que ante métodos de recuperación de documentos más avanzados, como la búsqueda semántica, que será presentada más adelante, y ante el incremento de documentos digitales y también de temas de investigación más específicos, este tipo de mediciones se queda un tanto rezagada y no muestra la real efectividad de los sistemas.

También hay que indicar es que las medidas “*precisión*” y “*recall*” pueden no necesariamente reflejar la satisfacción del usuario, ya que esta en muchos casos se ve afectada es por el grado de satisfacción con la interface de usuario que presente el Sistema de Recuperación de Información (Manning et al., 2008).

3.3 Procesamientos a los textos:

En esta sección mostramos métodos de manipulación y tratamiento de los textos. Lo primero que se indica es que hasta el año 2016 eran escasas las herramientas computacionales para el procesamiento de los textos 3.3.1 en el idioma español. Sabiendo que son justamente los textos, el insumo que recibe el Sistema propuesto en esta Investigación, la calidad en los procesamientos que sobre ellos se hagan, marcarán en gran medida la propia calidad del Sistema que se obtenga.

Frameworks para tareas de procesamientos de texto se basan en los proyectos de “*Universal Dependencies*” (de Marneffe et al., 2021), como es el caso del “coreNLP” de la Universidad de Stanford (Manning et al., 2014) que fue uno de los primeros sistemas en incluir procesamientos para el idioma español, sin disponer todas las utilidades que sí era viable realizar con textos en el idioma inglés, como la identificación de parte del discurso (*Part of Speech Tagging*) 3.3.1.2, ni el análisis morfológico (*Morphological Analysis*) (Straka and Straková, 2017) o el reconocimiento de entidades nombradas (*Named Entity Recognition*), sino algunas pocas como el tokenizador 3.3.1.1 y el separador de oraciones (*Sentences Splitting*).

Casos similares se presentaban con otras herramientas, siendo un caso aparte el esfuerzo del “CLiC- Centre de Llenguatge i Computación” quienes hicieron la anotación del Corpus AnCora⁴. También la Universidad Politécnica de Cataluña

⁴ **AnCora** es un corpus del **catalán (AnCora-CA)** y del **español (AnCora-ES)** con diferentes niveles de anotación como lema y categoría morfológica, constituyentes y funciones sintácticas, estructura argumental y papeles temáticos, clase semántica verbal, tipo denotativo de los nombres deverbales, sentidos de WordNet nominales, entidades nombradas (NER), relaciones de correferencia (<http://clic.ub.edu/corpus/es/ancora>)

creó la herramienta FreeLing⁵ que implementó para el español, y catalán, algunas de las funcionalidades con que sí disponía para el idioma inglés el “coreNLP”. No obstante, su integración en cadenas de trabajo y la actualización de sus modelos de entrenamiento, presentaron rezagos en comparación a otros modelos que actualmente se están usando, basados en el uso del aprendizaje mediante redes neuronales (Chen and Manning, 2014b) y que serán indicados con mayor detalle en la sección **Estado del Arte** 3.5 .

3.3.1 Procesamiento del Lenguaje Natural (Natural Language Processing-NLP):

El Procesamiento del Lenguaje Natural (PNL), son el conjunto de técnicas computacionales desarrolladas para permitir al computador representar e interactuar de una forma más efectiva con el “significado” de los textos . Al aplicar la *tokenización* 3.3.1.1 , el Etiquetado de Partes del Discurso 3.3.1.2, el *stemming* 3.3.1.3, la *lematización* 3.3.1.4 , entre otros métodos, se desea obtener un Corpus Anotado (Desagulier, 2017). Los métodos que se detallan a continuación fueron aplicados sobre el Corpus del SCSU.

3.3.1.1 Tokenizador:

Básicamente es separar el documento en palabras, o unidades semánticas que tengan algún significado a las cuales se le llaman *tokens* (Straka and Straková, 2017). Para el idioma español no representa un mayor reto, ya que se puede usar el espacio como delimitador de palabras, no así en otros idiomas como el chino donde el problema se aborda de manera distinta.

Al obtener las palabras como entidades separadas de un texto nos permite, por ejemplo, calcular la frecuencia de uso de las mismas.

Es común que las librerías de procesamiento de lenguaje natural contengan tokenizadores que presentan un 100% como métrica de precisión en el idioma español.

Hay que destacar que los tokenizadores para generar “embeddings”, ver 3.5.1, se comportan en algunos casos de forma distinta al hacer la separación de las unidades que conforman el texto.

⁵<https://nlp.lsi.upc.edu/freeling/node/1>

3.3. PROCESAMIENTOS A LOS TEXTOS:

3.3.1.2 Etiquetado de Partes del Discurso (*Part of speech tagging-POS*):

Consiste en asignar un rol sintáctico a cada palabra dentro de una frase (Eisenstein, 2019) siendo necesario para ello evaluar cómo cada palabra se relaciona con las otras que están contenidas en una oración y así se revela la estructura sintáctica.

Los roles sintácticos principales de interés en la elaboración de esta Investigación son los sustantivos, adjetivos y verbos.

- Los sustantivos tienden a describir entidades y conceptos.
- Los verbos generalmente señalan eventos y acciones.
- Los adjetivos describen propiedades de las entidades

Igualmente dentro del POS se identifican otros roles sintácticos como los adverbios, nombres propios, interjecciones entre otros.

El POS es un procesamiento que sirve de insumo para determinar la coocurrencia de palabras, que es una de las formas en que se representan los resultados de los *queries* en el SCSU.

En el estado del arte este etiquetado alcanza un 98% de precisión.

3.3.1.3 Stemming:

Stemming es un algoritmo que persigue encontrar la raíz de una palabra, teniendo como el de mayor uso el Algoritmo de Porter (Willett, 2006). Al ser usado se puede reducir considerablemente el número de palabras que conforman el vocabulario del *corpus* y así se mejoran los tiempos en que se ejecuta la búsqueda de un texto, ya que se disminuye el espacio de búsqueda. La aplicación de este tipo de algoritmos no toma en consideración el contexto en el que aparece la palabra a la que se le extrae la raíz. Como ejemplo se muestra que “yo canto, tú cantas, ella canta, nosotros cantamos, ellos cantan” donde todas las palabras tendrán como raíz “cant”.

Es necesario considerar que al crear el **índice invertido** 3.2.3.2 son las raíces las que se guardarán y no propiamente la palabra que aparece en el texto.

3.3.1.4 Lematización:

Es el proceso en que se consigue el *lema* de una palabra, entendiendo que el *lema* es la forma que por convenio se acepta como representante de todas las formas flexionadas de una misma palabra (de Marneffe et al., 2021). Los lemas, o lexemas, constituyen la parte principal de la palabra, la que transmite el significado. Los morfemas son el elemento variable de la palabra y son los que se busca desechar en el proceso de lematización.

Al buscar el *lema* se tiene presente la función sintáctica que tiene la palabra, es decir que se evalúa el contexto en el que ocurre. Una de las ventajas de aplicar esta técnica es que se reduce el vocabulario del Corpus y eso conlleva a que también se reduzca el espacio de búsqueda.

Un ejemplo de lematización se puede representar con estas tres palabras: “bailaré, bailamos, bailando” que tienen el mismo *lema* que es “bailar”.

En el estado del arte este etiquetado alcanza un 96% de precisión en varios de los modelos de aprendizaje automático preentrenados, no obstante no se disponen datos puntuales de esta métrica para el idioma español.

3.3.2 Minería de Texto:

La extracción de ideas útiles derivadas de textos mediante la aplicación de algoritmos estadísticos y computacionales, se conoce con el nombre de minería de texto, analítica de texto o aprendizaje automático para textos (*text mining*, *text analytics*, *machine learning from text*). Se quiere con ella representar el conocimiento en una forma más abstracta y así poder detectar relaciones en los textos (Aggarwal, 2018a).

La minería de texto surge para dar respuesta a la necesidad de tener métodos y algoritmos que permitan procesar estos datos no estructurados (Aggarwal and Zhai, 2012) y ha ganado atención en recientes años motivado a las grandes cantidades de textos digitales que están disponibles. Los procesamiento inherentes al NLP mencionados anteriormente son insumo para la minería de texto.

Algunos de los métodos que pertenecen a la Minería de Texto son:

3.3. PROCESAMIENTOS A LOS TEXTOS:

3.3.2.1 Term-Document Matrix:

Una vez que se tiene conformado un Corpus, se procede a conformar una matriz dispersa de una alta dimensionalidad que se denominará “*Sparse Term-Document Matrix*)” de tamaño $n \times d$, donde n es el número total de documentos y d es la cantidad de términos o vocabulario (palabras distintas) presentes entre todos los documentos. Formalmente se sabe que la entrada (i,j) de nuestra matriz es la frecuencia (cantidad de veces que aparece) de la palabra j en el documento i . Este procedimiento es similar al que fue revisado en 3.2.3.1.

Uno de los problemas que presenta la matriz obtenida es la alta dimensionalidad y lo dispersa que es, llegando a estar conformada en un 98% por ceros, que indican la ausencia de la aparición de una palabra en un determinado documento.

Para mejorar un tanto este tipo de representación del Corpus, se aplican otras técnicas, que en principio puedan colaborar a reducir la dimensionalidad, por medio de simplificar los atributos, es decir, disminuyendo el vocabulario aplicando el stemming 3.3.1.3.

3.3.2.2 Coocurrencia de Palabras:

En esta investigación se usará un método denominado “Coocurrencia de Palabras” para la detección de patrones en los textos y se hará la representación de aparición de las coocurrencias mediante grafos.

El método se explica en que se evalúan las palabras que coocurren, es decir, aquellas que forman parte del conjunto de palabras obtenidas de la intersección de los documentos que conforman el *corpus*, o del subconjunto de documentos recuperados mediante un determinado *query*.

También se puede establecer el nivel al que se quiere determinar la coocurrencia, por ejemplo, las palabras que coocurren una seguida de otra en los textos, o las que coocurren dentro de la misma oración, o dentro de un párrafo o dentro de todo el texto de cada documento.

Para la representación visual se usan los grafos, donde cada palabra representa un nodo y la coocurrencia de una palabra con otra implica que se extienda un arco entre ellas. Las palabras dispuestas para representarse en el grafo serán exclusivamente las que tengan la función dentro del discurso (POS) 3.3.1.2 de adjetivos y sustantivos, es decir que cada coocurrencia será un sustantivo con el

adjetivo que la acompaña, donde es posible tener una relación de un sustantivo con $\{0,1,\dots,n\}$ adjetivos.

La selección de las funciones gramaticales propuestas se hace para disminuir el espacio de representación y se considera que los sustantivos, al contar con el adjetivo que las acompaña, logran hacer una representación que muestra proximidad semántica y se representan los temas (*tópicos*) más relevantes (Segev, 2021).

En la figura 3.2 se visualiza lo expuesto de una manera gráfica al ver la representación en un grafo de la coocurrencia de palabras sobre los textos de los resúmenes de las Tesis y TEG de la Escuela de Física de la U.C.V.

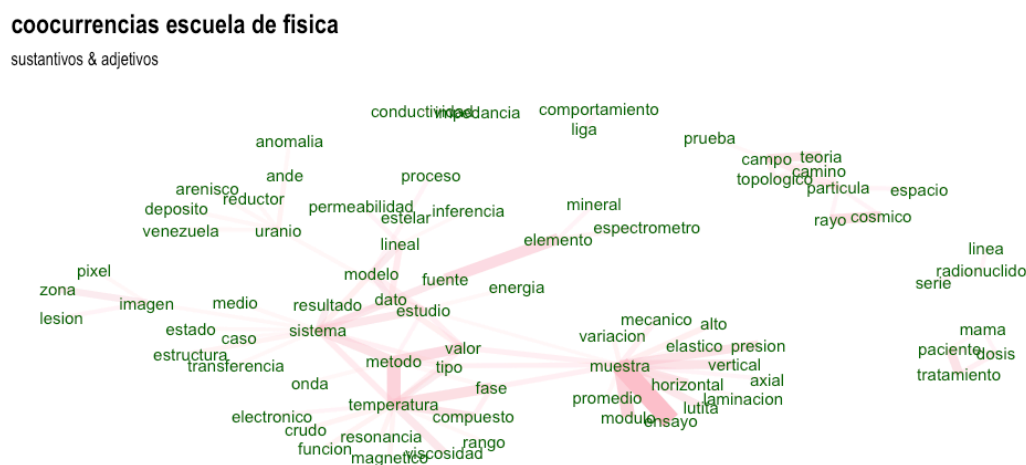


Figura 3.2: Coocurrencia de Palabras

3.3.2.2.1 Mapas de Conocimiento: La representación gráfica y el método de extracción de sustantivos y adjetivos, resulta similar a la propuesta metodológica realizada por (Dueñas et al., 2011) para crear “Mapas de Conocimiento” con “las palabras claves obtenidas a través de búsquedas recurrentes y relacionadas”. En esta Investigación se simplificará la obtención y representación de estos Mapas, asumiendo que las palabras claves son los sustantivos adjetivizados, equivalente a visualizar las personas, cosas o ideas que se mencionan y que son modificados por los adjetivos, al cambiar sus propiedades o atributos; seleccionando aquellas palabras que muestran una mayor aparición en el *query* realizado y que se interconectan mediante arcos.

3.3.3 Similitud de documentos:

Para poder realizar la recomendación de documentos, una de las técnicas que se usa es medir la similitud que presenta un documento con los otros contenidos en el corpus (Aggarwal, 2018a) . Un ejemplo de esta técnica es el uso de la similitud coseno que se explica con esta fórmula.

$$\cos(\mathbf{t}, \mathbf{e}) = \frac{\mathbf{t} \cdot \mathbf{e}}{\|\mathbf{t}\| \|\mathbf{e}\|} = \frac{\sum_{i=1}^n \mathbf{t}_i \mathbf{e}_i}{\sqrt{\sum_{i=1}^n (\mathbf{t}_i)^2} \sqrt{\sum_{i=1}^n (\mathbf{e}_i)^2}} \quad (3.1)$$

En la fórmula t representa un documento y e representa otro documento. Ambos documentos se asumen que están en un espacio con i atributos, o dimensiones, y la intención es calcular un índice de similitud entre ambos documentos.

Este es uno de los métodos más usados para detectar similitudes en los textos, aunque existen otras fórmulas para el cálculo de la similitud como los es el índice de Jaccard.

Al hacer la comparación de un documento i del Corpus que contiene n documentos, en un proceso iterativo con otra cantidad de $(n-1)$ documentos, se obtendrán $(n-1)$ índices de similitud. Aquel que obtenga un mayor valor se puede inferir que presenta una mayor similitud con el documento i .

El otro elemento de gran importancia en el resultado que se obtenga de esta medición, es la representación computacional que se haga del documento. Son distintas las técnicas que existen estando entre ellas la representación mediante “bolsas de palabras” o *bag of words*, similar a lo que se explicó en 3.3.2.1 donde un documento i es el vector correspondiente a una fila de la matriz y la cantidad de dimensiones que presenta es equivalente al tamaño del vocabulario.

La aplicación de realizar este tipo de comparaciones mediante la estimación de la similitud, es que ante un proceso de *query* también pueden ser recuperados, o sugerir al investigador, en la entrega de los documentos recuperados, aquellos documentos que también presenten alguna similitud, a manera de que el propio sistema tenga la capacidad de realizar recomendaciones.

Recientemente se han creado formas más complejas para la representación de los documentos, como lo son los *word embeddings* que son obtenidos mediante el entrenamiento de redes neuronales de aprendizaje profundo, lo que será expuesto en 3.5.1.

3.4 Sistemas Distribuidos:

Los distintos procesos y componentes de la Solución propuesta han sido diseñados e implementados como un sistema distribuido y por eso se hace la mención a este tema.

Una definición formal que se le puede dar a los sistemas distribuidos es “cuando los componentes de hardware y/o software se encuentran localizados en una red de computadores y estos coordinan sus acciones sólo mediante el pase de mensajes” (Coulouris, 2012).

Algunas de las principales características que tienen los sistemas distribuidos es la tolerancia a fallos, compartir recursos, concurrencia, ser escalables (Czaja, 2018) entre otras. Mencionamos estas, en particular, al ser propiedades que están presentes en la propuesta acá descrita:

1. Fiabilidad o la llamada tolerancia a fallos: en caso de fallar un componente del sistema los otros se deben mantener en funcionamiento.
2. Compartir recursos: un conjunto de usuarios pueden compartir recursos como archivos o base de datos.
3. Concurrencia: poder ejecutar varios trabajos en simultáneo.
4. Escalable: al ser incrementada la escala del sistema se debe mantener en funcionamiento el sistema sin mayores contratiempos.

3.4.1 Contenedores:

Un contenedor es una abstracción de una aplicación que se crea en un ambiente virtual, en el cual se encuentran “empaquetados” todos los componentes (sistema operativo, librerías, dependencias, etc.), que una aplicación necesita para poder ejecutarse. En su diseño se tiene presente que sean ligeros y que con otros contenedores pueden compartir el *kernel*, usando un sistema de múltiples capas, que también pueden ser compartidas entre diversos contenedores, ahorrando espacio en disco del *host* donde se alojan los contenedores (Nüst et al., 2020).

El uso de los contenedores permite crear, distribuir y colocar en producción aplicaciones de software de una forma sencilla, segura y reproducible. También a

3.5. ESTADO DEL ARTE:

cada contenedor se le puede realizar una asignación de recursos (memoria, cpu, almacenamiento) que garantice un óptimo funcionamiento de la aplicación que contienen.

Es importante señalar que el uso de esta tecnología añade un entorno de seguridad al estar cada contenedor en un ambiente aislado.

Para cada contenedor es necesario usar una imagen donde previamente se definen las dependencias (sistema operativo, librerías, lenguajes) necesarias para su funcionamiento.

3.4.2 Orquestador:

Al tener diversos contenedores, donde cada uno aloja una aplicación distinta, puede resultar necesario que todos se integren en un sistema. Para que esta integración sea viable es necesario contar con un orquestador (Cook, 2017). Su uso permitirá lograr altos grados de portabilidad y reproducibilidad, pudiendo colocarlos en la nube o en centros de datos, garantizando que se pueda hacer el *deploy* de forma sencilla y fiel a lo que se implementó en el ambiente de desarrollo.

En el caso de la Solución propuesta se adoptará el uso de *Docker Compose* como orquestador y en el Capítulo que contiene los Ciclos de Desarrollo 5.3.3 serán expuestas las funcionalidades de cada contenedor y se apreciará la integración que proporciona contar con un orquestador.

3.5 Estado del Arte:

Si bien anteriormente las búsquedas de información dentro de un corpus se procesaban determinando la aparición de palabras dentro de un texto, este método ha ido evolucionando para llegar hoy en día a un elevado nivel de abstracción, donde a partir de la necesidad de obtener una información, es decir, de aquello que necesitamos buscar, que antes consistía en hacer *match* con un objeto de información, se ha pasado de los motores de búsqueda (*search engines*) a los motores de respuestas (*answering engines*) (Balog, 2018), donde el sistema ante una determinada consulta del usuario, va a retornar una serie de resultados enriquecidos, mostrando la identificación de entidades, hechos y cualquier otro

dato estructurado que esté de forma explícita, e incluso implícita, mencionado dentro de los textos que conforman el corpus.

Para hablar sobre el Estado del Arte tanto en los Sistemas de Recuperación de Información 3.2 así como en el Procesamiento del Lenguaje Natural 3.3.1 y en la medición de similitud entre documentos 3.3.3 es necesario referir la representación de los textos mediante *embeddings*.

3.5.1 Embeddings

Para comprender qué son los *embeddings* se debe partir de estudiar la Hipótesis Distribucional, la cual se enmarca en al área de la lingüística y enuncia que la similaridad en significados, resulta en que también se presente una similaridad en la distribución lingüística. Dos palabras que sean próximas en significado, entendido como que sean intercambiables en un texto, es un fenómeno que también se detectará en la distribución que presentan dichas palabras dentro de un corpus. Más adelante se mostrará un ejemplo de esto.

De esta Hipótesis surge la propuesta de crear la “Distribución Semántica”, donde se representa el significado de una palabra, mediante el proceso en que se toma como entrada grandes cantidades de texto y se construye un modelo de distribución, también llamado “espacio semántico”, que logra extraer la representación semántica del vocabulario en un espacio n -dimensional, haciendo que una palabra se muestre como un vector en dicho espacio.

“Semántica” se entiende como el significado específico que puede tener una palabra en una oración. Al evaluar las siguientes dos frases:

1. “El modelo de banco de tres asientos está en oferta”
2. “Voy a depositar dinero al banco”

el significado de la palabra “banco” en los ejemplos tiene dos acepciones. Claramente el lingüista Firth J.R. enfrentó este problema en su famosa frase: “entenderás una palabra por aquellas que la acompañan”, donde hace entrever que para comprender el significado de una palabra hay que revisar el contexto en el que ocurre.

Al recordar 3.3.2.1 el modelo Term Document Frequency (TDF), para representar en un documento la aparición de una determinada palabra, se colocaba en la matriz el valor “1” en la posición i,j , o se colocaba un “0” en su ausencia,

3.5. ESTADO DEL ARTE:

correspondiendo la i al índice del documento y la j al índice de la palabra dentro del vocabulario; no obstante, en este método de representación no se puede lograr inferir la semántica de la palabra, sino simplemente la aparición, o no, dentro del texto, independientemente de la acepción que tenga la palabra “banco”, que siempre se representaría con un valor “1”, tanto en un documento que contenga “El modelo de banco de tres asientos está en oferta”, como en el otro “Voy a depositar dinero al banco”.

Lo anterior constituye un problema clave para los procesos de Recuperación de Información ya que si estuviésemos usando el modelo 3.2.3.2 de Índices Invertidos, que comparte algunos fundamentos del modelo 3.3.2.1 *Term Document Frequency (TDF)*, retomando el ejemplo en que usamos la palabra “banco”, al intentar encontrar aquellos documentos que mencionen a los “bancos” en su acepción de “Asiento, con respaldo o sin él...”, también se recuperaría el documento que habla de la “empresa que se dedica a realizar transacciones financieras”.

Este ejemplo, bastante trivial, plantea la necesidad de contar con modelos de representaciones con una estructura más compleja y que puedan facilitar mediante los métodos de *Information Retrieval*, la recuperación de los documentos que tengan la mayor relevancia 3.2.4 ante un *query* y que también se correspondan con lo que ciertamente se está buscando, como pudiera ser “fabrica de bancos para parques” y mejor aún sería si ante una búsqueda con el texto “fabrica de sillas”, también se recuperará el documento que indica “El modelo de banco de tres asientos está en oferta”, ya que asiento y silla pueden ocurrir en contextos semánticos similares.

Son los “*embeddings*” la representación que hoy constituye el Estado del Arte en el los procesos de Recuperación de Información ya que hacen posible aplicar distintos métodos algebraicos y computacionales para inspeccionar el vocabulario de un determinado corpus y tener nociones más precisas sobre la cercanía de una palabra con otra y hacer mediciones 3.3.3 de similitud entre un documento, que se ha transformado en partes o en su totalidad en un *embedding*, con otro que tenga el mismo tipo de representación vectorial.

En el siguiente ejemplo 3.1 que se obtiene del trabajo *Distributional Semantics and Linguistic Theory* (Boleda, 2020), se muestra una versión simplificada de un espacio semántico de dos dimensiones donde están los vectores que se corresponden con tres palabras que son “*postdoc*”, “*student*” y “*wealth*”:

Cuadro 3.1: Embedding bidimensional para representar palabras

Palabra	Dimensión.1	Dimensión.2
postdoc	0.71038	1.76058
estudent	0.43679	1.93841
wealth	1.77337	0.00012

Al tener cada palabra un vector de dos componentes, se puede hacer una representación gráfica en un plano, ver figura 3.3, donde al aplicar la medición de similitud coseno, revisada en 3.3.3, se determina que las palabras *postdoc* y *student* se encuentran más próximas y tienen una mayor similitud, que por ejemplo, *postdoc* y *wealth*.

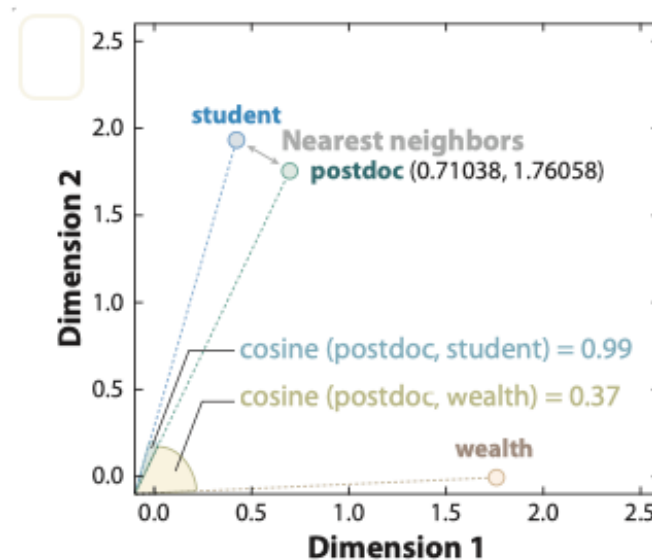


Figura 3.3: Representación de palabras en un plano

Al generar la representación completa de un espacio semántico, haciendo la búsqueda de una palabra, podemos encontrar también aquellas que son cercanas y no limitar la búsqueda al *match* que los modelos anteriormente estudiados sí imponían. Más adelante también veremos que el modelo semántico puede ser expandido y representar mediante un *embedding* oraciones (*sentences*), siendo esto el sustento de que al hacer una pregunta, o *query*, a un sistema de Recuperación de Información, este sea capaz de encontrar la respuesta dentro del corpus, ya que la pregunta o *query* se transforma en un *embedding* y luego se determina en el

3.5. ESTADO DEL ARTE:

espacio semántico cuál es la oración que más se aproxima, o guarda algún tipo de proximidad o relación de distancia vectorial con la pregunta formulada.

Los *embeddings* son representaciones numéricas densas de las palabras contenidas en un vocabulario. A diferencia de los modelos de tipo “one hot encoding (OHE)” de representación binaria, donde en un vector se usa un “1” para representar la aparición de una palabra dentro de un vocabulario y “0” para representar su ausencia, teniendo que la dimensionalidad del vector será la cantidad de n palabras que tenga el vocabulario. Es común que un corpus se puedan disponer de unas 20 mil o más palabras distintas, es decir un vocabulario con n igual a 20 mil, así sea para representar una sola palabra se requerirá en el OHE de unos 20 mil componentes con 19.999 ceros y un solo “1”, lo cual es una representación bastante dispersa que dificulta cálculos computacionales. Retornando al *embedding*, lo que ellos captan es una representación vectorial de las palabras pero con un número menor de componentes.

En algunas de la primeras representaciones realizadas, por ejemplo con el modelo “GloVe: Global Vectors for Word Representation” (Pennington et al., 2014), se tenían 100 componentes, cifra considerablemente menor a las 20 mil que del modelo de “one hot encoding”, evitando así la alta esparcidad. Otro cambio sustancial que se introdujo con este tipo de representación es que los componentes no son valores binarios de unos o ceros, sino se hace con números reales que pueden tener más de ocho decimales (floating point numbers). Bajo este modelo, la representación específica de la palabra “king” es el siguiente vector:

“0.50451, 0.68607, -0.59517, -0.022801, 0.60046, -0.13498, -0.08813, 0.47377, -0.61798, -0.31012, -0.076666, 1.493, -0.034189, -0.98173, 0.68229, 0.81722, -0.51874, -0.31503, -0.55809, 0.66421, 0.1961, -0.13495, -0.11476, -0.30344, 0.41177, -2.223, -1.0756, -1.0783, -0.34354, 0.33505, 1.9927, -0.04234, -0.64319, 0.71125, 0.49159, 0.16754, 0.34344, -0.25663, -0.8523, 0.1661, 0.40102, 1.1685, -1.0137, -0.21585, -0.15155, 0.78321, -0.91241, -1.6106, -0.64426, -0.51042”

el cual se puede representar graficamente como se observa en la figura 3.4 donde se mapean los números a una paleta de colores.

Usando el mismo modelo GloVe, una representación gráfica de las palabras “King”, “Man” y “Woman” en 50 dimensiones es la que se observa en la imagen 3.5.

El crédito a la visualización 3.5 (Alammar, 2019) corresponde al divulgador Jay

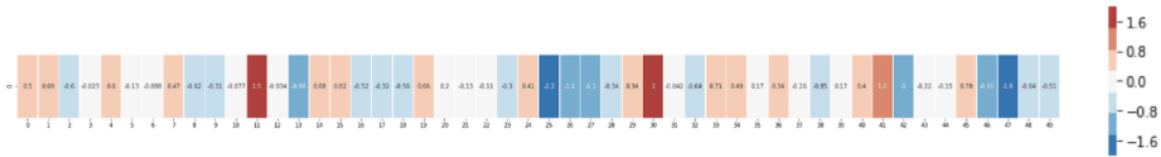


Figura 3.4: Representación de palabra "king" mediante el modelo GloVe



Figura 3.5: Representación de palabras mediante el modelo GloVe

Alammar. (2019). La representación muestra gráficamente que las palabras “man” y “woman” son más “parecidas” visualmente que “king” y “man” y esto es lo que se puede generalizar para entender como las distintas palabras que se representan en un espacio semántico pueden presentar proximidades, similitudes o diferencias entre si, obteniendo de esta forma un significado semántico según la posición relativa que cada una tenga con respecto a la otra en el espacio indicado.

No obstante, en los puntos expuestos aún no ha explicado cómo se generan los *embeddings*, lo cual es indispensable para entender sus capacidades. En el año 2003 mediante el entrenamiento de redes neuronales logran modelar la probabilidad de las secuencias de palabras demostrando la capacidad de las redes neuronales para capturar patrones complejos en datos textuales (Bengio et al., 2003) . Otro hito fue la investigación “Semantic Hashing” (Salakhutdinov and Hinton, 2009) donde usaron redes neuronales para transformar datos de alta dimensionalidad en representaciones binarias de baja dimensionalidad. Esa investigación añadió como aporte que se empezarán a usar técnicas de aprendizaje no supervisado para entrenar las redes neuronales, deshaciéndose de los cuellos de botella que previamente introducían los procesos de etiquetado, necesarios en métodos supervisados.

3.5. ESTADO DEL ARTE:

Ante la ampliación de capacidades de cómputo en sistemas distribuidos compuestos por tarjetas gráficas (Graphics Processing Unit - GPU), empieza a incrementarse el uso de redes neuronales de aprendizaje profundo y es cuando se presenta la investigación “Word2Vec: Efficient Estimation of Word Representations in Vector Space” (Mikolov et al., 2013a) que implementa la técnica *Skip-gram* y *Continuous Bag of Words (CBOW)* que permitieron a las redes neuronales el aprendizaje de representaciones semánticas de palabras a partir de grandes volúmenes de texto. Word2Vec no solo era eficiente computacionalmente, sino que también producía *embeddings* que capturaban relaciones semánticas y sintácticas, transformando cómo se abordaban las tareas de NLP y las aplicaciones de recuperación de información.

Una vez que se empezó a tener un método para capturar la semántica de las palabras debió seguir el paso de lograr representar el sentido semántico de frases (sentences) y expresiones más complejas. Esto se introdujo en la investigación “Distributed Representations of Words and Phrases and their Compositionality” (2013) (Mikolov et al., 2013b) donde se hicieron representaciones vectoriales distribuidas para frases, mejorando la capacidad de capturar significados contextuales y relaciones sintácticas en un nivel más alto, lo cual resultó crucial para mejorar los sistemas de recuperación de información y también para la traducción automática.

La investigación GloVe (Pennington et al., 2014) también implicó grandes avances ya que superó limitaciones que presentaban investigaciones anteriores, al permitir generar analogías del tipo: (vector de embedding para la palabra Rey) menos (vector de embedding para la palabra hombre) más (embedding para la palabra mujer) es igual, o muy aproximado en el espacio semántico, al (vector de embedding de la palabra reina) o simplificado como “rey-hombre+mujer=reina”.

Igualmente con esta investigación se intensificó el uso de este modelo en tareas de clasificación de texto como las revisadas en 3.3.1, ya que las redes neuronales empezaron a entrenarse con representaciones de vectores de gran densidad que contenían las palabras, el POS y el etiquetado de las dependencias conteniendo cada vector 200 componentes, alcanzando mejores indicadores de desempeño en el etiquetado (Chen and Manning, 2014a). Este trabajo fue el que dio soporte a la librería revisada anteriormente de nombre “coreNLP” de la Universidad de Stanford.

3.5.2 Arquitectura de Redes Neuronales *Transformers*:

En el año 2017 se publica “Attention Is All You Need” (Vaswani et al., 2017) el cual fue una investigación donde se introdujo una nueva arquitectura de redes neuronales que eliminó ciertas limitaciones que venían presentando los modelos de redes neuronales recurrentes y las convolucionales en poder trabajar con largas cadenas de texto. La solución introdujo los llamados “mecanismos de atención”⁶ que abrieron el camino para la creación de nuevos modelos de lenguaje como BERT (Devlin et al., 2018) que capturaban la riqueza de significados y las relaciones complejas del lenguaje mejorando la comprensión de textos, traducción automática y la generación de texto.

“RoBERTa: A Robustly Optimized BERT Pretraining Approach” (Liu et al., 2019) optimizó el entrenamiento preexistente de BERT al desvincular la tarea de pre-entrenamiento del tamaño de la cantidad de ejemplos de entrenamiento (*batch size*) y la duración del entrenamiento. Al escalar el tamaño del lote y la cantidad de datos, RoBERTa mejoró la comprensión del modelo sobre el lenguaje, logrando una capacidad de generalización excepcional. Estos métodos que venían innovando e incrementando las capacidades, por otra parte también hacían que el tamaño de los conjuntos de datos usados para el entrenamiento fuese creciendo exponencialmente, como se analizará en 3.5.3 la sección referida a los Largos Modelos del Lenguaje.

Otro modelo basado en la arquitectura de Transformers, que es necesario referir, ya que a un componente del SCSU le da soporte, es el que se publicó bajo el título “Sentence-BERT: Sentence Embeddings” (Reimers and Gurevych, 2019b) que a diferencia de los modelos anteriormente expuestos, que trabajaban con la codificación de palabras, en él se logra la codificación de oraciones usando una variante de BERT (Devlin et al., 2018). Al entrenar el modelo para entender la similitud semántica entre pares de oraciones, Sentence-BERT aprende representaciones de oraciones que capturan mejor las relaciones semánticas.

Un punto que fue necesario resolver, era lograr contar con representaciones de embeddings para distintos idiomas, ya que inicialmente estaban entrenados con textos en idioma inglés y uno de las investigaciones que permitió avanzar hacia modelos multilingües fue “Making Monolingual Sentence Embeddings

⁶Los mecanismos de atención permiten al modelo asignar ponderaciones dinámicas a diferentes partes de la entrada, lo que resulta en una comprensión más profunda y contextualizada del texto.

3.5. ESTADO DEL ARTE:

Multilingual” (Reimers and Gurevych, 2020), usando la técnica “Knowledge Distillation”, en lugar de entrenar modelos para cada idioma, este método utiliza un único modelo de referencia monolingüe para guiar el entrenamiento de modelos en múltiples idiomas, basándose en la idea de que “una frase traducida debe situarse en el mismo lugar del espacio vectorial que la frase original”.

Este recorrido por el desarrollo de los embeddings lleva finalmente al modelo “BETO: Spanish BERT” (Cañete et al., 2020), que bajo la arquitectura BERT fue entrenado por el Departamento de Ciencias de la Computación Universidad de Chile, disponible en el enlace <https://github.com/dccuchile/beto> . Este modelo para la fecha está considerado como el estado del arte para el idioma español, alcanzando una precisión del 98,97% en tareas como el POS 3.3.1.2.

Las investigaciones citadas se hicieron de dominio público y en muchos casos también se colocó a disposición de la comunidad científica los propios modelos preentrenados, lo que hizo que fuesen reproducibles tanto los modelos, como la evaluación de ellos.

Para obtener información sobre los modelos de *embeddings* que presentan una elevada precisión y son de alta demanda por la comunidad open source, siendo parte del estado del arte, se tienen herramientas como el “MTEB: Massive Text Embedding Benchmark” (Muennighoff et al., 2022) al que se puede acceder en el enlace <https://huggingface.co/spaces/mteb/leaderboard> donde muestra métricas de 140 modelos preentrenados en el área del lenguaje disponibles para el uso público.

3.5.3 Largos Modelos de Lenguaje:

Con la aparición de la arquitectura Transformers se abrió el camino para la aparición de los Largos Modelos de Lenguaje (*Large Language Models -LLM's*). En principio pudiese parecer estar fuera del alcance de este Trabajo de Grado exponer estos Modelos, pero el Estado del Arte de los Sistemas de Recuperación de Información se intersecta con ellos y no resultan ajenos a trabajos futuros que puedan suceder a esta investigación.

Según lo revisado en la sección de *embeddings* 3.5.1, la tendencia ha sido ir incrementando la cantidad de datos con que se entrenan estos modelos, igual que la cantidad de parámetros que conforman al propio modelo. En la figura 3.6 vemos las variaciones incrementales que se han dado desde la publicación del modelo basado en los Transformers en el 2017.

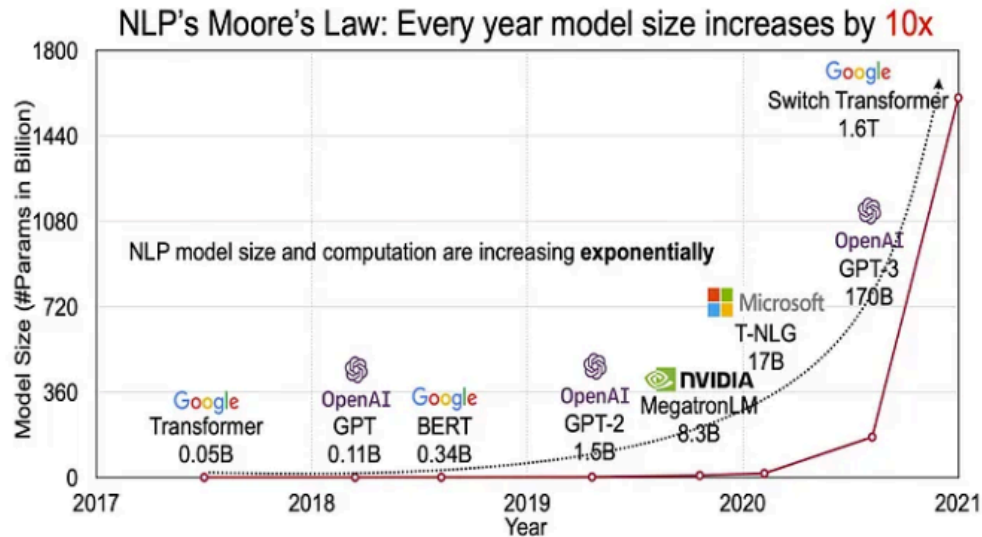


Figura 3.6: Evolución en la Cantidad de parámetros en los LLM

El crédito a la visualización 3.6 corresponde a Harishdatala (2023). Unveiling the Power of Large Language Models (LLMs). <https://medium.com/@harishdatablab/unveiling-the-power-of-large-language-models-llms-e235c4eba8a9> (acceso 23 el octubre, 2023).

Sin entrar en mayores consideraciones sobre este crecimiento y los costos asociados, que imposibilitan a instituciones educativas, empresas de mediano tamaño, investigadores independientes, poder acceder a los sistemas de computadores necesarios para entrenar modelos de estas características, a finales del año 2022 a uno de los modelos llamado “Generative Pre-trained Transformer 3” de la empresa OpenAI, del cual no se dispone mayor documentación sobre su arquitectura ni precisión sobre el método de entrenamiento, le es realizado un proceso de “fine tuning”, que es un ajuste a los parámetros mediante un reentrenamiento, y se creó lo que hoy se conoce comercialmente como ChatGPT 3.5, introduciendo mediante una interface de usuario, la capacidad de que un usuario pueda interactuar con el modelo simulando una conversación. Abstrayendo los procesos de entrenamiento, la magnitud de los parámetros y la innovación que representó crear el modelo de chat, lo que se tiene es un usuario interactuando con un modelo semántico de una magnitud gigante.

En general los Largos Modelos de Lenguaje son entrenados con enormes corpus de textos recopilados de foros de internet, de páginas web, de libros digitalizados y de un vasto cúmulo de textos. Si hacemos otra abstracción de un nivel más

3.5. ESTADO DEL ARTE:

alto, lo que se tiene es un usuario haciendo un *query* ante un enorme Corpus que excede y se organiza de una forma distinta a lo que habíamos revisado en los Sistemas de Recuperación de Información 3.2.1 clásicos donde se tenía una base de dato con documentos indexados. Ahora son distintos tanto el proceso de interacción usuario-computador y más importante aún es que también cambia la representación de la información, ya que cada vez que se coloca un *query*, este es transformado en un *embedding*, y mediante un proceso estocástico, el LLM va prediciendo la siguiente palabra, de una en una, y se van construyendo respuestas, que pueden llegar a ser fidedignas, o no tanto, dependiendo de la calidad del modelo y de las previsiones que se hayan tomado para mitigar sesgos o entradas de datos incorrectas en la fase de entrenamiento del modelo.

Como queda fuera del alcance de este trabajo explicar como funcionan los modelos del lenguaje, lo que sí se quiere indicar es que en el año 2023 algunas compañías e institutos de investigación privada, empezaron a liberar ciertos modelos, con distintos pesos y versiones, para la comunidad open source, como lo es el modelo Falcon (Penedo et al., 2023) o el modelo OpenLlama2 (Touvron et al., 2023)⁷. Con las facilidades para el desarrollo que aportan plataformas como huggingface.com para la implementación de aplicaciones de inteligencia artificial, mediante el almacenamiento de modelos preentrenados, conjuntos de datos para entrenamiento o sobre entrenamiento, así como librerías con *pipelines* de fácil integración mediante API's unificadas (Wolf et al., 2019), estos LLM's dejaron de tener un uso limitado sólo para grandes empresas o consorcios tecnológicos y para la fecha es viable que corran en computadoras con capacidades limitadas mediante métodos como la aplicación del quantized que se presenta en la investigación (Dettmers et al., 2023) que permite que un LLM que por ejemplo necesite unos 16 gb de memoria ram en GPU para ser desplegado, pueda disminuir una cuarta parte hasta los 4 gb.

La diversidad de modelos preentrenados, con distintas versiones de fine tuning o cuantizaciones, se puede ver en el enlace (Edward Beeching, 2023) https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard donde se encuentra un tablero que muestra los modelos que presentan mayor popularidad, descargas y métricas de evaluación de su comportamiento.

⁷la compañía que entrenó el modelo y lo liberó que es Meta indicó que es OpenSource, pero revisiones técnicas hechas a la licencia cuestionan que se pueda considerar que realmente cumpla las especificaciones para que sea considerado plenamente "open source". En el enlace Is Llama 2 open source? se encuentra un análisis sobre el tema.

3.5.4 Integración:

Las versiones *Open Source* de estos modelos es posible integrarla en procesos de *Information Retrieval* principalmente mediante tres técnicas.

1. **Retrieval Augmented Generation RAG** (Lewis et al., 2020): esta técnica permite que un LLM que fue entrenado con un determinado corpus, pueda activar la extracción de información desde fuentes externas, como páginas de internet, complementando la información que dispone el modelo. A nivel de interacción del usuario todo ocurre en el mismo entorno o API que dispone el modelo originalmente.
2. **Fine Tunning** (Lv et al., 2023): con un conjunto de datos etiquetado, de un volumen de datos mucho menor al que inicialmente fue entrenado un determinado LLM, se puede lograr que un modelo de lenguaje aprenda, sea sobreentrenado con métodos como el propuesto en “Universal Language Model Fine-tuning for Text Classification” (Howard and Ruder, 2018), con información de un dominio específico, mejorando su desempeño en esa particular área.
3. **Vector DataBase:** mediante una representación de datos en *embeddings* se crea una base de datos con los documentos que están contenidos en corpus. El manejador de base de datos ofrece un almacenamiento optimizado y capacidades de consulta para estructuras únicas de *embeddings* vectoriales, permitiendo hacer búsquedas semánticas, alto rendimiento, escalabilidad y recuperación de datos al comparar valores y encontrar similitudes.

Principalmente estas tres técnicas, por separado, o en paralelo, pueden implementarse para crear sistemas de recuperación de información que se adapten y sean expertos en áreas de estudio de la Universidad Central de Venezuela, modificando la forma en que anteriormente un investigador hacía la búsqueda de información.

3.5.5 Búsqueda Semántica:

Es una evolución en la forma en que los motores de búsqueda comprenden y responden a las consultas. A diferencia de la búsqueda tradicional basada en

3.5. ESTADO DEL ARTE:

palabras clave, que se centra en encontrar coincidencias literales entre la consulta y el contenido web, la búsqueda semántica tiene como objetivo comprender el significado contextual de las consultas y ofrecer resultados más relevantes y precisos.

En lugar de simplemente emparejar palabras clave, la búsqueda semántica utiliza *embeddings* para representar el texto y de una mejor forma comprender la intención detrás de una consulta. Esto implica analizar la relación semántica entre las palabras, interpretar el contexto y comprender el significado subyacente. Así, la búsqueda semántica busca proporcionar resultados que no solo coincidan con las palabras clave, sino que también aborden la verdadera intención del usuario.

Se apoya en representar cadenas de texto de un documento como vectores *embeddings* que son almacenados en una base de dato y dado un *query* se transforma el texto en otro *embedding* y se determina cuáles son los embeddings de la base de dato que presentan mayor similitud (Muennighoff, 2022).

Finalmente se mencionan algunos puntos de lo que hoy constituye el estado del arte en temas que hemos ido revisando a lo largo de este capítulo:

1. Con modelos como BERT se ha hecho implementaciones modificadas para hacer el re ordenamiento 3.2.5 de los resultados obtenidos en un proceso de búsqueda, bien sea mediante las técnicas tradicionales o mediante técnicas de búsqueda semántica(Nogueira and Cho, 2019) . Igualmente con redes neuronales se ha buscado simular el comportamiento humano para jerarquizar los resultados obtenidos en procesos de búsqueda (Pang et al., 2017).
2. Se están proponiendo nuevos métodos para evaluar la eficacia en los sistemas de “preguntas-respuestas” basados en similaridad semántica dadas las limitaciones que presentan las métricas tradicionales que no reflejan el desempeño de estos nuevos modelos (Risch et al., 2021).
3. Mediante técnicas de aprendizaje profundo se están creando conjuntos de datos sintéticos de dominio público que permitan evaluar el desempeño de los sistemas de recuperación de información, usando como entrada las publicaciones de Wikipedia y creando con estos modelos los *querys*, que permitan medir el grado de precisión que alcanza un determinado sistema (Frej et al., 2020).

Así culmina el recorrido por lo que es el Marco Teórico-Referencial 3 que soporta la investigación “Recuperación, Extracción y Clasificación de Información de SABER UCV”.

Capítulo 4

Capítulo Marco Metodológico:

En este capítulo, se presenta el enfoque metodológico adoptado para este estudio. La metodología Kanban 4.1 se empleó para gestionar el proceso general, mientras que la metodología de Desarrollo Adaptable de Software 4.2 guió la creación del software, permitiendo una implementación eficiente y adaptable a las necesidades cambiantes del proyecto **Recuperación, Extracción y Clasificación de Información de SABER UCV**.

4.1 Metodología de Trabajo Kanban:

En esta metodología se fomenta la cultura de mejora continua, incremental, al identificar cuellos de botella y la limitación del trabajo en curso para aumentar la eficiencia y la productividad. Gracias a su enfoque basado en la colaboración, flexibilidad y respuesta rápida a los cambios, puede considerarse que está dentro de las metodologías “Ágiles”.

Mediante la visualización del flujo de trabajo en un tablero, que se aprecia en la figura 4.1, se puede facilitar la gestión del proyecto, desde la concepción de una idea, hasta su implementación y entrega. En el caso del desarrollo de software, facilita la toma de decisiones informadas y promueve la transparencia al proporcionar una visualización clara de las tareas y actividades involucradas (Stephens, 2015).

Durante el desarrollo de este Sistema se necesitaba contar con la flexibilidad que ofrece esta metodología, ya que en ella no se tienen que definir roles específicos en el equipo desarrollador, ni tampoco es necesario establecer períodos fijos para

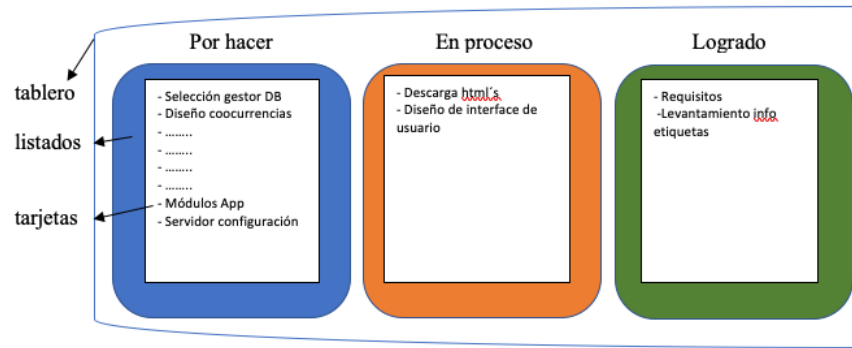


Figura 4.1: Representación de un tablero según la metodología Kanban

alguna fase en particular.

4.2 Desarrollo Adaptable de Software:

El **Adaptive Software Development (ASD)** (Highsmith, 2000) es una metodología ágil de desarrollo de software que se centra en la adaptabilidad y capacidad para adaptarse a los cambios, proporcionando una retroalimentación temprana y frecuente, dando flexibilidad para abordar los desafíos cambiantes del desarrollo de software. A diferencia de los enfoques tradicionales, el ASD reconoce la naturaleza impredecible del desarrollo de software y se adapta continuamente para satisfacer las necesidades del cliente en un entorno dinámico y complejo, haciendo énfasis en el principio “Entregar el proyecto que se necesita al final, no el proyecto que se pidió al principio”.

4.2.1 Características:

Colaboración y Comunicación Constante: fomenta la colaboración cercana entre los equipos de desarrollo y los *stakeholders*. La comunicación constante permite una comprensión profunda de los requisitos del cliente y facilita ajustes rápidos según las necesidades cambiantes.

Iteraciones Incrementales: divide el proyecto en iteraciones cortas y manejables. Cada iteración produce un incremento funcional del software, lo que permite obtener retroalimentación temprana que permite corregir errores y ajustar el rumbo del proyecto antes de que los problemas se vuelvan críticos.

4.2. DESARROLLO ADAPTABLE DE SOFTWARE:

Flexibilidad y Adaptabilidad: reconoce que los requisitos del proyecto pueden cambiar con el tiempo. Por lo tanto, se adapta fácilmente a los cambios, permitiendo una rápida reevaluación y ajuste de las estrategias y metas del proyecto asegurando que el producto final esté alineado de manera óptima con las necesidades y expectativas del cliente, incluso en un entorno de desarrollo volátil.

4.2.2 Ciclos:

El desarrollo adaptable de software se basa en un proceso dinámico e iterativo donde cada ciclo contiene las siguientes fases: Especular-Colaborar-Aprender. El proceso se enfoca en el aprendizaje continuo y la colaboración intensiva entre desarrolladores y clientes, fundamental para enfrentar las cambiantes dinámicas empresariales. Es relevante destacar que, en algunas circunstancias, los ciclos pueden avanzar simultáneamente en ciertas iteraciones, permitiendo así una optimización del tiempo y recursos.

En cada ciclo, se pueden realizar múltiples iteraciones con el objetivo de desarrollar exhaustivamente todos los requisitos contemplados.

Estos ciclos representan un enfoque metodológico que asegura la coherencia y calidad del desarrollo del sistema. A través de la especulación, la colaboración y el aprendizaje continuo, se logra un refinamiento progresivo de las funcionalidades del sistema, garantizando así su robustez y adaptabilidad a las demandas del entorno. Este enfoque iterativo y colaborativo constituye una práctica fundamental en el proceso de desarrollo, facilitando la identificación temprana de posibles desafíos y fomentando la innovación constante en cada etapa del ciclo.

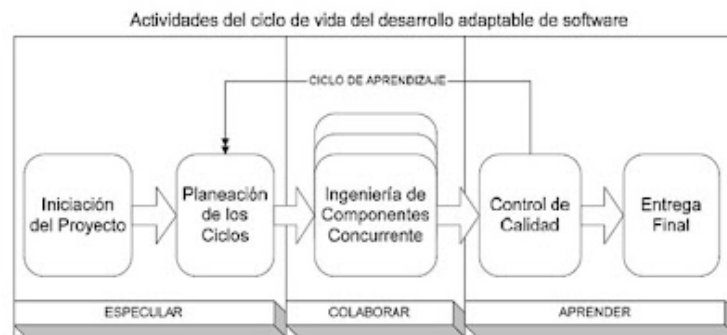


Figura 4.2: Ciclo ASD

4.2.2.1 Especulación:

Este componente ofrece un espacio para la exploración y la comprensión de la incertidumbre. Permite desviarse del plan inicial sin temor, transformando los errores en oportunidades de aprendizaje. Aceptar que no se sabe todo impulsa la disposición para aprender y experimentar.

4.2.2.2 Colaboración:

Las aplicaciones complejas requieren la recopilación y el análisis de grandes volúmenes de información y ejecución de tareas. Este proceso es inmanejable para un individuo. En entornos dinámicos, donde fluyen grandes cantidades de datos, es esencial la colaboración. Un solo individuo o un pequeño grupo no puede abarcar todo el conocimiento necesario.

4.2.2.3 Aprendizaje:

La evaluación continua del conocimiento a través de retroalimentaciones y reuniones grupales al final de cada ciclo iterativo es esencial. Este enfoque difiere de la evaluación al final del proyecto. Evaluar constantemente permite enfrentar y resolver de manera efectiva los cambios constantes del proyecto y su adaptación.

Capítulo 5

Desarrollo de la Solución:

En este Capítulo en 5.1 se presenta la **Descripción General de la Solución**. Posteriormente se muestra la 5.2 **Arquitectura de la Solución** con el “Modelo-Vista-Controlado”. En 5.3 se exponen los cuatro **Ciclos de Desarrollo** que se efectuaron, mientras que en 5.4 **Pruebas** se encuentran las distintas pruebas que fueron ejecutadas para medir el comportamiento y rendimiento del software desarrollado.

5.1 Descripción General de la Solución:

Se implementa un Sistema de Recuperación de Información sobre un corpus de documentos de tesis de grado y trabajos de grado que originalmente se encuentran alojados en el repositorio digital Saber UCV . Utilizando técnicas de extracción de datos de archivos HTML, desde la ficha de cada investigación, se obtiene: el título, el nombre del autor, palabras clave, fecha de publicación, el resumen y el url de descarga del documento que sustenta la investigación.

Posteriormente el Sistema descarga el documento referenciado en cada ficha, el cual contiene el texto completo de la investigación, da lectura y clasifica información sobre el nombre de la facultad, la escuela o postgrado donde fue realizado el trabajo e igualmente extrae el nombre del tutor.

Todos los datos obtenidos son sometidos a técnicas del estado del arte en el Procesamiento del Lenguaje Natural y la Minería de Texto para conformar un corpus anotado, un índice invertido y una tabla con los vectores de *embeddings* (vector database), esenciales para un eficiente manejo de la base de datos.

El Sistema incluye una aplicación web que permite a los usuarios desde un

navegador explorar extensivamente el corpus anotado, realizando consultas de texto y aplicando varios filtros como la selección de la jerarquía, el área académica y el rango de fechas.

La relevancia de los resultados recuperados se determina mediante una función de ponderación y los documentos se presentan de manera priorizada para mejorar la experiencia del usuario.

Adicionalmente, el Sistema ofrece recomendaciones de documentos que presentan similitud con aquellos que fueron recuperados en el proceso anterior y muestra “Mapas de Conocimiento” mediante una herramienta gráfica interactiva de visualización.

La solución implementada cuenta con procesos automatizados de actualización para incorporar las nuevas investigaciones que sean añadidas al repositorio Saber UCV.

El Sistema se soporta en un sistema distribuido conformado por contenedores que son gestionados por un orquestador con la arquitectura “modelo-vista-controlador”.

5.2 Arquitectura de la Solución:

La arquitectura “Modelo-Vista-Controlador” se muestra en la figura 5.1 y posteriormente se describe el comportamiento y las interacciones de los componentes.

5.2.1 Modelo:

El *Modelo*, en el contexto de esta propuesta, es la parte del Sistema que se ocupa de la obtención, manipulación y gestión de los datos. Esto incluye la descarga de la información, la clasificación, el Procesamiento del Lenguaje Natural, la Minería de Texto y la creación del índice invertido. Esta parte del Sistema también incluye la lógica para generar los *embeddings* y manejar el corpus anotado. En la implementación el manejador de base de datos que usa es PostgreSQL.

Igualmente le corresponde realizar las tareas de actualizar el corpus periódicamente y las recomendaciones de documentos a medida que se agreguen nuevos textos en Saber UCV. En 5.3.3 el tercer ciclo de desarrollo se expondrán con detalle los

5.2. ARQUITECTURA DE LA SOLUCIÓN:

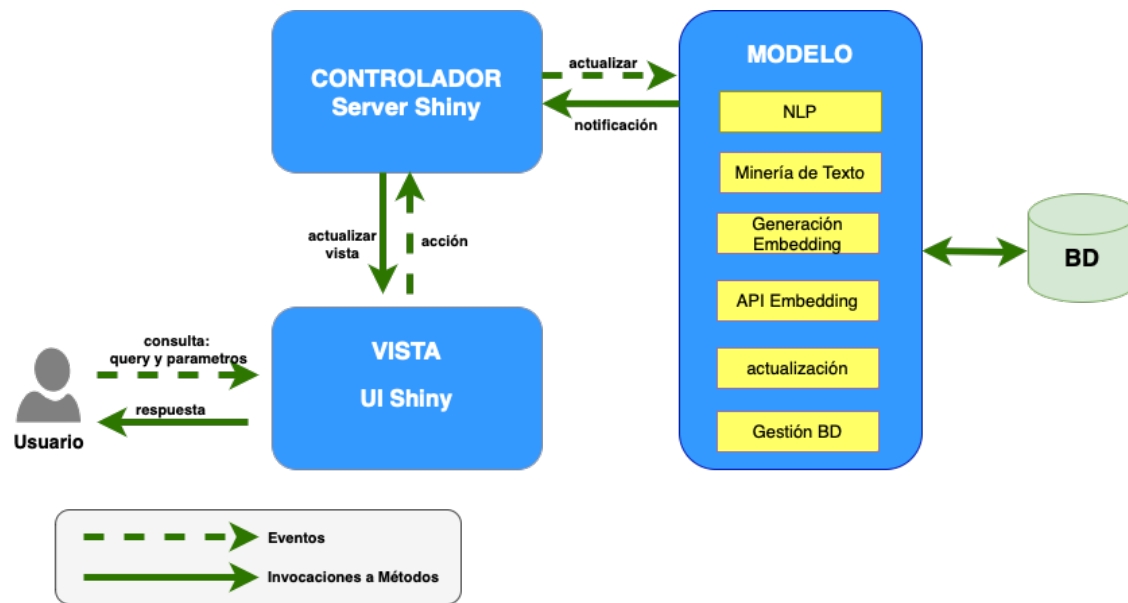


Figura 5.1: Modelo de Arquitectura MVC

componentes del *Modelo* y a continuación se enumeran los principales procesos que él sustenta:

1. Creación del Índice Invertido:

- Organizar los términos y sus ubicaciones en los documentos para permitir búsquedas eficientes.
- Asociar cada término con la lista de documentos en los que aparece.

2. Procesamiento de Texto:

- Tokenización: Dividir el texto en palabras o frases significativas.
- Lematización: Reducir las palabras a su forma base para un análisis más preciso.
- POS: etiquetado de partes del discurso.

3. Minería de Texto:

- Análisis de Frecuencia: Determinar la frecuencia de ocurrencia de palabras o frases.

4. Generación de Embeddings:

- Utilización de un modelo preentrenado para convertir palabras o frases en vectores numéricos.
- Convertir en vectores las palabras que componen el *query* para realizar comparaciones semánticas y determinar similitudes entre palabras o documentos.

5. Gestión de la Base de Datos:

- Almacenar y recuperar datos estructurados para su posterior consulta.
- Actualizar el corpus con nuevos datos.

6. Cálculo de Relevancia:

- Aplicar algoritmos para calcular la relevancia de los documentos en función de las consultas del usuario.
- Ordenar los resultados en función de su relevancia para presentar los documentos más relevantes primero.

7. Actualización de datos:

- Los procesos mencionados previamente son ejecutados y/o actualizados periódicamente.
- Se realiza la validación de la integridad de datos para asegurar que los nuevos datos se integren correctamente sin errores o inconsistencias, eliminando posibles duplicados o valores incorrectos.

5.2.2 Vista:

La *Vista* se implementa mediante el *framework* “Shiny” (Chang et al., 2023a) ¹ que permite crear aplicaciones web interactivas y tiene un componente de “User Interface (UI)” donde el usuario interactúa al introducir el texto con el que se hará la búsqueda y aplica filtros como jerarquía, área académica y rango de fechas. Posterior a la definición de los atributos del *query*, se desencadenan acciones que son enviadas al *Controlador* y al recibir la respuesta la *Vista* se actualiza y muestra las tablas con los resultados de la búsqueda, las representaciones visuales como el “Mapa del Conocimiento” y las recomendaciones de documentos similares.

¹El *framework* Shiny incluye dos componentes principales. El primero es la UI (interfaz de Usuario), que corresponde a la “Vista”. El otro componente es el “Server” que en la representación actual es el *Controlador*.

5.3. CICLOS DE DESARROLLO:

5.2.3 Controlador:

El *Controlador* se implementa mediante el *framework* “Shiny” que tiene un componente denominado “Server” que es el responsable de manejar las interacciones del usuario, gestionar las consultas de texto y aplicar los filtros seleccionados. También se encarga de orquestar las operaciones entre el *Modelo* y la *Vista*, asegurando que los datos se presenten correctamente y que las consultas se procesen de manera eficiente. Desde el *Controlador* se hace el llamado al componente del *Modelo* donde se encuentra la API para generar el *embedding* del *query* y determinar la relevancia de los documentos recuperados. El *Controlador* aplica el re ordenamiento para mostrar en orden los resultados más relevantes. En él se conforma la estructura para representar el “Mapa del Conocimiento” con datos obtenidos del *Modelo*.

5.3 Ciclos de Desarrollo:

Los Ciclos de Desarrollo 4 constituyen fases críticas del proceso, donde se conciben, diseñan y perfeccionan las funcionalidades del Sistema. Cada uno de estos ciclos está estructurado en tres etapas fundamentales: la etapa de especulación, donde se plantean las ideas y se exploran posibles soluciones; la etapa de colaboración, donde se trabaja en equipo para implementar estas ideas y se evalúan los resultados; y la etapa de aprendizaje, donde se analizan las experiencias pasadas y se ajustan las estrategias para futuras iteraciones.

Para el desarrollo del SCSU se hizo un proceso iterativo donde en cada ciclo se abordó cada una de las fases descritas y así incrementalmente se fueron añadiendo funcionalidades al Sistema.

La literatura en este tema siempre especifica a un cliente del que hay que obtener retroalimentación temprana, para así adaptar el producto a medida que evoluciona. Esto fue lo que se hizo en reuniones continuas en la materia *Tópicos Especiales en Sistemas de Información y Gerencia* que representó a la unidad requirente (cliente) y así se fueron evaluando los requisitos y se formularon las correspondientes hipótesis, se observó y se midió el desempeño, por ejemplo, en los modelos de aprendizaje automático preentrenados usados para los procesamiento de los textos.

Los Ciclos que se van a exponer son los siguientes: en 5.3.1 se muestran las tres iteraciones realizadas para la **Conformación del Conjunto de Datos**. En 5.3.2 se revisan las iteraciones para el desarrollo del **Prototipo del SCSU**, mientras que en 5.3.3 se hace la **Integración de los Componentes del Software** y en 5.3.5 se hace una versión del Sistema que incluye un **Buscador Semántico**.

5.3. CICLOS DE DESARROLLO:

5.3.1 Ciclo - Conformación del Conjunto de Datos:

En este ciclo es donde se ejecutaron las tareas que permitieron conformar el conjunto de datos acorde a lo planteado en el 2.4.2 **Objetivo Específico 1**, siendo el insumo con el que se desarrollará el **Sistema Complementario Saber UCV**.

Se realizaron tres iteraciones para lograr el objetivo. La primera 5.3.1.1 fue la **Extracción de Datos web**, también conocidas como “web scraping”, la segunda iteración 5.3.1.2 correspondió al **Levantamiento de las Categorías**, que son los nombres de las carreras de pregrado y de los postgrados que se imparten en la Universidad Central de Venezuela, mientras que en la tercera 5.3.1.3 iteración se hizo la **Clasificación de los Trabajos** asociando a cada investigación el nombre de la carrera o del postgrado e igualmente se hizo la extracción del nombre del tutor, acorde a lo planteado en el 2.4.2 **Objetivo Específico 2**.

5.3.1.1 Iteración- “Extracción de Datos web Saber UCV”:

5.3.1.1.1 Especulación: El repositorio Saber UCV en la sección “Comunidades/Tesis” aloja las cantidades de trabajos por nivel académico que se muestran en el cuadro 5.1 :

Cuadro 5.1: Cantidades de Trabajos por Categoría

Pregrado	Otras	Maestría	Doctorado
8.305	1.477	743	318
cifras de Saber.UCV a la fecha 15/10/2023			

En la minería de datos con la extracción de datos web es posible “recolectar, procesar, analizar y extraer útiles conocimientos a partir de los datos disponibles” (Aggarwal, 2018b). Por esto se planteó replicar en un conjunto de datos alojado localmente, la información contenida en el repositorio Saber UCV que asciende a una cantidad de 10.843 investigaciones, incluyendo: categoría (pregrado, otros, maestría, doctorado), el título, autor, fecha de publicación, palabras clave, *url* de

descarga ² y el texto del resumen. Al obtener esta información se puede dar inicio a la conformación del corpus.

5.3.1.1.2 Colaboración: En esta etapa se realizaron dos procesos de extracción de datos web usando el lenguaje de programación R version 4.3.2 (2023-10-31) (R Core Team, 2023).

1. El primero fue encontrar los *url*'s de cada trabajo alojado en el repositorio Saber UCV. Usando usando la extensión *SelectorGadget* (<https://selectorgadget.com/>) del navegador *Google Chrome* se puede obtener mediante un clic en un elemento de la página la etiqueta *css* asociada al nodo que se desea extraer. En este caso al visitar la página “<http://saber.ucv.ve/handle/10872/1957/browse?type=dateissued>” se identificó la etiqueta *'evenRowOddCol'*, ver figura 5.2, que identifica a los nodos dentro de la página que tienen los enlaces *href* a las fichas de cada investigación.

Posteriormente con el paquete *rvest* (Wickham, 2022a) que permite la descarga de páginas web y la manipulación de nodos XML se pudieron extraer los 10.843 *urls* a visitar.



Figura 5.2: Etiquetas nodos *url*'s

2. En una segunda fase, mediante la misma técnica indicada en el punto anterior, se localizó en la ficha de un trabajo la etiqueta *css*, en este caso la

²este *url* corresponde a el documento escrito del trabajo de grado o tesis que se encuentra alojado en word o pdf.

5.3. CICLOS DE DESARROLLO:

'metadataFieldValue' , que está asociada al nodo que contiene los valores del: título, autor, fecha de publicación, palabras clave, url de descarga del documento y el texto del resumen. En la figura 5.2-b se aprecia una imagen de una ficha. Al contar con el listado de url's y la identificación de los datos a extraer, se hizo un bucle para visitar cada enlace, se descargó el archivo html, se accedió al nodo, se extrajeron los valores y se fueron almacenando en una estructura de datos.

5.3.1.1.3 Aprender: Se enfrentaron las siguientes dificultades y se adoptaron en algunos casos las correspondientes soluciones:

1. Se realizaron varios intentos para la descarga y extracción de los valores. Para la obtención de cada campo, en principio se tomó de referencia la posición fija en que aparecía dentro de la ficha, porque se había asumido que estas tenían la misma estructura para todos los trabajos, sin embargo en algunas aparecían otros valores, p. ej. el de “colección”, alterando la posición en que se encuentra el dato a extraer. La solución adoptada fue que primero se localizarán, dentro de cada ficha, los títulos de los campos, con esto se generó el listado de los valores posicionales y relativo a estos se extrajeron los valores propuestos.
2. Algunos valores de las fechas contenían información parcial faltando el mes y/o el día. Se adoptó un método de imputar el valor “1” tanto al mes como día faltante.
3. Adoptar previsiones para caídas del servidor de Saber UCV y resguardar en cada vuelta del bucle la información extraída, para no perder el trabajo de extracción acumulado en caso de una falla remota o local en el acceso.
4. La revisión del conjunto de datos obtenido mostró que existen 861 valores duplicados en el “título” de los trabajos. Esta situación también se presenta con un subconjunto distinto al anterior, donde existen “resúmenes” que están repetidos. Esto ocurre por la introducción de algún carácter adicional o mínimas alteraciones en el texto. Para descartar estos registros, se aplicó una función de limpieza al texto (convertir a minúscula, remover signos puntuación, etc.). Posteriormente se obtuvieron sendos valores *hash* sobre el título y el resumen y luego se descartaron los *hashes* duplicados.

Adicionalmente se decidió usar el *hash* obtenido del “título procesado” como el identificador único de cada documento. La remoción de investigaciones que puedan estar duplicadas es importante efectuarla, ya que al ejecutar los procesos de recuperación de información o la representación de los resultados en Mapas de Conocimiento, al incluir textos repetidos, creará representaciones distorsionadas. En el cuadro 5.2 se muestra la cantidad de los valores detectados por Jerarquía.

Cuadro 5.2: Cantidades de Trabajos Duplicados

Jerarquía	Disponibles	Únicos	Duplicados
pregrado	8.305	7.634	671
otras	1.477	1.343	134
maestría	743	695	48
doctorado	318	310	8
cifras de Saber.UCV a la fecha 15/10/2023			

5.3.1.2 Iteración- Levantamiento de Categorías:

5.3.1.2.1 Especulación: Para poder clasificar cada investigación es necesario contar con las categorías que serán asignadas. Se entiende por “categoría” el nombre de la carrera de pregrado o el postgrado, junto con la facultad, que constituyen la oferta de la Universidad Central de Venezuela en educación universitaria.

Al no encontrarse el listado de categorías disponible en el propio repositorio Saber UCV fue necesario realizar una búsqueda web de esta información, extraerla y estructurarla, para así contar con el conjunto de datos de categorías que permita ejecutar la siguiente iteración, que es la de **Extracción y Clasificación de las Investigaciones** 5.3.1.3.

5.3. CICLOS DE DESARROLLO:

5.3.1.2.2 Colaboración: Se visitó al sitio oficial de la Universidad Central de Venezuela para revisar la oferta de pregrados y postgrados. Para los postgrados se encontró para cada categoría (especialización, maestría y doctorado) una página con el listado, p. ej. <http://www.ucv.ve/organizacion/maestria.html>³. En la figura 5.3 se aprecian las potenciales etiquetas para las maestrías.



Figura 5.3: Listado de Maestrías

Mediante la técnica de recuperación de datos web la información descrita en 5.3.1.1 se procedió a extraer el nombre de cada postgrado, añadir el nivel académico y asociar la facultad a la cual está adscrito. La cantidad de postgrados por nivel académico: Doctorado, Maetría y Especializaciones se muestran en el cuadro 5.3.

Cuadro 5.3: Cantidades de Postgrados por Categoría

Doctorado	Maestría	Especialización
46	101	228

En cuanto a los pregrados no se encontró en el sitio de la Universidad en una página centralizada la información y se procedió a obtenerla de la página wikipedia asociada a la U.C.V. recuperando un total del 50 nombres de escuelas de pregrado junto con la facultad de dependencia.

³previniendo posibles modificaciones en las páginas que contienen los listado de postgrados, se procedió a respaldarlas y forman parte del contenido del repositorio asociado a esta Investigación para garantizar la reproducibilidad de los resultados obtenidos. Para la fecha de redacción de este documento el contenido de los *urls* indicados fue modificado

En la figura 5.4 se muestran los totales apilados de pregrados y postgrados que se encontraron por Facultad-Centro de Investigación durante el levantamiento de información, cifra que al totalizar los postgrados y pregrados alcanza la cantidad de 425 áreas de conocimiento.

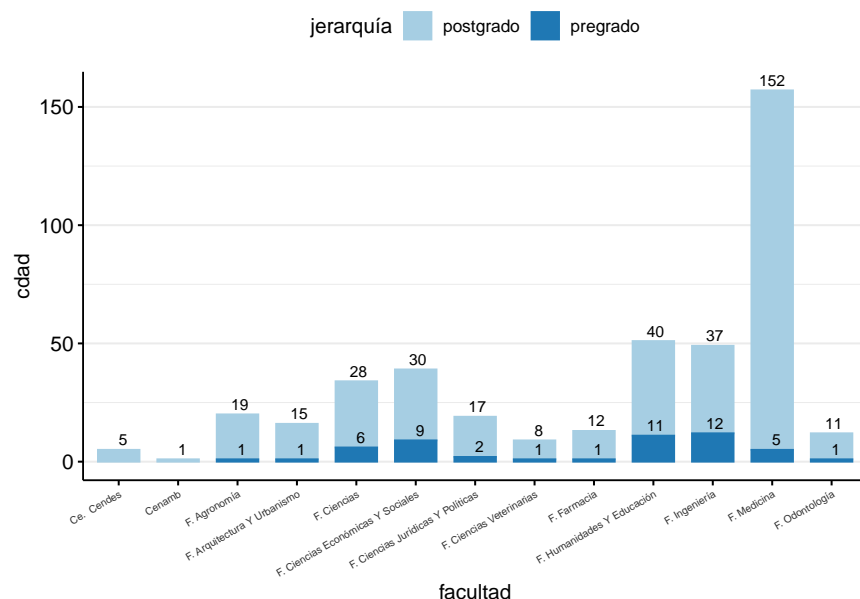


Figura 5.4: Total pregrados y postgrados por Facultad-Centro

5.3.1.2.3 Aprender:

1. En el caso de los postgrados, inicialmente se esperaba que sólo estuvieran asociados a Facultades pero también se encontró que el Centro de Estudios del Desarrollo y el Centro de Estudios Integrales del Ambiente imparten este tipo de estudios.
2. Se evaluó que existen nombres postgrados duplicados con la misma categoría, lo que puede generar problemas en la clasificación de las investigaciones teniendo como ejemplo la “Maestría en Estadística” que se dicta en la Facultad de Agronomía y en la Facultad de Ciencias Económicas y Sociales.
3. Se detectó que en pregrado existen escuelas que otorgan distintos títulos, p. ej. de la “Escuela de Administración y Contaduría” se pueden obtener los títulos de “contador” o de “licenciado en administración”. Esto es algo a tener presente al momento de hacer la categorización, ya que el nombre de

5.3. CICLOS DE DESARROLLO:

la escuela no sirve en estos casos para realizarla, siendo necesario agregar al conjunto de datos el atributo del nombre de los títulos emitidos.

4. Sobre los textos se tuvieron que realizar modificaciones, ya que en los listados se encontró que algunos nombres les faltaban palabras. La revisión final de nombres, dada la cantidad total de 425 dependencias, se hizo manualmente para evitar posteriores categorizaciones erróneas .

5.3.1.3 Iteración- Extracción y Clasificación de las Investigaciones:

En esta iteración se mencionan los principales obstáculos y las estrategias implementadas para alcanzar el objetivo de realizar la clasificación de cada trabajo alojado en Saber UCV, no obstante se omite especificar algunos de los problemas que se encontraron, ya que extenderse en esto abultaría considerablemente el contenido expuesto.

5.3.1.3.1 Especulación: Para las investigaciones que reposan en Saber UCV que cuentan con un archivo anexo, correspondiente al documento de la misma, es posible realizar la descarga, extraer una porción de texto y adoptando métodos basados en reglas de coincidencia de patrones, con las etiquetas obtenidas en la iteración 5.3.1.2 **Levantamiento de Categorías**, hacer la categorización por área de estudio, asignando el nombre del pre o postgrado, la escuela-postgrado y la facultad-centro de adscripción. Igualmente de esta porción de texto se estima viable extraer el nombre del tutor.

5.3.1.3.2 Colaboración: Motivado a que en la primera iteración 5.3.1.1 para conformar el conjunto de datos se había obtenido el *url* asociado al documento soporte de la investigación, se procedió mediante un bucle a realizar la descarga de cada documento y extraer una cantidad de dos mil caracteres, partiendo del principio de que los trabajos de grado o tesis en sus primeras páginas tienen el nombre de la carrera o el postgrado, el nombre de la facultad-centro donde se cursó el estudio, el nombre del título al que optan y el nombre del tutor.

En esta iteración fue necesario realizar distintas adaptaciones para lograr la coincidencia de patrones. Teniendo en cuenta que son 425 etiquetas las que se usarán para realizar la clasificación, llegando a tener 14 palabras algunas

categorías, es elevada la probabilidad de que no se pueda hacer el “*pattern matching*” entre el texto y la etiqueta.

Lo anterior motivo a realizar un proceso de limpieza, modificación y disminución de la cantidad de palabras, tanto en las etiquetas como en el texto extraído. Se evaluó en cada adaptación cuáles razones impedían clasificar los documentos aún pendientes, se tomaron los correctivos y así se fue incrementando, de forma iterativa, la precisión en este proceso.

También se tuvo que tomar en cuenta el orden en que se iba a ejecutar la secuencia de encontrar las coincidencias. Ejemplo es que varias facultades contienen las mismas tres palabras en la parte inicial de su nombre: *Facultad de Ciencias*, *Facultad de Ciencias* Jurídicas y Políticas, *Facultad de Ciencias* Económicas y Sociales y la *Facultad de Ciencias* Veterinarias. La secuencia para hacer la detección de la coincidencia fue buscar en orden decreciente por el total de caracteres que tenga el nombre de la facultad, para evitar clasificaciones erróneas.

Adicionalmente en el proceso de hacer coincidir las frases, se encontraron 17 postgrados que no estaban en el listado previamente conformado, los cuales se tuvieron que agregar al conjunto de datos de las categorías.

Para aquellos casos donde no se podía hacer *match* se aplicó el algoritmo “Smith Waterman” (Smith and Waterman, 1981), expuesto en el Capítulo del Marco Teórico 3.1, el cual permite alinear dos cadenas de texto cuando una de ellas no tiene coincidencia absoluta con la otra, como puede pasar en este caso por la introducción de caracteres adicionales.

Un elemento que introdujo ruido en el texto leído de los documentos, fue la aparición de diversos *encodings* que no resultó viable codificarlos a “UTF-08”, haciendo que aparecieran caracteres no reconocidos dentro del texto, dificultando la tarea de lograr realizar el proceso de “*pattern matching*”. El algoritmo SW resultó eficaz para solucionar este problema, aunque igualmente se hicieron pruebas con otros métodos como el algoritmo de “Distancia de Levenshtein” o similares.

Sobre un total de 9.982 potenciales documentos se lograron clasificar 9.585 investigaciones, mientras que 244 no disponían información en el texto del documento y resultaba inviable hacer la categorización. En algunos casos esta falta de información estuvo motivada en que el archivo contenía imágenes por estar escaneado el contenido o los documentos anexos no eran trabajos de grado o tesis sino informes de algún otro estilo.

5.3. CICLOS DE DESARROLLO:

La cantidad de categorías distintas, con al menos una investigación asignada a una facultad o centro, ascendió a 284 que se distribuyen por Facultad según lo que se observa en la figura 5.5

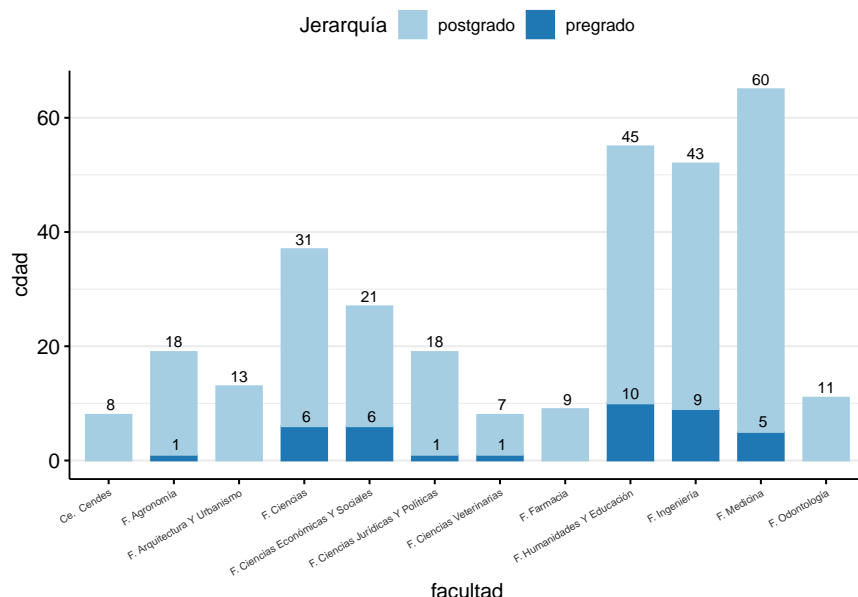


Figura 5.5: Cantidades de categorías por facultad y nivel académico

En en la figura 5.6 se pueden ver primero la cantidad total de investigaciones que pudieron ser clasificadas por cada Facultad - Centro de estudios y en la segunda posición, la cantidad de documentos disponibles por fecha de publicación que abarcan el período 01/01/1977 al 06/15/2023.

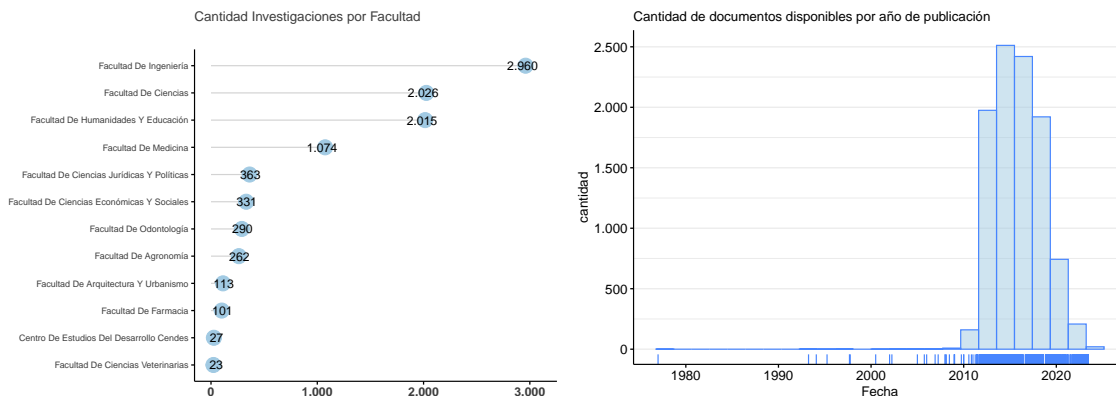


Figura 5.6: Cantidades de investigaciones clasificadas por Facultad y por año de publicación

En cuanto a la obtención de los nombres de los tutores, el procedimiento adoptado fue nuevamente realizar algunas limpiezas sobre el texto como remover dígitos,

abreviaturas de títulos académicos (PhD, MgS, etc), y luego extraer el texto que se encontraba delimitado entre la propia palabra “tutor” y el brinco de línea “\n” más próximo a la aparición de dicha palabra. Con el procedimiento descrito se pudo extraer un total de 7.969 nombres, equivalente al 79,8% de las investigaciones, así como 3.718 nombres únicos. En la figura 5.7 se aprecia la frecuencia para la cantidad de investigaciones que corresponden a un mismo tutor.

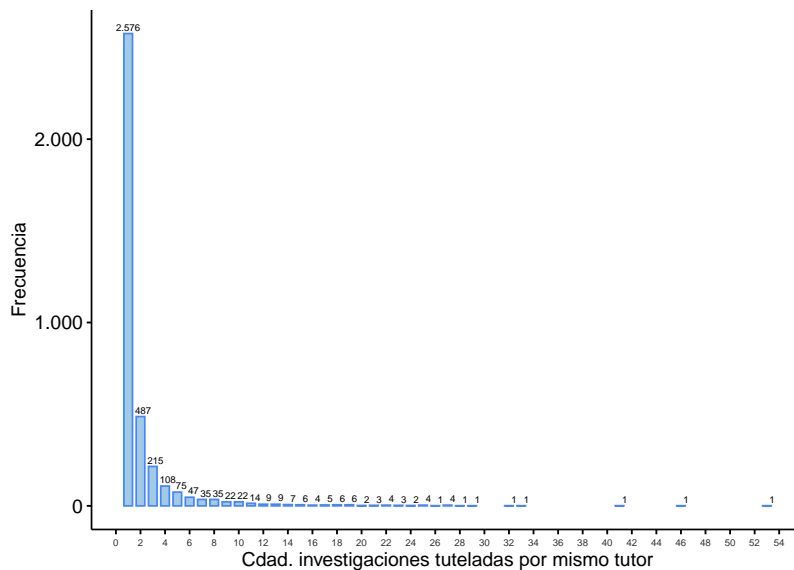


Figura 5.7: Histógrama Nombres Tutores Extraídos

La cantidad de trabajos donde fue detectado que un tutor tuvo dos o más trabajos tutelados es 5.393, equivalente a un 54%. La cifra anterior tiene la significancia de mostrar que para más de la mitad de los trabajos existe la certeza de que se encontró un nombre idéntico, en al menos dos casos, brindando mayor confianza sobre el proceso ejecutado.

En los otros 2.576 casos, en que sólo se encontró un tutor por investigación, al realizar un análisis exploratorio, se pudo apreciar que en una variedad de casos el nombre no fue escrito en forma idéntica entre un trabajo y otro, p. ej. dejando únicamente la inicial del segundo apellido u omitiendo el segundo nombre. Escapa al alcance de este trabajo realizar la depuración que permita consolidar más trabajos de grado al correspondiente tutor.

5.3.1.3.3 Aprender: A continuación se agrupan y enuncian los principales inconvenientes que se encontraron y se indica el aprendizaje obtenido para fases sucesivas de desarrollo:

5.3. CICLOS DE DESARROLLO:

1. Aparición de errores en la redacción y ortografía por parte de los autores, como por ejemplo, escribir incorrectamente el nombre del título al que optan o la facultad donde realizaron los estudios. Esto implicó crear reglas para realizar reemplazos de palabras en los textos y limpiezas para disminuir el ruido y facilitar el proceso de obtener la coincidencia. Un problema que se presentaría al intentar extender el SCSU a otros repositorios, cuando los trabajos de grado no estén categorizados, es que el procedimiento anterior no es completamente generalizable si los nombres de los postgrados-escuelas y el de las facultades son distintos, ya que las reglas de modificación de texto son específicas para el corpus de la Universidad Central de Venezuela.
2. Cambios en el estilo y formalidades con que se deben presentar los documentos de grado en las distintas facultades o niveles académicos, cuestión que dificultó la detección de las reglas para hacer la comparación. Ante esto se buscó encontrar las formas más genéricas en el procesamiento, así como la adopción de un procedimiento que progresivamente intentaba obtener la coincidencia: primero con el nombre del título, en caso de fallo se sigue con el nombre del pregrado o postgrado, si nuevamente fallaba se procedía a hacer la búsqueda del nombre de la facultad y finalmente si ninguna de las estrategias anteriores tenía éxito, se aplicaba el algoritmo “Smith Waterman”, con el cual se pudieron clasificar 1.658 trabajos, que equivalen a un 16,6% del total. Es importante señalar que al aplicar este último recurso se pueden generar “falsos positivos” y por esto en la Sección 5.4 se hace una estimación estadística del error que puede representar acudir a este método.
3. Dentro de los archivos descargados se encontraron algunos en formato de presentaciones *power point* los cuales fueron desechados sólo siendo procesados los que estuviesen en formato *word* o *pdf* . Esto llevo a condicionar posteriormente que para ejecutar la descarga y hacer el procesamiento de extracción de datos, la extensión debía alguna de las mencionadas como válidas.
4. También se encontraron trabajos que contaban con más de un archivo disponible para descargar. En la fase inicial se descargaron 12.765 de documentos aunque la cantidad de trabajos disponibles en el repositorio (incluyendo duplicados) era 10.843. Al evaluar las razones que motivaban que existiera una cantidad superior de archivos vs. documentos, se

encontraron casos en que por investigación existía un archivo por capítulo y no se encontraba algún elemento que indicará la secuencia en que se debía cargar cada archivo para armar el documento unificado. Para estos casos sólo se tomó el primer archivo en la lista de *url's* disponibles para tratar de hacer el proceso de clasificación.

5. Se hicieron algunas simplificaciones sobre postgrados que dependen de dos facultades imputándolo sólo a una que fuese la primera en aparecer en el texto. Como queda fuera del alcance de esta Investigación determinar los casos en que existen este tipo de adscripciones compartidas se hizo esta simplificación⁴.
6. Algunos trabajos en su primera página incluyen el nombre de dos facultades o escuelas creando errores en la clasificación. P. ej., investigaciones que indican en la portada el siguiente texto “Facultad de Ciencias, Escuela de Computación, título: se realiza la propuesta de un sistema de gestión académica para la Escuela de Economía de la Facultad de Ciencias Económicas...”, lo cual genera múltiples clasificaciones. En estos casos se optó por realizar la imputación con base en la primera coincidencia detectada.
7. En la obtención del nombre del tutor es importante destacar que en varios casos el texto extraído no se corresponde propiamente al nombre del mismo, motivado a que la escritura de la portada puede responder a la representación visual. Se encontraron trabajos que tienen tutor académico, tutor industrial, cotutor y otras variantes, donde se hace una disposición en la escritura de colocar el tipo de cada tutor alineado cada uno en los extremos de una línea y los nombres también se disponen en los extremos de la línea superior, quebrando la regla de extracción que se había diseñando. Esto pareciera un problema a enfrentar con técnicas de segmentación de archivos que tienen en consideración la disposición visual. En el caso de esta investigación no se adoptaron métodos para abordar este problema. En las Sección 5.4 se hace una evaluación estadística de la precisión alcanzada en esta extracción.
8. Se simplificaron algunos nombres de especializaciones por la cantidad de palabras que tienen estableciendo un límite, o *prunning*, de 5 palabras para el

⁴Un ejemplo de esto es la Maestría en Física Médica que es impartida de manera conjunta por la Facultad de Ciencias y por la Facultad de Medicina. Al no disponer de información oficial sobre otros casos de postgrados que presenten esta característica, se adoptó el método mencionado de imputar sólo a una dependencia que sea la primera en aparecer en el texto.

5.3. CICLOS DE DESARROLLO:

nombre del postgrado, lo que implica que algunos trabajos habrán quedado agrupados en la misma categoría. Estos casos mayormente están asociados a las especializaciones en el área de medicina, teniendo de ejemplo la figura 5.8.



Figura 5.8: Ejemplo de nombres simplificados

9. En este proceso de clasificación no resultaba conveniente usar técnicas de aprendizaje automático dada la cantidad de categorías y el gran desbalanceo de clases.
10. Se detectó que en Saber UCV en la categoría que se denomina “otros” en Saber UCV se encuentran documentos que corresponden a especializaciones y también 60 trabajos de ascenso de profesores.

5.3.1.4 Objetivos alcanzados:

1. Obtener las fichas de 9.982 investigaciones, equivalente al 100% de los trabajos de pre y postgrado alojados en Saber UCV (descartando los duplicados) , conforme a lo propuesto en el Objetivo Específico 1.
2. Obtener 375 nombres de postgrado y 50 de carreras de pregrado, insumo necesario para alcanzar lo propuesto en el Objetivo Específico 2.
3. Categorizar 9.585 investigaciones por área académica, equivalente al 96% de toda la información a clasificar, de acuerdo al Objetivo Específico 2.
4. Extraer 7.969 nombres de tutores, equivalente al 79,8% del total de investigaciones, que también se estableció como parte del Objetivo Específico 2.

5.3.2 Ciclo-Prototipo del SCSU:

En este ciclo se desarrolló el **Prototipo del Sistema Complementario Saber UCV** con tres iteraciones. En la primera iteración de este ciclo 5.3.2.1 se realizó la **Preparación del Corpus**. En la segunda iteración se hicieron las pruebas para realizar las **Recomendaciones** de cada documento, basado en la similitud que presente con el resto de las investigaciones. En la tercera 5.3.2.2 5.3.2.3 se creó una aplicación web interactiva en la que se implementó el **Prototipo** donde ante un *query* se presentan los resultados y la visualización de los “**Mapa de Conocimiento**” (ver 3.3.2.2.1) .

Todo las rutinas y codificaciones que se mencionan a continuación fueron realizadas en el lenguaje R version 4.3.2 (2023-10-31) (R Core Team, 2023).

5.3.2.1 Iteración- Preparación del Corpus:

En esta iteración el Corpus que se conformó en 5.3.1, fue sometido a distintos procesamiento conocidos como “Preparación del Corpus” mediante la aplicación de técnicas de PLN, ver 3.3.1. .

5.3.2.1.1 Especulación: Creando un corpus anotado con métodos del Procesamiento del Lenguaje Natural se facilita el acceso a información de relevancia para los investigadores mediante el análisis lingüístico de los documentos recolectados (Ruiz Fabo and Bermúdez-Sabel, 2019).

5.3.2.1.2 Colaboración: Para realizar el etiquetado de la “Parte del Discurso (POS)” 3.3.1.2, se hizo una revisión exhaustiva de distintos *frameworks* para realizar el anotado del Corpus, como *Freeling* (Padró and Stanilovsky, 2012), CoreNLP (Manning et al., 2014), UDPIPE (Wijffels, 2023a) y se decidió acudir a la librería “spacyr” (Benoit and Matsuo, 2020a) que es un *wrapper* de la librería Spacy (Honnibal et al., 2020), que se ejecuta en el lenguaje python. Esta librería, en lo relativo al etiquetado de las palabras por su función gramatical, lo hace acorde al marco de trabajo propuesto en la investigación “Universal Dependencies” (de Marneffe et al., 2021). La implementación de *Spacy* dispone de un encadenamiento en el flujo de trabajo, resultando conveniente aplicarlo dentro del desarrollo del SCSU. Otra de las características que dispone es contar

5.3. CICLOS DE DESARROLLO:

con varios modelos preentrenados para realizar el etiquetado, así como disponer de métricas de desempeño que se encuentran dentro del estado del arte en el campo del Procesamiento del Lenguaje Natural.

La librería carga el modelo preentrenado de aprendizaje automático para el idioma español denominado “es_core_news_lg”. Mediante un *pipeline* que se observa en la figura 5.9 se ejecutan procesos que permiten conformar el Corpus Anotado.

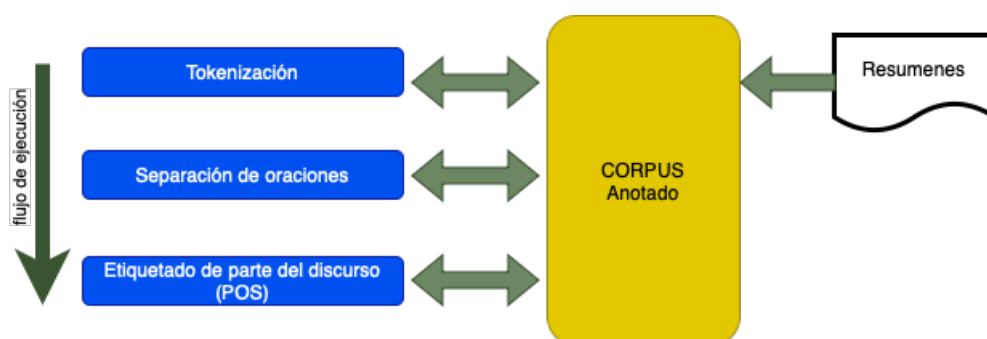


Figura 5.9: Arquitectura General pipeline Spacy

Ilustrativamente se muestra el texto “... y algunos no-metales. Contrariamente a lo...”, que al aplicar los métodos que dispone “spacyr”, permite generar el Corpus Anotado que se puede ver en la figura 5.10, donde se aprecia:

- id del documento
- id de la oración dentro del documento
- id del token dentro del documento
- token
- lema
- etiquetado de parte del discurso (POS) para cada token.

El Corpus Anotado cuenta con 9.982 documentos que generan 3.147.740 tokens, un vocabulario de 101.066 palabras distintas y 83.402 lemas únicos. Los tokens, agrupados por función gramatical se presentan ⁵ en el gráfico 5.11 :

Realizando el análisis exploratorio del Corpus, es interesante ver la cantidad de palabras únicas por función gramatical y el ratio que presentan con respecto al

⁵El proyecto Universal Dependencies disponible en el enlace <https://universaldependencies.org> contiene la documentación sobre las distintas funciones gramaticales que fueron etiquetadas.

doc_id	sentence_id	token_id	token	lemma	pos
<chr>	<int>	<int>	<chr>	<chr>	<chr>
c123875c1eab48ad98d434c750ea01b1	1	33	y	y	CCONJ
c123875c1eab48ad98d434c750ea01b1	1	34	algunos	alguno	DET
c123875c1eab48ad98d434c750ea01b1	1	35	no-metales	no-met...	PROPN
c123875c1eab48ad98d434c750ea01b1	1	36	.	.	PUNCT
c123875c1eab48ad98d434c750ea01b1	2	1	Contrariamente	contra...	ADV
c123875c1eab48ad98d434c750ea01b1	2	2	a	a	ADP
c123875c1eab48ad98d434c750ea01b1	2	3	lo	él	PRON

Figura 5.10: Detalle anotación del Corpus

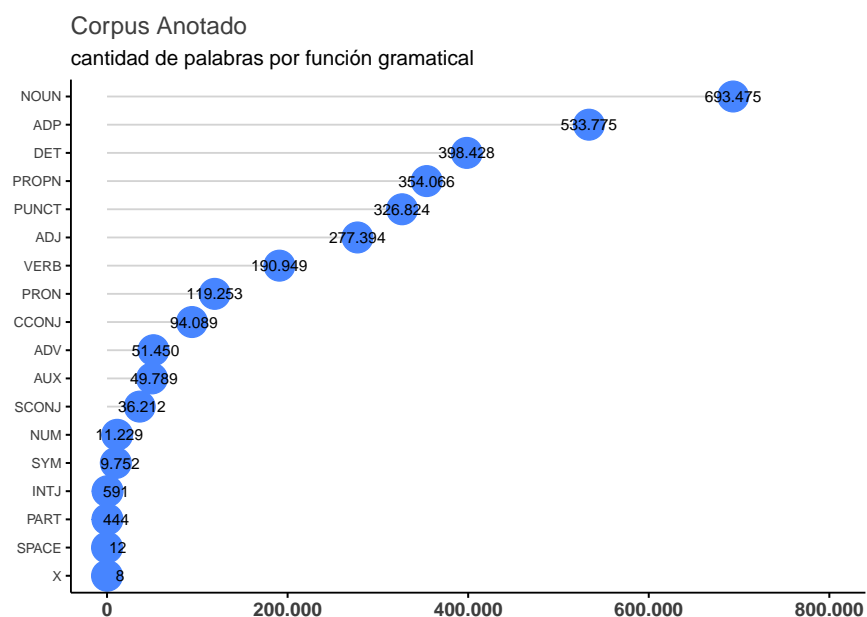


Figura 5.11: Cantidad de palabras por función gramatical

5.3. CICLOS DE DESARROLLO:

total de palabras por “POS” que se vio en la figura 5.11. En la tabla 5.4 se observa que las categorías del “etiquetado de la parte del discurso” que contienen mayor cantidad de información son los nombres propios (PROPN), los adjetivos (ADJ), números (NUM) y los sustantivos (NOUN), pero los nombres propios al ser únicos en cada trabajo y los números representar cantidades, realmente la mayor cantidad de información del Corpus se encuentra en los adjetivos y sustantivos, lo que refuerza la construcción y representación de los Mapas de Conocimiento mediante la selección de las palabras que tienen las funciones gramaticales de sustantivos y adjetivos, al presentar la mejor relación señal/ruido.

Cuadro 5.4: Ratio de Cantidad Única de lemas Vs. Cantidad total de lemas

POS	Cantidad	Cdad. únicas	ratio
PROPN	354.066	47.501	13,42
ADJ	277.394	16.584	5,98
NUM	11.229	542	4,83
NOUN	693.475	21.968	3,17
VERB	190.949	5.630	2,95
ADV	51.450	1.364	2,65
SYM	9.752	93	0,95
AUX	49.789	166	0,33
PRON	119.253	287	0,24
CCONJ	94.089	138	0,15
SCONJ	36.212	45	0,12
PUNCT	326.824	236	0,07
DET	398.428	246	0,06
ADP	533.775	262	0,05

Contar con un Corpus Anotado permite ir inspeccionando los patrones que se presentan en él. Un ejemplo se tiene en la figura 5.12 donde mediante el método *Rake*, que basado en la determinación de la frecuencia de palabras y patrones co-ocurrentes, logra extraer términos clave. En la figura se muestran las palabras claves asociadas al subconjunto de trabajos que fueron realizados en la Escuela de Computación.

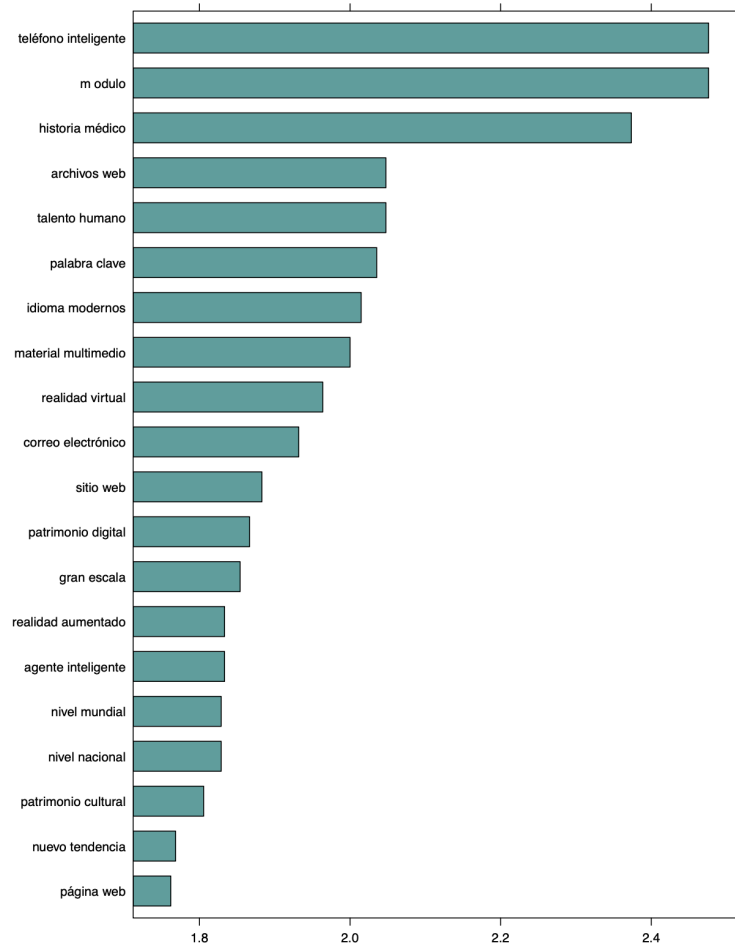


Figura 5.12: Palabras Claves en investigaciones de la Escuela de Computación

La tabla que conforma el Corpus Anotado para esta fase de la Investigación se alojó en memoria RAM en una estructura de datos tabular denominada *dataframe* del lenguaje R, sin ser registrada en un gestor de base de datos.

5.3.2.1.3 Aprender: Al analizar el Corpus anotado se detectaron algunos puntos a considerar para análisis posteriores.

5.3. CICLOS DE DESARROLLO:

- El proceso de separación por tokens y de lematización, como ya se mencionó, se hizo con la librería “spacyr”, que hace estos procesos mediante el uso de un modelo preentrenado de aprendizaje automático. Hubo ciertas palabras, o términos que son muy específicos de un área de conocimiento, como en la química, donde los *compuestos*, *moléculas*, *nomenclaturas* u otros, contienen denominaciones conformadas por cadenas de letras mayúsculas seguidas de puntos, que no presentan estructuras gramaticales propias del lenguaje natural, sino de un dominio específico, ver figura 5.13. Al tener un modelo intentando hacer la separación de los tokens o el etiquetado, bajo unos textos con los cuales no fue entrenado, el proceso falla en la clasificación. Este problema, para ese tipo de términos, no fue resuelto al no disponer con un modelo entrenado para este dominio. En investigaciones posteriores pudiera realizarse un sobreentrenamiento que incluyera el etiquetado de estos *tokens* de dominios muy especializados, como generalmente se presenta en carreras científicas y así poder mejorar la fase del etiquetado de las funciones gramaticales (POS).

En el presente estudio se determinaron las constantes de acidez de los ligandos ácido trans-1,2-diaminociclohexano-N,N,N',N'-tetraacético CDTA (H4C), ácido dietilentiaramino-N,N,N',N',N''-pentacético DTPA (H5C) y ácido etilenglicol-bis(2-aminoetilet)-N,N,N',N'-tetraacético EGTA (H4C), y las constantes de formación para los sistemas H+-VV(CDTA, H+-V(V)-DTPA y H+-V(V)-EGTA. Todas las constantes se determinaron a través de medidas de emf(H) a 25°C en KCl 3,0 M. Se empleó un procedimiento experimental en dos etapas, en la primera se valoró una alícuota de [H] con [OH] hasta alcanzar la neutralidad para determinar los parámetros Eo y JH. En la segunda etapa, sin remover los electrodos, se añadió una cantidad pesada de los ligandos y se valoraron con [H] el CDTA y EGTA; y el DTPA con [H] y [OH]. En la segunda etapa, para los sistemas H+-V(V)-CDTA, H+-V(V)-DTPA y H+-V(V)-EGTA, se añadió al reactor una alícuota de {VO3-} y una cantidad pesada del ligando (R = 1/2, 1 y 2 para el DTPA; R = 1, 2 y 4 para el CDTA; y R = 1 y 2 para el EGTA) y se valoró con [H]. El tratamiento de los datos experimentales se realizó utilizando el programa computacional de mínimos cuadrados generalizados LETAGROP. Los resultados obtenidos de las constantes de acidez para el DTPA son: pKaH7C2+ = 0,7(2), pKaH6C+ = 1,51(6), pKaH5C = 2,34(7), pKaH4C = 2,46(6), pKaH3C2- = 4,26(4), pKaH2C3- = 8,37(4), pKaHC4- = 9,96(2). Para el CDTA: pKa H6C2+ = 1,2(4), pKa H5C+ = 1,50(7), pKa H4C = 2,52(3), pKa H3C- = 3,25(2), pKa H2C2- = 6,25(3), pKa HC3- = 11,91(2). Para el EGTA: pKa H6C2+ = 1,52(4), pKa H5C+ = 2,07(4), pKa H4C = 2,35(3), pKa H3C- = 3,57(3), pKa H2C2- = 9,22(6), pKa HC3- = 9,69(4). Es importante mencionar que en el sistema H+-VO2+-DTPA se encontraron las especies mononucleares [HVO2C]3-, [H2VO2C]2-, [H3VO2C]-, [H4VO2C] y las especies dinucleares [H(VO2)2C]2-, [H2(VO2)2C]-, [H3(VO2)2C] con las siguientes constantes de estabilidad: $\beta_{111} = 1020,35(6)$, $\beta_{211} = 1025,53(3)$, $\beta_{311} = 1028,72(3)$, $\beta_{411} = 1030,76(3)$, $\beta_{121} = 1027,35(7)$, $\beta_{221} = 1030,70(6)$, $\beta_{321} = 1032,63(1)$. Para el sistema H+-VO2+-CDTA las especies mononucleares [HVO2C]2-, [H2VO2C]-, [H3VO2C]-, [H4VO2C]+ y [H5VO2C]2+ y la especie dinuclear [H2(VO2)2C] con las constantes de estabilidad $\beta_{111} = 1019,2(1)$, $\beta_{211} = 1023,5(1)$, $\beta_{311} = 1027,20(3)$, $\beta_{411} = 1029,71(4)$, $\beta_{511} = 1031,97(2)$ y $\beta_{221} = 1028,9(1)$ respectivamente. En el caso del sistema H+-VO2+-EGTA las especies mononucleares [H2VO2C]-, [H3VO2C] y la especie dinuclear [H2(VO2)2C] presentan las constantes de estabilidad, $\beta_{211} = 1025,79(3)$, $\beta_{311} = 1028,16(4)$ y $\beta_{221} = 1029,6(2)$.

Figura 5.13: Texto "Resumen" de Trabajo de Grado de Maestría en Química. Autora: Margarita González

- La lematización permitió reducir en un 17,5 % el vocabulario al pasar de 101.066 palabras a 83.402 lemas. Específicamente los lemas que se van a usar en fases posteriores del análisis, son los que están categorizados en el etiquetado del discurso como “ADJ” (adjetivos) que totalizan 277.394 289.283, equivalentes a un 8,8% del total de palabras y los “NOUMS” (sustantivos) son 693.475 equivalentes al 22% de las palabras presentes en el corpus. En el registro en base de datos se conservará completo el corpus anotado independientemente de la función gramatical con se etiquetasen.
- Los textos del “Resumen” que se encuentran en la ficha de cada trabajo

alojado en Saber UCV, en 396 casos, equivalentes al 4.3% del total, contienen una porción de texto en idioma inglés referente al “Abstract”, siendo óptimo aislar únicamente las partes que se encuentran en idioma español para que funcione correctamente el “etiquetado de la parte del discurso”.

5.3.2.2 Iteración - Recomendación Documentos:

En esta iteración se revisó la creación de recomendaciones de investigaciones, basándose en la similitud que presente un documento con el resto de los documentos presentes en el Corpus.

5.3.2.2.1 Especulación: Se ha determinado que los investigadores no necesariamente realizan la búsqueda de documentos que le puedan ser de interés mediante un *query* que contenga términos claves sino a partir de un documento que les resulta de interés quieren localizar otros que puedan compartir ciertos aspectos (Zhou et al., 2018).

La similitud de un documento con otro se puede entender, en esta implementación, como aquellos que tienden a compartir palabras y dentro de una “term document matrix” generan vectores similares (Jurafsky and Martin, 2009). Es por esto que se considera que los documentos que presenten mayor similitud, pueden resultar de interés para los investigadores, expandiendo así las posibilidades de inspección del Corpus.

5.3.2.2.2 Colaboración: Mediante la creación de una “term document matrix”, vista en 3.3.2.1, usando el framework para análisis cuantitativo de textos “quanteda” (Benoit et al., 2018a), la matriz obtenida refleja características esenciales de los documentos, en lo relativo a las palabras que lo conforman y su frecuencia.

En cada fila de la matriz se representa un documento, también equivalente a un vector. Midiendo la similitud *coseno*, revisada en 3.3.3, con la función `textstat_simil`, entre un determinado documento y el resto de los que conforman el corpus, es viable determinar cuáles son los más “parecidos”.

Se realizó un proceso iterativo para calcular la similitud coseno entre documentos y se pudo apreciar que los resultados de similitud pueden resultar de interés en

5.3. CICLOS DE DESARROLLO:

diversos casos, como el que se muestra en la figura 5.14.

Evaluación de la problemática de remoción de Fósforo y cloruros de los efluentes residuales De una empresa manufacturera de alimentos Ubicada en el estado aragua
Diseño y Evaluación de un sistema colorimétrico para análisis clínicos en celdas ELISA.
Desarrollo de herramientas electrónicas para los servicios industriales del complejo mejorador de crudo de Petrolera Ameriven.
Aumento de la prestación energética de los edificios con el uso de sistemas de iluminación de alta eficiencia
Diseño del proceso de preparación de bebidas carbonatadas no alcohólicas

Figura 5.14: Resultado de recomendaciones al documento: "Diseño de un sistema para la desinfección de aguas de consumo humano y de uso industrial empleando un material inorgánico antibacterial"

5.3.2.2.3 Aprendizaje: Es necesario señalar que el método adaptado no es se considera que esté dentro del "Estado del Arte" para realizar recomendaciones, no obstante, se hace la evaluación de este recurso dentro del SCSU, al ser de sencilla implementación, cálculos rápidos, robustos que no consumen mayores recursos computacionales, los cuales quedan disponibles para otras tareas que ejecuta el SCSU.

5.3.2.3 Iteración- Implementación Prototipo:

En esta iteración se hace el desarrollo del Prototipo de un "Sistema de Recuperación de Información" que servirá de base para probar distintas funcionalidades con las cuales contará el Sistema Complementario Saber UCV. En particular se hace el Prototipo de la aplicación web, mediante el uso del framework Shiny (Chang et al., 2023b) y se establecen los casos de uso, para que el usuario pueda realizar procesos de búsqueda, sobre el corpus anotado que se conformó en la iteración anterior 5.3.2.1.

En este Prototipo solo se va a trabajar con el subconjunto de las investigaciones que han sido realizadas en la Facultad de Ciencias y ante un *query*, son extraídos los documentos que contengan tales palabras y con ellos se representan los "Mapas de Conocimiento".

Esta implementación se apoya en el uso del sistema gestor de base de datos PostgreSQL, ya que este software nativamente cuenta con la funcionalidad de construir un índice invertido, sobre un conjunto de documentos, y así se da sustento a las "búsquedas de texto completo".

5.3.2.3.1 Especulación: Contar con un prototipo permitirá evaluar las posibles interacciones entre el usuario y la versión que entre a producción del Sistema

Complementario Saber UCV, así como analizar los tiempos de respuesta y los resultados de las visualizaciones.

Para el desarrollo de este prototipo se seleccionan los textos resúmenes que fueron categorizados como pertenecientes a la Facultad de Ciencias.

Durante la fase de especulación de este ciclo se realizaron los Diagramas de Caso de Uso que se ven en las figuras 5.15 y 5.16:

5.3. CICLOS DE DESARROLLO:

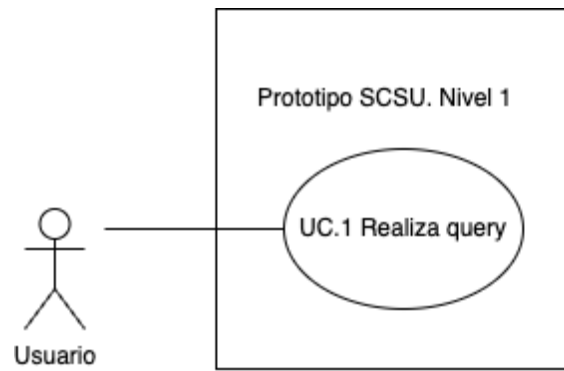


Figura 5.15: Caso de Uso 1 - Prototipo SCSU - Nivel 1

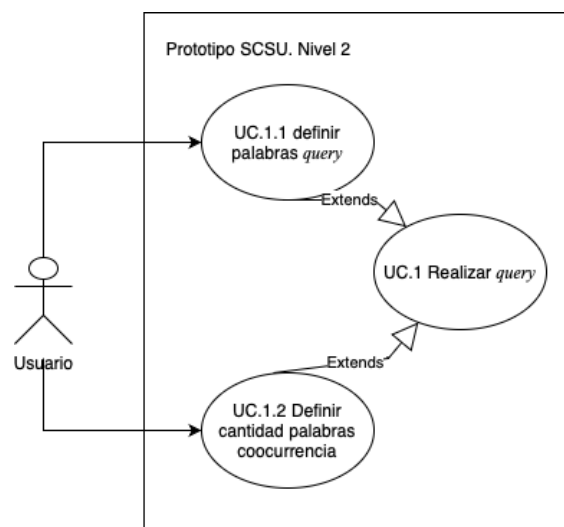


Figura 5.16: Caso de Uso 1.1 - Prototipo SCSU - Nivel 2

En el cuadro 5.5 se muestra el caso de uso UC.1 del Prototipo del SCSU.

Cuadro 5.5: Prototipo SCSU UC.1

Nombre	UC.1: Realizar proceso de recuperación de información (query)
Descripción	El usuario realiza búsquedas sobre los textos que conforman el Corpus del Prototipo
Actor	usuario
Flujo de Eventos	
Flujo Básico	
<p>El caso de uso inicia cuando el usuario ingresa a la aplicación y el prototipo muestra dos campos:</p> <ol style="list-style-type: none"> 1) En el primero debe introducir el texto a buscar. 2) En el segundo, en el que aparece por defecto el valor 60, se corresponde con las cantidades de coocurrencias que serán presentadas en los "Mapas del Conocimiento". 3) El usuario hace "clic" en el "buscar palabras" 4) El Prototipo presenta los resultados obtenidos 5) El caso de uso termina 	
Flujo Alternativo	
Título	Descripción
N/A	N/A
Precondiciones	
Título	Descripción
N/A	N/A
Postcondiciones	
Título	Descripción
Éxito	<p>El prototipo presenta:</p> <ol style="list-style-type: none"> 1) Total de investigaciones por jerarquías de las investigaciones que fueron recuperadas 2) Gráfico con "Mapas de Conocimiento". 3) Tabla con datos de los texto Resumen recuperados de acuerdo al query
Fracaso	N/A

5.3. CICLOS DE DESARROLLO:

En la figura 5.17 se muestra el mock-up de la Interfaz para el prototipo del SCSU.

Texto a buscar:

**Cantidad de
coocurrencias:**

Figura 5.17: Mock-Up de campos de entrada en la Interfaz del Prototipo

En cuanto a los “Mapas de Conocimiento (MC)”, motivado a que visualizar gráficamente los resultados es una forma conveniente de ayudar a los usuarios a descubrir relaciones y patrones presentes en los textos que pueden estar ocultos (Li, 2018), en la figura 5.18 se muestra la propuesta de representación de los “MC” en el Prototipo.

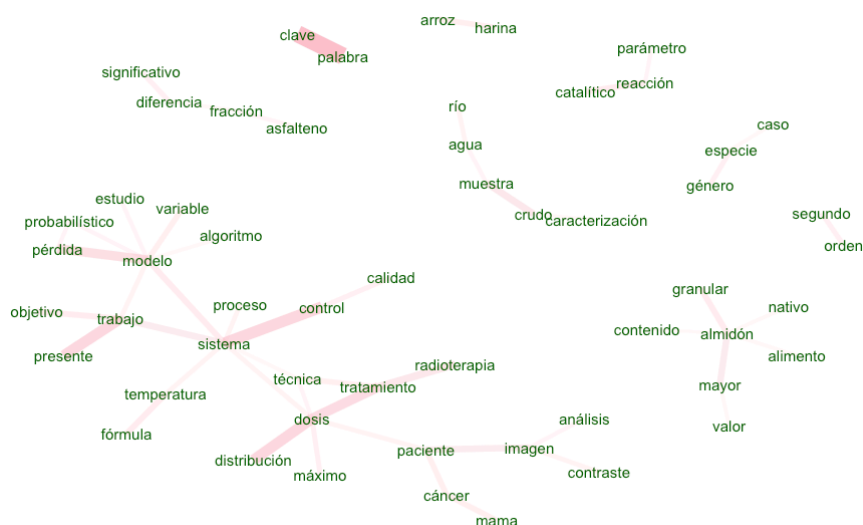


Figura 5.18: Representación Mapas de Conocimiento

Para poder implementar la búsqueda de texto se usará *postgreSQL V16*, por lo cual en la fase de “especulación” se realizó el diseño del modelo “entidad-relación” que se ve en detalle en la figura 5.19.

En el modelo, la entidad “Corpus” se apoya en la tabla “POS” que contiene el detalle del etiquetado de la parte del discurso para cada palabra (token) presente en

Diagrama Entidad Relación

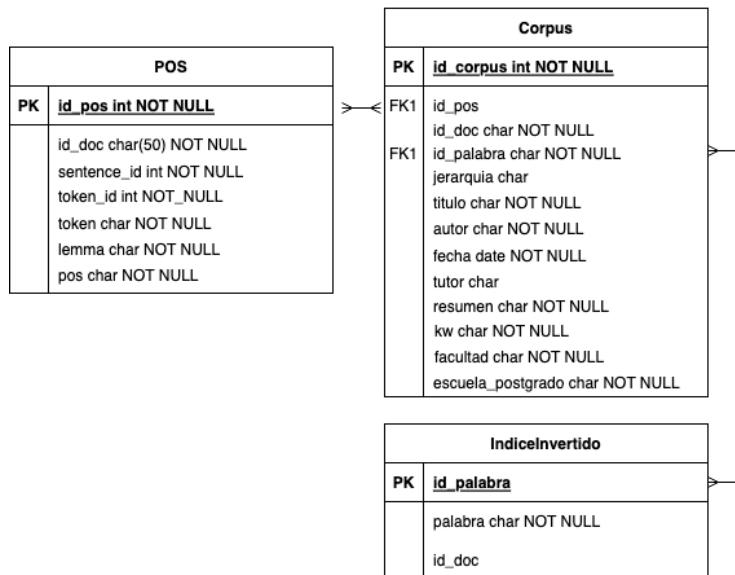


Figura 5.19: Modelo Entidad-Relación

cada documento, mientras que la tabla “Índice Invertido” se destinará a almacenar la estructura que permite general la búsqueda de texto y básicamente almacena cada palabra que compone el vocabulario y el *id* de los documentos donde aparece cada palabra.

Un factor determinante para la selección de PostgreSQL es el soporte que tiene para ejecutar procesos de “Búsqueda de Texto Completo (*Full Text Search*)” en el idioma español.

El esquema de la arquitectura general de la aplicación, el cuál se implementará en el *framework* shiny, se representa en la figura 5.20.

5.3.2.3.2 Colaboración: La aplicación web que formará parte del Prototipo, una vez que se tiene conformado el corpus anotado, como se vio en 5.3.2.1, se debe iniciar el “Poblado de la Base de Dato” con las tres tablas que se mostraron en el diagrama de entidad relación propuesto en 5.19 . A continuación se van a describir los procesos con los que se realizó el “Poblado” que sirve al Prototipo.

1. **Tabla Corpus:** con los datos del Corpus Anotado, se conforma la tabla con los siguientes datos: jerarquía, título, nombre autor, fecha publicación, nombre de

5.3. CICLOS DE DESARROLLO:

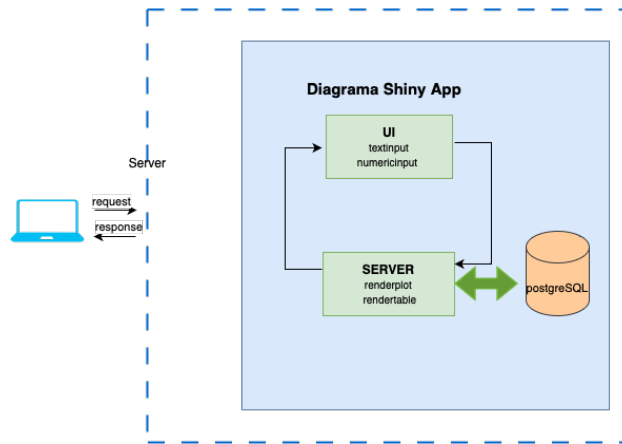


Figura 5.20: Esquema General del Prototipo de la Aplicación

tutor, texto resumen, palabras claves, nombre de la facultad y nombre de la escuela-postgrado y el código único identificador de cada documento.

2. **Tabla “índiceinvertido”:** mediante una extensión de *PostgreSQL* *V16.1* denominada "TS_Vector“, se crea una columna de nombre "document_tokens", donde se almacena una estructura de datos de tipo "tsvector“, que facilita al gestor de BD la búsqueda de texto completo. El script siguiente hace el trabajo de crear el índice invertido mediante la función "to_tsvector", para los datos del "título", "palabras claves", "autor" y del "resumen". En el mismo proceso se añaden pesos distintos (importancia para el criterio de reordenamiento de los resultados, según lo revisado en 3.2.5) a cada atributo mediante la función "setweight".


```
ALTER TABLE Corpus ADD COLUMN document_tokens tsvector
GENERATED ALWAYS AS ((setweight(to_tsvector('spanish',
    coalesce(autor, '')), 'A') ||
    setweight(to_tsvector('spanish',
    coalesce(titulo, '')), 'B') ||
    setweight(to_tsvector('spanish',
    coalesce(kw, '')), 'C') ||
    setweight(to_tsvector('spanish',
    coalesce(resumen, '')), 'D')) STORED;
```

En la figura 5.21 se puede ver parcialmente la estructura de datos de tipo “tsvector” que se genera para el texto “Las planicies de inundación son sistemas asociados al margen de un río que están sujetas a pulsos estacionales de inundación y sequía. La desembocadura del Río Mapire constituye un sistema complejo de planicie de inundación, como resultado del aumento del nivel de agua en el río por un efecto de represamiento por el Río Orinoco durante la época lluviosa. El gradiente de inundación que se forma genera un estrés ambiental en el sistema suelo que depende de la duración y profundidad....”.

document_tokens

```
'4':298 'abiot':439 'abord':190 'activ':270 'adapt':411 'afect':382,399 'agu':70,113,452
'alejandr':2A 'alt':555 'ambiental':97 'anual':560 'anzoategui':19B 'aquell':148,339
'asim':377 'asoci':9B,34,132,512 'aspect':3B,192,334 'aument':66 'bacteri':131,235
'baj':390 'bas':237 'bien':406 'biogeoquim':335,483 'biotic':437 'bosqu':497 'cad':188
'cambi':362,462,581,594 'cantid':615 'capitul':187,218,221,257,331 'caracteriz':121
'carbon':457,492,602 'cicla':490 'complej':58,442
'comun':6B,26C,126,230,244,345,358,408,466,539 'comunitari':158 'conclu':566
'condicion':169,414,635 'consider':206,282 'constitu':55,572 'conten':365,598
'context':274 'cual':343,469,620 'cuant':480 'cuy':630 'deb':578,591 'defin':303,453
'depend':103,499,528,632 'descomposicion':249,474,502 'describ':222
'desembocadur':51 'determin':147,394,471,487,522 'diferent':202,419,534,547,586,618
'dinam':224,455,508 'discut':214 'disolu':516 'disponibil':373,639 'distribu':631
```

Figura 5.21: Estructura de datos Índice Invertido

La función “to_tsvector” hace el parseo de un documento a tokens y estos son modificados realizando el *stemming* para extraer la raíz de la palabra, proceso que se revisó en 3.3.1.3, lo cual ayuda a disminuir el espacio de búsqueda y a realizar la recuperación de textos de manera más eficiente. Para lograr ejecutar este proceso, es necesario dar el parámetro “spanish” a la función, para que sea sobre la base del idioma español, que sea extraída la raíz.

5.3. CICLOS DE DESARROLLO:

La posición de la palabra dentro del texto, también queda registrada para así poder determinar en un *query* de varias palabras, la cercanía que tienen dentro del texto las palabras que son buscadas, dando una mayor relevancia a las que estén más cercanas.

Otro aspecto a destacar es que ciertas palabras de uso común y frecuente, denominadas *stopwords*, como: “el”, “la”, “y”, “a”, “en”, “con”, “para”; no son registradas, es decir que son omitidas así como los signos de puntuación, con lo cual un query que contenga una *stopword* o un signo de puntuación, no será buscada la coincidencia al no formar parte de la estructura de datos “*tsvector*”.

Para culminar el proceso de creación del “Índice Invertido” se debe ejecutar el comando `"CREATE INDEX document_weights_idx ON Corpus USING GIN (document_tokens);"` que cierra el ciclo de la generación del índice y es lo que permite realizar los procesos de búsqueda en una forma más eficiente, disminuyendo los tiempos de respuesta, pero debiendo tener presente que la creación del índice puede incrementar entre un cincuenta y doscientos por ciento el espacio de almacenamiento al generarlo. Una de las bondades que adicionalmente brinda la función `"TS_VECTOR"`, es que al añadir nuevos documentos a la base de datos, el índice será actualizado automáticamente.

3. **Tabla “POS”:** con los procesos que fueron revisados en el etiquetado de la parte del discurso en 5.3.2.1, se obtuvieron los datos que formarán parte de esta tabla, donde cada palabra que conforma el corpus, pasa a ser una fila identificando el documento de origen, la oración dentro del documento donde se encuentra la palabra, el número de orden de aparición de la palabra, la palabra, su lema y la función gramatical asignada, idéntico a lo que se observó en la figura 5.9.

En cuanto a la aplicación web, mediante el *framework* Shiny se implementó el Prototipo, con la codificación de dos componentes principales que son: la interfaz de usuario (UI) y el Servidor (Server).

Shiny se basa en la “programación reactiva”, lo que significa que ante cambios en los “campos de entrada” en la interfaz de usuario (UI), los elementos asociados se actualizan dinámicamente desde el componente “Server” sin necesidad de recargar la página, permitiendo crear una experiencia al usuario fluida. Shiny utiliza

conceptos como “observadores” y “reactividad” para mantener sincronizados los elementos de la UI con los datos subyacentes, que en el caso del Prototipo son el texto del *query*, el valor para establecer la cantidad de coocurrencias a representar en el Mapa del Conocimiento y el botón para ejecutar el *query*.

A nivel de la aplicación web, en la interfaz de usuario están estos tres componentes:

1. **textInput**: que corresponde a la casilla del *query*.
2. **numericInput**: la cantidad de palabras que se van a mostrar en la representación de los Mapas de Conocimiento.
3. **actionBtn**: botón de acción para ejecutar el *query*.

En el servidor de la aplicación, a nivel de generación de estructuras a ser representadas como resultados, dispone de estos componentes:

1. **dataTableOutput**: mediante la librería “datatable” (Xie et al., 2023a) se genera una tabla con los resultados obtenidos donde se incluyen los atributos: “Fecha”, “Título”, “Autor”, “Resumen”, “Facultad”, “Escuela”, “Nombre Tutor”.
2. **renderPlot**: con el uso de la librería “ggraph” (Pedersen, 2022a) se crea un grafo que reproduce los “Mapas de Conocimiento”, representando las palabras mediante nodos y la coocurrencia implica la unión mediante arcos, y a mayor grosor en el arco, quiere decir que la cantidad de veces que se presenta la coocurrencia es mayor.

En la figura 5.22 se muestra la versión implementada del prototipo donde ante la búsqueda de la palabra “física”, se recuperan los documentos que incluyen tal palabra y se generan los “Mapas de Conocimiento”.

Explicación de los servicios ofrecidos por la aplicación: el usuario introduce un texto con el cual se va a generar el *query* y la cantidad de palabras que quiere visualizar en los Mapas de Conocimiento. Posteriormente hace clic en “buscar palabras”.

Lógica de la aplicación:

5.3. CICLOS DE DESARROLLO:

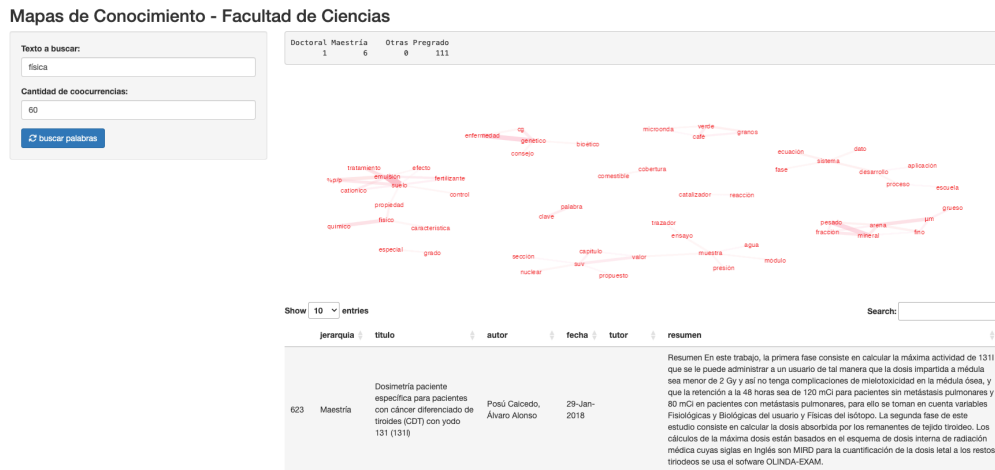


Figura 5.22: Estructura de datos Índice Invertido

En el detalle que se expone en las siguientes líneas es necesario mencionar que Shiny, funciona a nivel de computo como monohilo (“single-threaded”), por lo cual es estrictamente necesario que los pasos sean ejecutados secuencialmente según lo descrito.

1. **Definición “texto búsqueda”:** el usuario define el texto del query y la cantidad de coocurrencias.
2. **Procesamiento Query texto:** Cuando el servidor recibe el texto para generar el *query*, este es procesado con la sintaxis del siguiente código:

```
"select titulo, id_doc, facultad, jerarquia ,fecha,
autor, kw, resumen, nombre, tutor from Corpus where
document_tokens @@ websearch_to_tsquery('spanish',
'texto a buscar')
order by ts_rank_cd (document_tokens,
websearch_to_tsquery('spanish','texto a buscar')) desc
```

Lo relevante, es apreciar que el texto del *query*, mediante la función "websearch_to_tsquery", es sometido a un parseo que lo convierte a la estructura de datos “tsvector” que fue revisada anteriormente y así la hace compatible con el contenido de la columna “document_tokens”. El operador de coincidencia “@@” es el que determina si el texto del tsquery coincide con los distintos textos registrados en el “tsvector”. Si la frase

del query, por ejemplo "física química", que incluye dos palabras, el operador lógico que se intercala entre cada palabra es el "y (and)", no obstante, la propia función `websearch_to_tsquery` permite que se defina que pueda ser el operador "o (or)", por ejemplo si se escribe "física OR química".⁶

También incluye la función `order by ts_rank_cd` la cual es una implementación del método "*cover density ranking*" que fue introducido en la investigación de (Clarke et al., 2000), donde la relevancia se determina mediante la proximidad y coocurrencia de las palabras que conforman el query dentro de cada documento del Corpus ejecutando el reordenamiento, según lo visto en 3.2.5, teniendo como base los criterios de peso que habían sido definidos al crear el "tsvector" y también toma en cuenta la proximidad que puedan tener las distintas palabras que componen el query. Es conveniente citar la documentación de PostgreSQL relativa a esta función "*...es decir, consideran la frecuencia con la que los términos de la consulta aparecen en el documento, la proximidad de los términos en el documento y la importancia de la parte del documento en la que aparecen. Sin embargo, el concepto de relevancia es vago y muy específico de cada aplicación. Diferentes aplicaciones pueden requerir información adicional para la clasificación, por ejemplo, la hora de modificación del documento*".

3. **Query datos MC:** Con la lista de "id_docs", que fue obtenida en el paso anterior, se ejecuta un segundo *query* sobre la tabla "POS" y se seleccionan las palabras que cumplan con la condición de ser sustantivos y adjetivos, obteniendo la tabla filtrada con "doc_id", "token_id", "sentence_id", "pos", "lemma".

Contar en esta tabla con el "doc_id", "token_id" y el "sentence_id", sirve para determinar el nivel de representación de las coocurrencias, a modo de granularidad, permitiendo que posteriormente el resultado pueda ser representado con palabras que coocurren: a) una seguida de otra, b) dentro de la misma oración, o, c) dentro del mismo texto resumen.

4. **Generación de estructura Mapas de Conocimiento-coocurrencia:** a

⁶Otros operadores que se pueden usar en esta función son: "!" para excluir un término, "<->" para indicar que las palabras deben aparecer una seguida de otra. En la documentación oficial de PostgreSQL disponible en el enlace <https://www.postgresql.org/docs/current/textsearch-controls.html> se encuentra más información sobre los operadores que se pueden aplicar en los procesos de búsqueda.

5.3. CICLOS DE DESARROLLO:

los datos obtenidos en el paso (3) se les aplica la función *cooccurrence* de la librería (Wijffels, 2023b) que convierte los datos en una tabla de coocurrencias, donde se puede ajustar la granularidad revisada en (3).

5. **Render resultados:** El servidor hace el render del **dataTableOutput** y del **renderPlot** mencionados como los componentes del server.

5.3.2.3.3 Aprender: En la fase de aprendizaje se pudo detectar que es conveniente hacer distintas representaciones de los “Mapas de Conocimiento” con distintas granularidades y que incorporar interactividad a la representación puede constituir una herramienta de filtrado adicional para inspeccionar el Corpus. En el Ciclo de Integración de Componentes del Software 5.3.3 será abordado en detalle la propuesta final que se implementó en este particular.

Igualmente se constató la necesidad de mejorar los niveles de reproducibilidad para realizar la implementación del Sistema, mediante configuraciones que puedan ser independientes del sistema operativo y demás dependencias que estén preinstaladas en el Host, lo que llevó a realizar el diseño de la aplicación y sus componentes, mediante el uso de contenedores orquestados, lo cual se expone en 5.3.3.

Al crear el GIN (General Inverted Index) se pudo apreciar que el peso de la tabla con los datos del Corpus se incrementó en aproximadamente un 40%, no obstante, por ser un conjunto de datos relativamente pequeño, esto no representa un mayor problema, pero sí se debe tener en cuenta en caso de que el Sistema diese soporte a un Corpus mucho mayor.

Otro proceso de aprendizaje de este ciclo fue hacer los ajustes a la función de reordenamiento, según la relevancia, donde se estableció la siguiente jerarquía:

1. Autor
2. Título
3. Palabras Claves
4. Texto Resumen

Lo que quiere decir que si ante un texto de una búsqueda, en los documentos que conforman el Corpus, las palabras de query presentan coincidencia en el título, la

función de relevancia le otorgará mayor peso ese documento por sobre otro que pueda tener la misma coincidencia de aparición, pero el en texto resumen. Este ejemplo aplica para el resto de las combinaciones posibles.

Realizar el etiquetado de las partes del discurso, previamente a realizar la selección de los textos que serán representados mediante Mapas de Conocimiento, representa una mejora en los tiempos que tarda el Prototipo en realizar el render, lo cual refuerza la necesidad de contar con un sistema gestor de base de datos que pueda tener indexadas las distintas tablas que conforman el SCSU.

5.3.2.4 Objetivos Alcanzados:

1. Implementar un buscador de texto sobre un subconjunto del Corpus.
2. Hacer las pruebas de integración entre la base de datos con la aplicación web.
3. Contar con almacenamiento persistente para los datos.
4. Con los textos del Corpus generar recomendaciones de documentos que sean similares a una determinada investigación.

5.3. CICLOS DE DESARROLLO:

5.3.3 Ciclo Integración de Componentes del Software:

En este Ciclo se realiza la integración de los elementos constituidos y evaluados en procesos anteriores, tomando en cuenta los “aprendizajes” para realizar las modificaciones que permitan implementar la primera versión del **Sistema Complementario Saber UCV (SCSU)**. En la sección 5.3.3.0.1 se presenta el proceso de Especulación, en 5.3.3.0.2 se expone la Colaboración y finalmente en 5.3.3.0.3 el Aprendizaje alcanzado a lo largo del Ciclo.

5.3.3.0.1 Especulación: Usando las técnicas y métodos aplicados en ciclos anteriores e integrando los aprendizajes obtenidos, se puede implementar el SCSU. Concretamente se replicarán en este ciclo:

1. Usar la arquitectura “Modelo-Vista-Contralador” revisada en 5.2.
2. “Conformación del Corpus” revisado en 5.3.1, 5.3.1.1 y 5.3.2.1.
3. “Clasificación de Documentos” revisado en 5.3.1.3.
4. “Búsqueda de textos” revisado en 5.3.2 y 5.3.2.3.
5. Representación de “Mapas de Conocimiento (MC)” revisado en 5.18.
6. Generación de “Recomendaciones” revisada en 5.3.2.2.

En el proceso de especulación de este ciclo se aplicaron los métodos de la “Ingeniería de Software” necesarios para implementar el SCSU estableciendo formalmente los requerimientos funcionales y no funcionales, los diagramas y tablas de casos de uso, el modelo entidad-relación y demás aspectos necesarios, para posteriormente en la fase de Colaboración, realizar la implementación del Sistema.

5.3.3.0.1.1 Requerimientos Funcionales del SCSU:

1. Interactividad:

- **Descripción:**

La aplicación web debe ser altamente interactiva permitiendo al usuario de una manera fluida y receptiva interactuar con la interfaz para realizar búsquedas, explorar resultados y utilizar funcionalidades como filtros e inspección de “mapas del conocimiento”.

- **Criterios de Aceptación:**

- La interfaz de usuario debe responder de manera rápida y eficiente a las interacciones del usuario.

2. Búsqueda de Contenido

- **Descripción:**

El sistema debe permitir a los usuarios realizar búsquedas de contenido utilizando palabras clave o frases. La búsqueda debe ser insensible a mayúsculas y minúsculas, y devolver resultados en función a las palabras clave ingresadas.

- **Criterios de Aceptación:**

- El sistema debe proporcionar una interfaz de usuario intuitiva para la entrada de términos de búsqueda.
- Los resultados de la búsqueda deben mostrar: títulos, autores, fecha elaboración, palabras clave, facultad, nombre tutor, resúmenes.
- La búsqueda debe ser eficiente, respondiendo en un tiempo razonable.
- Se debe permitir a los usuarios hacer clic en un resultado para obtener más detalles sobre una investigación.

3. Filtrado de Búsquedas

- **Descripción:**

El sistema debe permitir a los usuarios aplicar filtros avanzados a los resultados de búsqueda para refinar y limitar la información recuperada. Los filtros pueden incluir fechas, jerarquía (nivel académico), facultad, escuela-postgrado.

- **Criterios de Aceptación:**

5.3. CICLOS DE DESARROLLO:

- Los filtros deben ser fácilmente accesibles y configurables.
- Las opciones de criterios de búsqueda deben actualizarse dinámicamente al aplicar los filtros.
- Debe ser posible combinar múltiples filtros para refinar aún más la búsqueda.

4. Generar Mapas del Conocimiento

• Descripción:

El sistema debe tener la capacidad de generar “mapas del conocimiento” con la información recuperada. Estos mapas deben representar de manera clara las relaciones y conexiones entre los conceptos dentro de la información mediante una estructura de grafos.

• Criterios de Aceptación:

- La generación de mapas del conocimiento debe ser automática y basarse en la estructura y contenido de los documentos recuperados.
- Los mapas deben ser interactivos, permitiendo a los usuarios explorar y comprender las relaciones entre los conceptos.
- Deben existir opciones para ajustar la complejidad y el nivel de detalle de los mapas.

5. “Rankiar” los Documentos Recuperados

• Descripción:

El sistema debe asignar un *ranking* a los documentos recuperados en función de su relevancia con respecto a la consulta realizada. La clasificación debe ser transparente y basada en algoritmos que consideren diversos factores como la frecuencia y proximidad de términos clave.

• Criterios de Aceptación:

- Aplicar algoritmo para reordamamiento y evaluar las métricas de desempeño.

- Se debe proporcionar una opción para ordenar los resultados de búsqueda según diferentes criterios, como relevancia o fecha.

6. Generar Recomendaciones

- **Descripción:**

El sistema debe ofrecer recomendaciones de contenido relevante basadas en similitud que tenga un documento con los otros que conforman el Corpus.

- **Criterios de Aceptación:**

- Las recomendaciones deben ser presentadas de manera clara indicando el título de la investigación recomendada con un vínculo al trabajo presentado.

7. Actualizar Periódicamente las Investigaciones

- **Descripción:**

El sistema debe realizar actualizaciones periódicas de la información almacenada, garantizando que los resultados de búsqueda sean siempre actuales y reflejen los documentos incorporados a Saber U.C.V. y que estos sean clasificados por área de conocimiento.

- **Criterios de Aceptación:**

- Deben establecerse intervalos regulares de actualización de la base de datos.
- Las actualizaciones no deben afectar negativamente el rendimiento del sistema durante las operaciones normales.

8. Cuadros de Ayuda (Tooltips)

- **Descripción:**

El sistema debe proporcionar cuadros de ayuda (tooltips) contextuales para guiar a los usuarios durante la interacción con la interfaz. Estos cuadros deben explicar términos técnicos, funciones específicas y proporcionar orientación sobre el uso eficaz del sistema.

5.3. CICLOS DE DESARROLLO:

- **Criterios de Aceptación:**

- Deben existir cuadros de ayuda accesibles en puntos clave de la interfaz de usuario.
- Los tooltips deben ser informativos y fácilmente comprensibles.

9. Accesibilidad Web

- **Descripción:**

El acceso a la aplicación se debe realizar mediante desde cualquier sitio mediante el uso de navegadores web.

- **Criterios de Aceptación:**

- Se debe garantizar la compatibilidad con los navegadores web más utilizados, como Chrome, Firefox, Safari y Edge.

5.3.3.0.1.2 Requerimientos No Funcionales del SCSU:

1. Rendimiento del Sistema

- **Descripción:**

El sistema debe ser capaz de manejar eficientemente los volúmenes de datos proporcionando tiempos de respuesta rápidos. El rendimiento del sistema debe optimizarse para garantizar una experiencia de usuario fluida y satisfactoria, independientemente de la complejidad de las consultas o la cantidad de usuarios concurrentes.

- **Criterios de Aceptación:**

- El tiempo de respuesta promedio para una consulta estándar no debe superar un segundo.

2. Utilización de Componentes Open Source

- **Descripción:**

El SCSU debe utilizar componentes de software (bibliotecas, frameworks y herramientas) que estén bajo licencias *open source*.

- **Criterios de aceptación:**

- Se debe realizar una evaluación de la licencia de los componentes utilizados para garantizar su compatibilidad con el modelo de desarrollo y distribución del proyecto.

3. Reproducibilidad del SCSU

- **Descripción:**

El sistema debe ser reproducible, lo que significa que la construcción, implementación y ejecución del software deben proporcionar resultados consistentes en diferentes entornos y momentos, siguiendo prácticas y estándares que faciliten la reproducibilidad del proceso de desarrollo.

- **Criterios de aceptación:**

- Todas las dependencias externas, incluyendo bibliotecas y componentes *open source*, deben estar claramente especificadas y gestionadas.
- Se debe proporcionar documentación detallada sobre cómo configurar el entorno de desarrollo y construir el Sistema.
- La aplicación debe ser capaz de ejecutarse correctamente en diferentes sistemas operativos y configuraciones de infraestructura.

4. Preprocesamientos:

- **Descripción:**

El Sistema, bien sea al momento de conformar la base de datos o al realizar actualizaciones, en la mayor medida posible debe ejecutar el preprocesamiento de los textos (PLN-POS, remoción stop words, índice invertido, etc.) para minimizar los cálculos al momento de la demanda de recursos cuando el usuario realiza las búsquedas.

- **Criterios de aceptación:**

- El sistema debe realizar el preprocesamiento de textos de manera eficiente durante la construcción inicial de la base de datos y las actualizaciones periódicas.

5. Modularidad del Sistema

- **Descripción:**

El sistema debe ser modular, dividiendo sus componentes en piezas simples que utilizan contenedores . Estos componentes deben estar orquestados de manera eficiente, permitiendo el intercambio, la asignación de recursos y la modificación de módulos individuales con un impacto mínimo en el resto del sistema.

- **Criterios de aceptación:**

- Los componentes del sistema deben estar encapsulados en contenedores, facilitando su independencia y despliegue consistente.
- Los componentes del sistema deben estar orquestados de manera eficiente, permitiendo una coordinación efectiva entre ellos y permitiendo la definición de asignación de recursos.
- Los componentes deben ser intercambiables y modificables de manera independiente, fomentando la flexibilidad y la evolución del sistema a lo largo del tiempo.

6. Mantenibilidad del Sistema

- **Descripción:**

El sistema debe ser diseñado de manera que permita la realización periódica de mantenimiento sin degradar el rendimiento ni la operatividad del Sistema.

- **Criterios de aceptación:**

- La realización de mantenimiento no debe causar una degradación significativa en el rendimiento del sistema garantizando la continuidad operativa.
- La realización del mantenimiento debe definirse con carácter periódico en la configuración.

5.3.3.0.1.3 Diagramas de Caso de Uso: El SCSU cuenta con los casos de uso que se observan en las figuras 5.23, 5.24, 5.25, 5.26, 5.27, 5.28 ; seguido de las tablas 5.6 ,5.7 , 5.8, 5.9, 5.10, 5.11, 5.12 que contienen la descripción de cada caso.

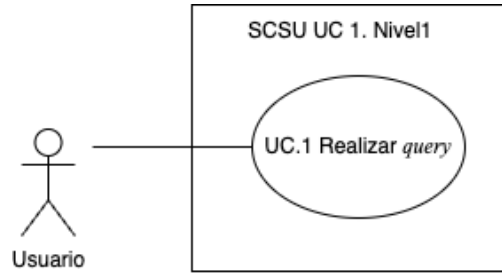


Figura 5.23: SCSU: Diagrama de Casos de Uso 1, Nivel 1.

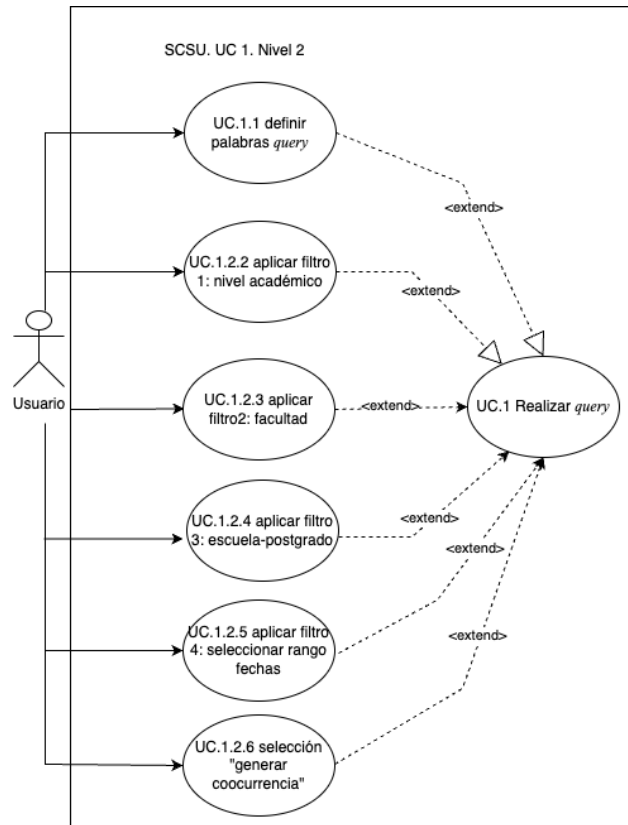


Figura 5.24: SCSU: Diagrama de Casos de Uso 1, Nivel 1.

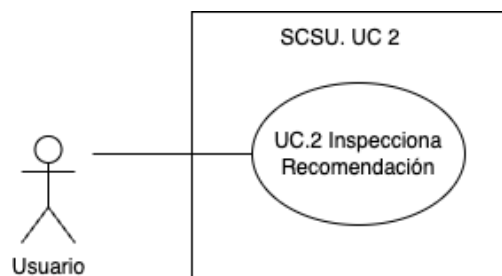


Figura 5.25: SCSU: Diagrama de Casos de Uso 2.

5.3. CICLOS DE DESARROLLO:

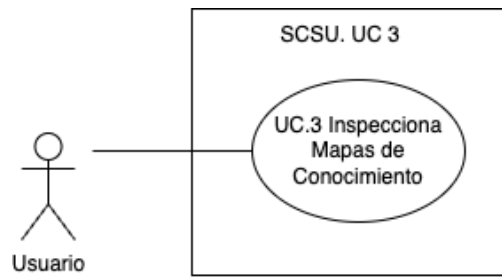


Figura 5.26: SCSU: Diagrama de Casos de Uso 3, Nivel 1

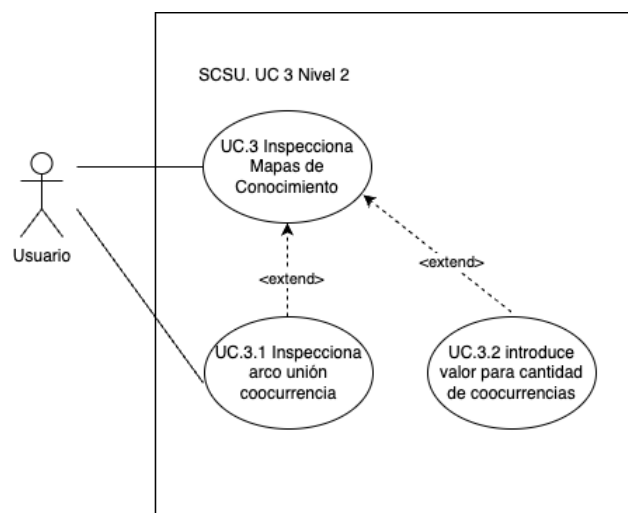


Figura 5.27: SCSU: Diagrama de Casos de Uso 3, Nivel 2.

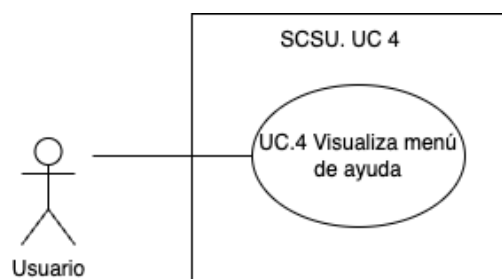


Figura 5.28: SCSU: Diagrama de Casos de Uso 4.

Cuadro 5.6: SCSU UC.1.

Nombre	UC.1: Realizar proceso de recuperación de información (query)
Descripción	El usuario realiza búsquedas sobre los textos que conforman el Corpus
Actor	usuario
Flujo de Eventos	
Flujo Básico	
El caso de uso inicia cuando el usuario ingresa a la aplicación:	
1) Introduce el texto a buscar.	
2) El usuario hace "clic" en "búsqueda"	
4) El SCSU presenta los resultados obtenidos	
4) El caso de uso termina	
Flujo Alternativo	
Título	Descripción
Aplica filtros	El usuario aplica filtros para restringir la búsqueda
Precondiciones	
Título	Descripción
N/A	N/A
Postcondiciones	
Título	Descripción
Éxito	El prototipo presenta: 1) Tabla con los resultados que incluye los campos "fecha", "título", "palabras claves", "autor", "facultad", "dependencia" y "tutor" 2) Gráfico con frecuencia de aparición del texto del query por año
Fracaso	No presenta ningún resultado

5.3. CICLOS DE DESARROLLO:

Cuadro 5.7: SCSU UC.1, Nivel 2.

Nombre	UC.1. Nivel 2: Realizar proceso de recuperación de información (query) aplicando filtros.
Descripción	El usuario realiza búsquedas sobre los textos que conforman el Corpus aplicando filtros
Actor	usuario
Flujo de Eventos	
Flujo Básico	
<p>El caso de uso inicia cuando el usuario ingresa a la aplicación y decide aplicar filtros al realizar la búsqueda. Los campos que se muestran son:</p> <p>1) Introducir el texto a buscar. 2) Filtrar el nivel académico. 3) Filtrar la Facultad 4) Filtrar la escuela o postgrado 5) Definir rango de fechas 6) Seleccionar "generación de coocurrencias (mapas de conocimiento)" 7) El usuario hace "clic" en el "búsqueda" 8) El SCSU presenta los resultados obtenidos 9) El caso de uso termina</p>	
Flujo Alternativo	
Título	Descripción
Sin filtros	El usuario no aplica ningún filtro y aparecen todos los resultados que contienen el texto de query
Precondiciones	
Título	Descripción
N/A	N/A
Postcondiciones	
Título	Descripción
Éxito	<p>El SCSU presenta:</p> <p>1) Tabla con los resultados que incluye los campos "fecha", "título", "palabras claves", "autor", "facultad", "dependencia" y "tutor"</p> <p>3) Gráfico con frecuencia de aparición del texto del query por año</p> <p>4) Gráfico con "Mapas de Conocimiento"</p>
Fracaso	No presenta ningún resultado

Cuadro 5.8: SCSU UC. 2.

Nombre	UC.2: Realizar Inspección de Recomendaciones
Descripción	El usuario inspecciona un documento de interés haciendo "clic" y se muestran los títulos y vínculos a investigaciones recomendadas
Actor	usuario
Flujo de Eventos	
Flujo Básico	
<p>El caso de uso inicia cuando</p> <p>1) el usuario inspecciona la tabla con los resultados de la búsqueda y hace "clic" sobre una fila y se expande la fila asociada a una investigación.</p> <p>2) el caso de uso termina</p>	
Flujo Alternativo	
Título	Descripción
no inspecciona	No se despliega el área que muestra las recomendaciones
Precondiciones	
Título	Descripción
Realizar query	Haber realizado el UC. 1.
Postcondiciones	
Título	Descripción
Éxito	<p>El SCSU presenta:</p> <p>1) listado con hasta cinco títulos de documentos recomendados que contienen el hipervínculo a la investigación alojada en Saber UCV</p>
Fracaso	No se muestran recomendaciones por no disponer de documentos similares en el Corpus

5.3. CICLOS DE DESARROLLO:

Cuadro 5.9: SCSU UC. 3.

Nombre	UC.3: Realizar inspección de "Mapas de Conocimiento"
Descripción	El usuario revisa los "Mapas de Conocimiento" generados con los documentos que fueron recuperados en el query
Actor	usuario
Flujo de Eventos	
Flujo Básico	
<p>El caso de uso inicia cuando el usuario:</p> <ol style="list-style-type: none"> 1) Hace "clic" en la pestaña de nombre "Coocurrencias". 2) Se muestra un gráfico interactivo con los "Mapas de Conocimiento" generados. 3) El caso de uso termina 	
Flujo Alternativo	
Título	Descripción
Sin Coocurrencia	El usuario no selecciona la ventana "coocurrencia" y no se muestran los "Mapas de Conocimiento"
Precondiciones	
Título	Descripción
Realizar query	Haber realizado el UC. 1.
Postcondiciones	
Título	Descripción
Éxito	El usuario navega con las flechas del teclado o con la rueda de scroll del mouse haciendo zoom in o zoom out sobre el gráfico de "Mapas de Conocimiento" para ver el detalle de los datos representados
Fracaso	No se muestran "Mapas de Conocimiento" por no disponer de datos para que sea generado el gráfico

Cuadro 5.10: SCSU UC. 3. Nivel 2.1.

Nombre	UC.3. Nivel 2.1: Realizar inspección "Mapa de Conocimiento"
Descripción	El usuario inspecciona un arco del "Mapa de Conocimiento"
Actor	usuario
Flujo de Eventos	
Flujo Básico	
El caso de uso inicia cuando el usuario:	
1) Hace "clic" sobre un arco de los "Mapas de Conocimiento".	
2) Aparece un "popup" con resultados.	
3) El caso de uso termina	
Flujo Alternativo	
Título	Descripción
Sin clic arco	El usuario no selecciona ningún arco en "Mapas de Conocimiento"
Precondiciones	
Título	Descripción
1) Realizar "query"	2) Haber realizado el UC. 1.
Selecc ventana "Coocurrencia"	Haber realizado el UC. 3.
Postcondiciones	
Título	Descripción
Éxito	Ventana popup con el listado de documentos que contienen las palabras que las une el arco sobre el cual se realizó el "clic" . En el listado se muestra el "título", "fecha", "palabras claves", "autor", "facultad", "escuela/postgrado"
Fracaso	No se muestra tabla por no disponer de datos para generarla

5.3. CICLOS DE DESARROLLO:

Cuadro 5.11: SCSU UC. 3. Nivel 2.2.

Nombre	UC.3. Nivel 2.2: modificar cantidad de coocurrencias en "Mapa de Conocimiento"
Descripción	El usuario modifica el valor correspondiente a la cantidad de coocurrencias a ser representadas en el "Mapa de Conocimiento"
Actor	usuario
Flujo de Eventos	
Flujo Básico	
El caso de uso inicia cuando el usuario:	
1) Modifica el valor que aparece por defecto (60) de coocurrencias a representar.	
2) El caso de uso termina	
Flujo Alternativo	
Título	Descripción
No modifica el valor	El usuario no modifica el valor que aparece por defecto
Precondiciones	
Título	Descripción
1) Realizar "query"	2) Haber realizado el UC. 1.
Selecc ventana "Coocurrencia"	Haber realizado el UC. 3.
Postcondiciones	
Título	Descripción
Éxito	Se modifica el "Mapa de Conocimiento" representado.
Fracaso	No se muestra tabla por no disponer datos suficientes para generarla

Cuadro 5.12: SCSU UC. 4.

Nombre	UC.4: ver cuadros de ayuda.
Descripción	El usuario al pasar el mouse sobre áreas de introducción de datos o pestañas, ve menús de ayuda contextuales para obtener información
Actor	usuario
Flujo de Eventos	
Flujo Básico	
El caso de uso inicia cuando el usuario:	
1) Pasa el mouse, sin hacer clic, sobre una determinada área de la interfaz de usuario	
2) El caso de uso termina	
Flujo Alternativo	
Título	Descripción
N/A	N/A
Precondiciones	
Título	Descripción
N/A	N/A
Postcondiciones	
Título	Descripción
Éxito	aparece un recuadro en el lateral próximo al puntero del mouse
Fracaso	No se muestra el menú contextual

5.3. CICLOS DE DESARROLLO:

5.3.3.0.1.4 Modelo Entidad Relación: El modelo “entidad-relación” que se ve en detalle en la figura 5.29, se construyó teniendo como base el diseño del Prototipo. En esta nueva versión hay otras tablas que sirven de apoyo a la entidad “Corpus” como la tabla “Facultad”, “Escuela_Postgrado” y “jerarquia”.

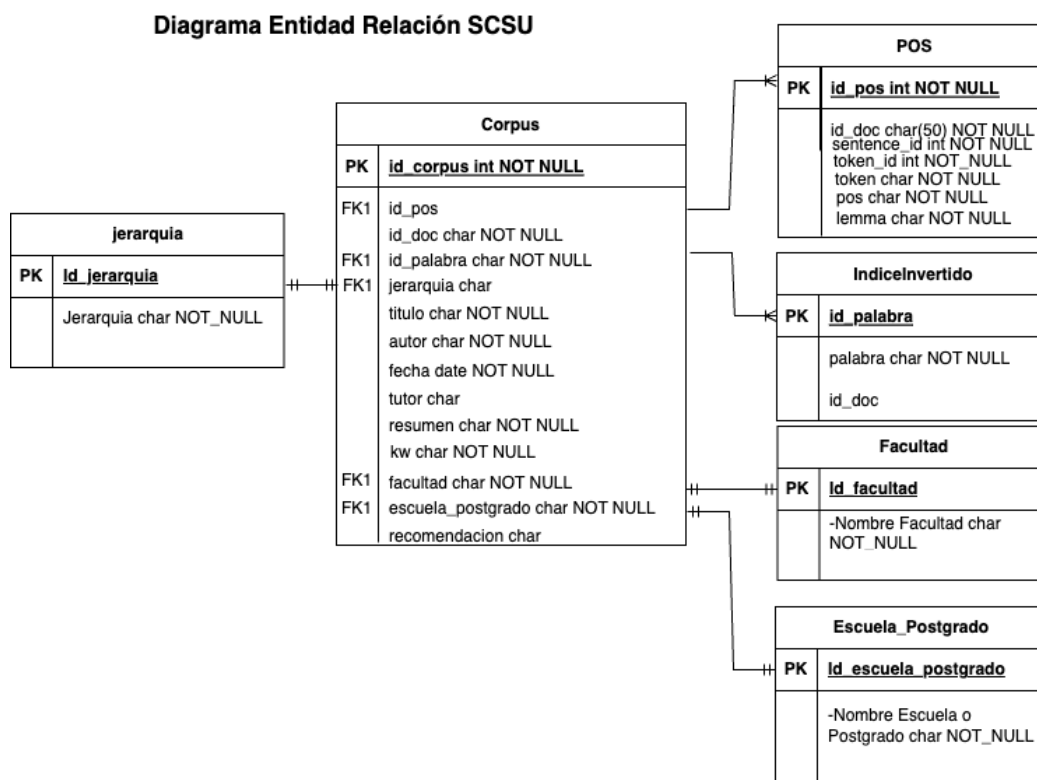


Figura 5.29: SCSU: Diagrama Entidad Relación

Adicionalmente a la tabla “Corpus” se le añade una columna que se corresponde con las “recomendaciones” generadas para cada documento, donde se almacena una *string* que corresponde a una estructura de datos “html”.

5.3.3.0.1.5 Diagrama General del Sistema- versión contenedores: En la figura 5.30 se define la estructura que será implementada, teniendo en cuenta los Requerimientos Funcionales y No Funcionales indicados anteriormente. La propuesta funciona como un Sistema Distribuido, ver 3.4, donde mediante el uso de contenedores, ver 3.4.1, se crean instancias en las que son ejecutadas los servicios necesarios para que el Sistema Complementario Saber UCV puede cumplir con los objetivos propuestos. La integración y la asignación de recursos a cada contenedor se hace mediante el uso de un orquestador, ver 3.4.2.

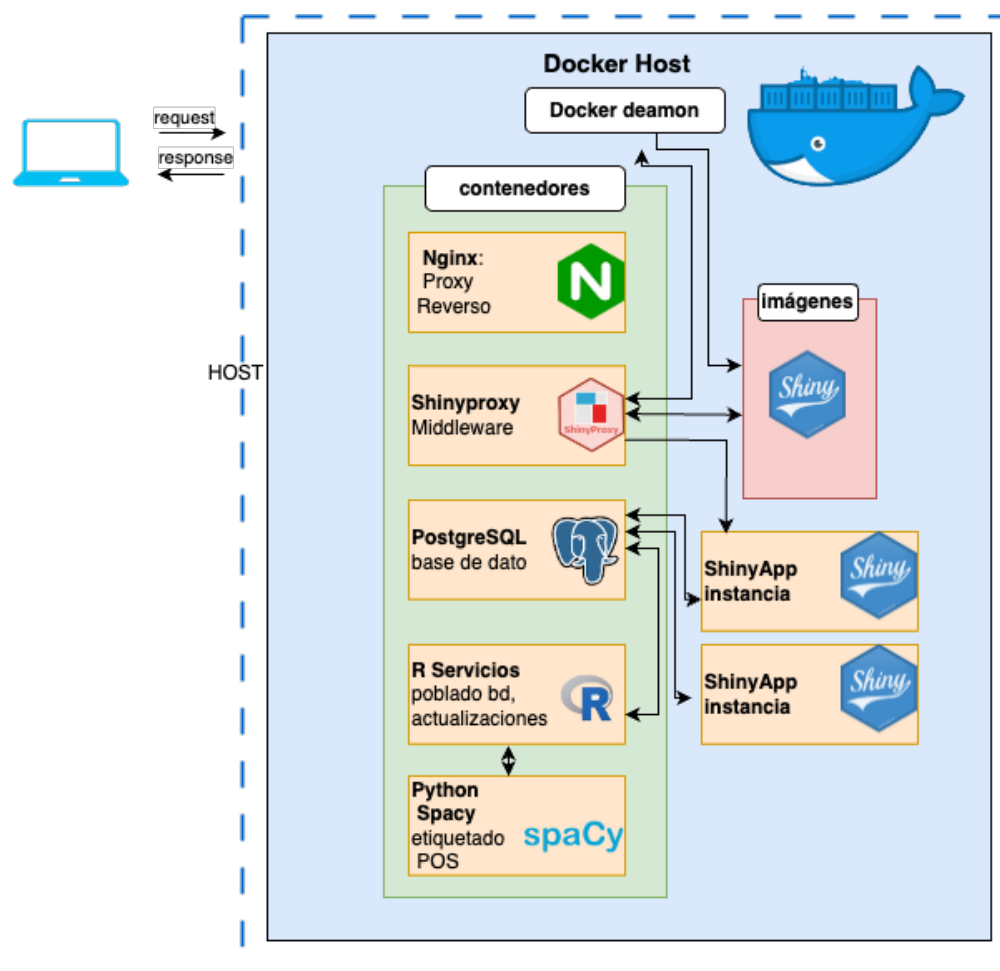


Figura 5.30: SCSU: Diagrama General del Sistema en Contenedores

5.3. CICLOS DE DESARROLLO:

En la fase de colaboración de este ciclo 5.3.3.0.2 se especificarán las tareas y funciones asignadas a cada contenedor.

5.3.3.0.1.6 Esquema General de Clasificación y Extracción de Datos del SCSU: En la figura 5.31 se observa el proceso de extracción y clasificación de datos que realizará el SCSU, obteniendo los valores desde el repositorio Saber UCV, teniendo dos ramas principales que alimentan a la base de datos.

La lógica de este proceso es que en la primera rama se descarga el documento anexo a cada investigación, cuando se cumplan las condiciones, se hace la lectura del documento y se inicia el proceso de clasificación por facultad y área de estudio, como se vio en la “Iteración- Extracción y Clasificación de las Investigaciones” en 5.3.1.3.

En la segunda rama se hace el “etiquetado del discurso”, que fue revisado en la “Iteración- Preparación del Corpus” en 5.3.2.1.

Ambos procesos alimentarán a las correspondientes tablas en la Base de Datos.

El proceso indicado se ejecuta inicialmente para realizar el poblado de la Base de Datos e igualmente se replica periódicamente, según parámetro definido en el archivo de configuración del orquestador, para actualizar e incorporar las nuevas investigaciones que se encuentren disponibles en el repositorio Saber UCV.

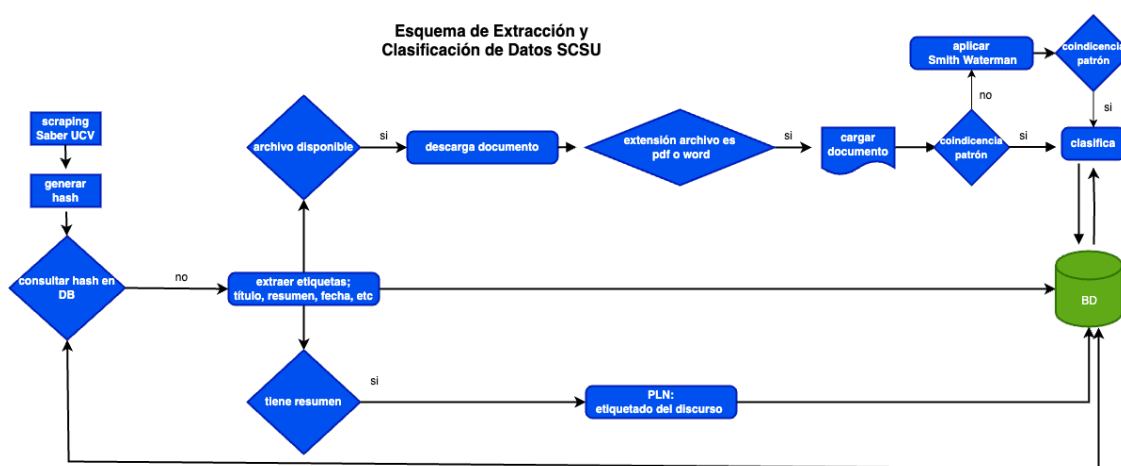


Figura 5.31: Esquema Extracción y Clasificación de Datos

5.3.3.0.1.7 Interfaz Visual (Mock Up): La interfaz cuenta con tres componentes:

1. **Sidebar:** en la barra lateral que se ubicará en la parte izquierda de la aplicación y es el área donde el usuario definirá el texto de búsqueda así como los parámetros que le acompañan. En la figura 5.32 se puede ver el diseño propuesto.

parámetros

texto a buscar (usar comillas dobles para búsqueda exacta)

Filtro 1: niveles académicos/categoría (opcional):

Filtro 2: por Facultad/Revista (opcional):

Filtro 3: por Escuela o Postgrado/Edición (opcional):

rango de fechas:

01-01-1977 hasta 12-09-2023

☐ generar coocurrencias, se incrementa el tiempo de ejecución

búsqueda

Figura 5.32: Interfaz de Usuario - Sidebar

2. **Main Tabs:** es la sección de pestañas que se encontrará en la parte media alta de la aplicación y es donde el usuario podrá seleccionar la visualización de los “tabla resultados”, bien sea para los tabla de los textos o para ver los “mapas del conocimiento”. En la figura 5.33 se representa la propuesta. Al seleccionar una pestaña está cambiará el color de fondo verde a gris.



Figura 5.33: Interfaz de Usuario - Pestañas

3. **Resultados:** debajo de “Main Tabs” se presentan los resultados obtenidos en el proceso de búsqueda. La representación es contextual basada en la selección de pestaña que está seleccionada (“tabla resultados” o “mapas de conocimiento”).

1. En las figura 5.34 se puede ver la representación de la tabla de resultados de una búsqueda.

5.3. CICLOS DE DESARROLLO:

Cantidad de resultados obtenidos en búsqueda "fisica": 911. Tiempo de ejecución: 0.5758

Buscar:

	fecha	titulo	kw	autor	facultad	nombre	texto_tutor
	All	All	All	All	All	All	All
1	2018-07-28	Niveles de autoconcepto físico en una muestra de deportistas de la Universidad Central de Venezuela.	psicología, autoconcepto, autoconcepto físico de goñi, práctica físico-deportiva, deportistas, universidad central de venezuela, asesoramiento psicológico	Alonso, Aroldi, López, Manuel	facultad de humanidades y educación	escuela de psicología	Phy Diego Arzola

Figura 5.34: Interfaz de Usuario - Tabla Resultados de Búsqueda

2. En la figura 5.35 se puede ver la representación propuesta de “mapas de conocimiento”.

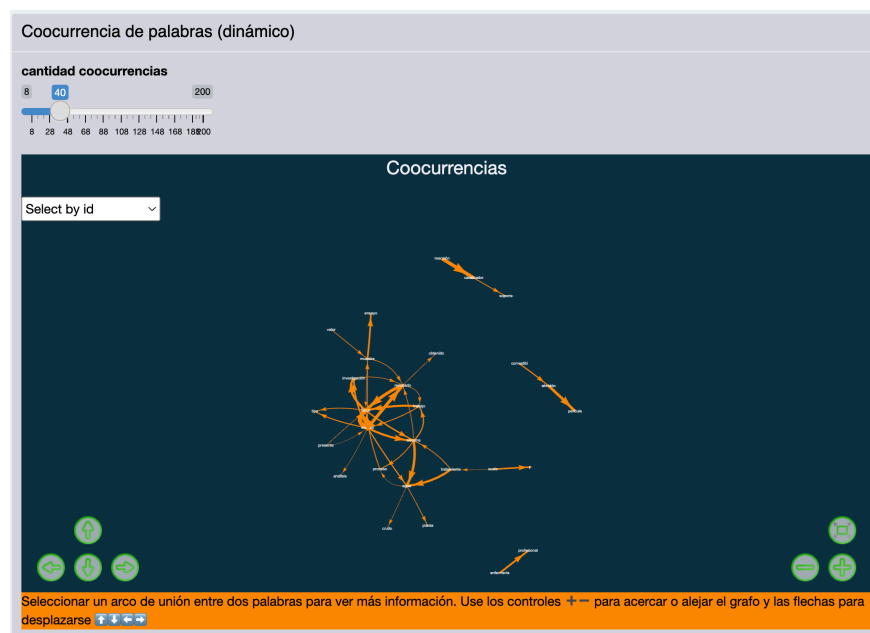


Figura 5.35: Interfaz de Usuario - Mapas de Conocimiento

Adicional a estos tres componentes la propuesta de interfaz incluye:

1. **Tooltips:** al hacer hover por áreas de introducción o selección de valores se desplegará un texto de ayuda según se ve en la 5.36.

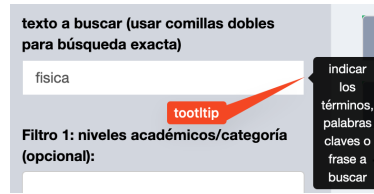


Figura 5.36: Interfaz de Usuario - Tooltips

2. **Inspección documento:** la tabla de resultados dispondrá de un ícono “+” que expande la tabla para mostrar el detalle y recomendaciones de un documento, como se observa en la figura 5.37.

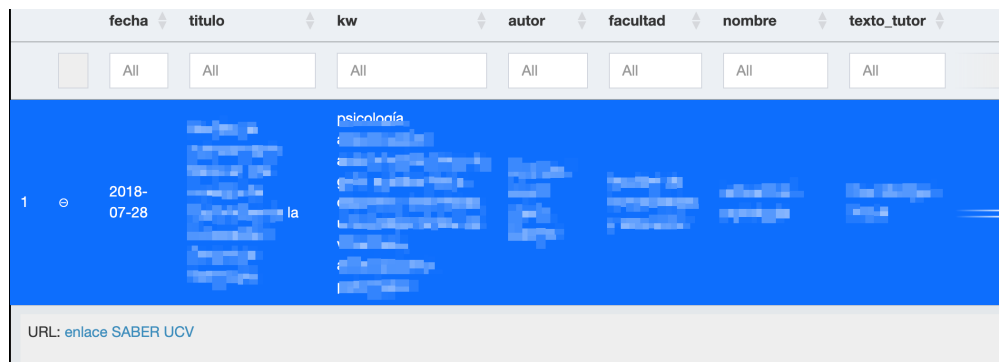


Figura 5.37: Interfaz de Usuario -Inspección Documento





3. **Inspección “mapas de conocimiento”:** la propuesta de visualización del detalle de aparición en documentos de una determinada coocurrencia de dos palabras que se encuentran unidas por un arco se ven en la figura 5.38. La tabla que se muestra aparecerá mediante una ventana desplegable.

5.3.3.0.1.8 Módulos Aplicación Web: La aplicación web, que será desarrollada en el *framework* “Shiny”, según las prácticas de desarrollo recomendadas, se hace bajo la arquitectura de módulos denominados “Shiny Modules”, descomponiendo en partes independientes cada par de funciones relacionadas en la *UI* (define la interfaz de usuario) y el *Server* (define la lógica del servidor), facilitando la comprensión del funcionamiento de la aplicación, la legibilidad del código, pruebas unitarias y el mantenimiento (Wickham, 2021). En la figura 5.39 se puede ver la arquitectura de la aplicación que se integrará dentro del Sistema distribuido.

5.3. CICLOS DE DESARROLLO:

Palabras seleccionadas: almidón - película

buscar:

fecha	título	kw	autor	facultad	nombre
1 2015-03-25	Propiedades físicas, mecánicas y bioensayo con <i>Sitophilus oryzae</i> de películas comestibles de almidón con probióticos, prebióticos y omega-3	películas, coberturas comestibles, protección, alimentos, barrera, agua, gases, vehículo, ingredientes, carácter bioactivo, bioactivo, biológicos, gorgojo de arroz, carácter nutricional	 C.	facultad de ciencias	escuela de biología
2 2015-05-04	Modificación del almidón de yuca (<i>Manihot esculenta</i> Crantz) por metilación para aplicaciones en alimentos y biomedicina.	almidón de yuca, caracterización física, química, funcional, reológica, biocompatibilidad, análisis proximales, fisicoquímicos, dimetil sulfato, empaque orgánico		facultad de ciencias	escuela de biología
3 2015-07-01	Evaluación Fisicoquímica y Nutricional de Almidones y Películas Comestibles a base de Almidón	Películas comestibles, Almidón, alimentos, polímero, Yuca, Maíz, Ocumo chino, Papa, Mapuey, Apio, Almidón resistente, Digestibilidad, Bioensayo del gorgojo de arroz, Análisis de componentes principales, Fibras alimentarias		facultad de ciencias	doctorado en biología celular
4 2017-10-18	Elaboración de coberturas comestibles a partir de almidón de yuca (<i>Manihot esculenta</i> C.) y su aplicación en la conservación de cambur manzano (<i>Musa AAB</i>)	recubrimientos comestibles, conservación, cambur manzano		facultad de ciencias	escuela de biología

Mostrando registros del 1 al 4 de un total de 6 registros

Anterior 1 2 Siguiente

cerrar esta ventana

Figura 5.38: Interfaz de Usuario- Inspección Mapas de Conocimiento

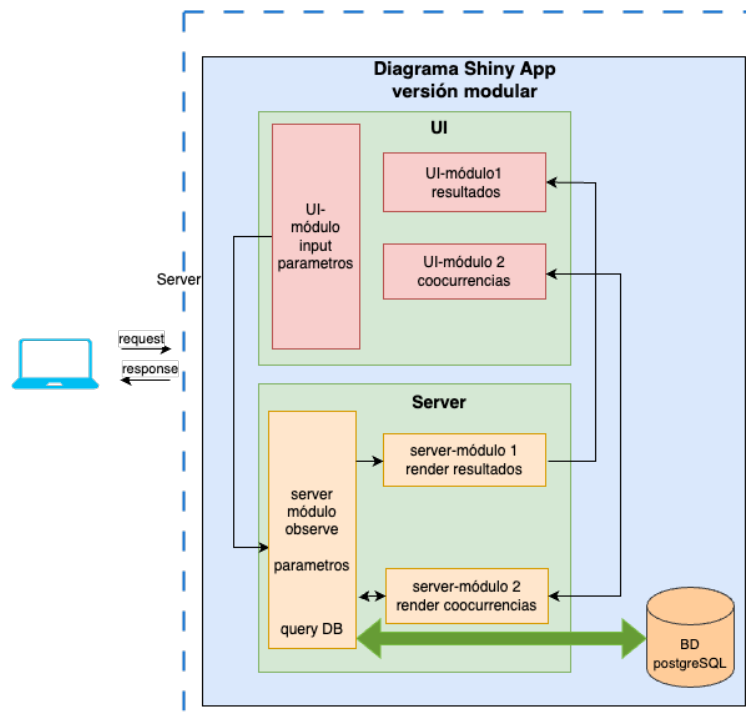


Figura 5.39: Diagrama Aplicación Web Shiny Versión Modular

5.3.3.0.2 Colaboración: Según lo propuesto en la fase de Especulación 5.3.3.0.1 se hizo el desarrollo del Sistema mediante un conjunto de contenedores orquestados, tomando en consideración que es necesario implementar los mecanismos de comunicación necesarios con la definición de una red y de disponer de un volumen compartido con la máquina host.

En la implementación se usaron ampliamente los conocimientos adquiridos en ciclos anteriores, no obstante también se tuvieron que revisar documentaciones, probar distintas opciones de librerías e imágenes de contenedores que permitiesen llevar a un entorno de producción, con la debida integración de los componentes, el Sistema planificado.

5.3.3.0.2.1 Contenedores: Para realizar la implementación del Sistema, apoyado en contenedores para los distintos servicios que lo conforman, se eligió docker como herramienta. Mediante un archivo denominado “dockerfile” para cada servicio se especificó el entorno, dependencias y configuraciones necesarias para crear la imagen que servirá para instanciar cada contenedor.

A continuación se indican los contenedores que forman parte del Sistema y los puertos que disponen para interactuar unos con otros:

1. **Nginx:** Es el servidor web/proxy inverso de código abierto que en este Sistema se usa para redireccionar las peticiones del cliente recibidas en el puerto 80 y 443 al puerto 8080. En este contenedor también se almacena el certificado SSL para permitir conexiones por el protocolo HTTPS en caso de configurar un dominio⁷. Este contenedor fue generado desde una imagen oficial de NGINX que se encuentra en el “docker hub” sin añadir ninguna capa (layer) adicional.
2. **Cerbot:** es una herramienta de código abierto que permite tramitar y habilitar las conexiones mediante el protocolo HTTPS con el uso de un certificado “Let’s Encrypt”. El uso de este certificado está asociado al uso de un dominio en el *deploy* de la aplicación. Este contenedor fue generado desde una imagen de CERBOT del “docker hub” sin realizar ninguna modificación.

⁷Al momento de presentar este Trabajo de Grado no se dispone de un dominio para realizar la configuración del certificado para conexiones HTTPS, no obstante en un momento previo en que se tenía disponible un dominio se hicieron las pruebas para comprobar la integración y correcto funcionamiento.

5.3. CICLOS DE DESARROLLO:

3. **Shinyproxy:** es una solución de código abierto para alojar aplicaciones Shiny (shi, 2023) que cuenta con una tecnología estable y probada. El uso de este contenedor se justifica en el hecho de que una aplicación Shiny funciona como “single thread” y es necesario que ante cada petición de acceso, al servidor despliegue un *workspace* completamente aislado, es decir, un contenedor distinto. Shinyproxy es una implementación del servidor “*Spring boot*” que tiene la capacidad de hacer la replicación indicada y permite controlar los recursos de memoria y cpu asignados, así como el timeout a cada instancia que se despliega. Otras ventajas que aporta, no implementadas en esta versión del Sistema, es que permite establecer *login* en el uso de la aplicación y la creación de grupos de usuarios con perfiles distintos, contando con soporte para distintos métodos de autenticación. Si bien en estos momentos la aplicación está concebida para el libre acceso, en algún momento se puede restringir y no sería necesario hacer modificaciones en la arquitectura, más allá de variaciones en el archivo de configuración. Este contenedor habilita el puerto 8080 para escuchar las peticiones y fue generado desde la imagen del docker hub *Shinyproxy*.
4. **“Shiny Web App”:** En este contenedor se encuentra la imagen de la aplicación web con todas las dependencias y librerías necesarias para remitir los *queries* al contenedor del manejador de la base de datos, presentar los resultados y generar las visualizaciones correspondientes. Como se indicó anteriormente, cada vez que ocurre desde el navegador del cliente una petición de acceso, desde el contenedor *shinyproxy*, se crea una instancia de este contenedor con todos los elementos necesarios para que la app funcione. En caso de presentar alguna falla, el sistema es tolerante a los mismos, porque se pueden seguir recibiendo peticiones que replicarían una imagen nueva del contenedor sin afectar al que presentase el fallo, o viceversa. Desde este contenedor se realiza el acceso de lectura al contenedor que contiene *PostgreSQL* donde reposa la base de datos que contiene los textos ya procesados. La imagen que se usa en este servidor fue definida a medida. Parte de las configuraciones de la interfaz de usuario se realizan en un archivo *css* para facilitar modificaciones que puedan resultar de interés.

Lógica de la aplicación:

El esquema general de la aplicación es similar al que se presentó en el Prototipo en la figura 5.20. A continuación se mencionan las entradas y

salidas que se generan dentro del contenedor.

1. Entradas:

1. **Defición query:** Contiene un campo para la entrada de texto con el que se generará el query.- Contiene un selector para indicar si se quiere generar la coocurrencia de palabras- Contiene tablas para seleccionar: 1) Nivel académico del trabajo. Opciones (pregrado, especialización, maestría, doctorado). 2) Facultad o Centro de adscripción. Opciones: 11 Facultades más un centro (CENDES). 3) Nombre del pregrado o postgrado. En total son 412 las opciones. Cada una de las tablas anteriores se actualiza según se vayan seleccionando las relaciones y la disponibilidades. Por ejemplo, al seleccionar pregrado solo se mostrarán los nombres de las carreras de pregrado, pero si se selecciona también el nombre de la Facultad, sólo se mostrarán las carreras de pregrado dentro de la Facultad seleccionada. Para una determinado filtro se permiten selecciones múltiples dando una total flexibilidad al momento de ejecutar los *queries*. El texto del *query* es procesado convirtiéndolo a minúsculas y removiendo signos de puntuación.
2. **Selector “generar coocurrencias”:** es una casilla que sirve para seleccionar si se van a representar los Mapas de Conocimiento o no. Por defecto está deselccionado ya que la representación de los mapas incrementa el tiempo de procesamiento de los datos.
3. **“Clic” inspeccionar documento:** al obtener los resultados en una tabla se puede inspeccionar un documento particular para ver el texto resumen y las recomendaciones de documentos similares.
4. **Cantidad de coocurrencias:** teniendo como condición que previamente se seleccionara “generar coocurrencias”, dentro de la pestaña de “Coocurrencias”, correspondientes a los Mapas de Conocimiento, se puede modificar la cantidad de coocurrencias a representar. El valor por defecto representado es 60.
5. **“Clic” inspeccionar Mapa de Conocimiento:** dentro de la pestaña de “Coocurrencias”, correspondientes a los Mapas de Conocimiento, al tener representadas dos palabras mediante nodos, existe un arco que las une. Sobre el arco se puede hacer clic e inspeccionar los documentos que contienen las palabras asociadas por el arco.

2. **Salidas:** Ante el *query* se genera:

1. **Resultados query:** en una tabla creada con la librería “DT” se muestra el listado de documentos recuperados mostrando en cada fila los siguientes atributos: autor, fecha, palabras clave, texto resumen, nombre tutor. Los documentos a presentar y el orden en que son presentados viene desde el contenedor “PostgreSQL”. Adicionalmente se muestra un enlace al repositorio Saber UCV donde se encuentra alojado el respectivo trabajo (el documento en PDF). Igualmente se presentan los textos que tienen mayor similitud con el documento seleccionado. Un gráfico con la frecuencia por año de los trabajos extraídos mediante el *query*. El gráfico se generará con la librería “apexcharter”, por lo cual tiene ciertas interactividades mostrando con el *hover* el valor de la cantidad por año que está representada en cada columna.
2. **Mapas de Conocimiento dinámico:** Gráfico de coocurrencia interactivo de palabras que se genera mediante la librería de “VisNetwork” . Este gráfico permite distintas interacciones con el usuario como hacer zoom (in-out), seleccionar un determinado nodo, usar las teclas izquierda, derecha, arriba y abajo del teclado, así como también permite seleccionar un arco de unión entre dos palabras coocurrentes. Al realizar la selección se filtran un subconjunto de los documentos que contienen ambas palabras representadas por nodos. Los documentos filtrados se mostrarán en una tabla contigua, también generada en “DT”, donde sólo se incluye el texto resumen de cada trabajo.
3. **Mapas de Conocimiento Estático:** gráfico de coocurrencias estático de palabras, donde mediante la librería “ggraph” son generados un par de gráficos con distintas granularidades. El primero exhibe la misma coocurrencia de palabras expuesta en el punto con una visualización estática. En cuanto a la granularidad se muestran las palabras que coocurren dentro de todo el resumen independientemente de la proximidad que tengan. El segundo gráfico también muestra la coocurrencia, pero solo de palabras que se encuentran en el texto resumen una seguida de otra representando los resultados con una menor granularidad. Con la librería “UdPipe” se generan las estructuras de datos necesarias para generar los grafos

(arcos y nodos).

En el Apéndice 7 se listan los paquetes que usa este contenedor.

5. **“PostgreSQL”**: es el contenedor donde se encuentra el manejador de PostgreSQL versión 16 y en él se almacenan todas las tablas que usa el Sistema. También recae la funcionalidad de generar el “ts_vector”, convertir el query a un “ts_query” y mediante la función “ts_rank” reordenar los resultados extraídos con base en un ranking junto con el indexado de las distintas tablas. En el ciclo “Iteración- Implementación Prototipo” 5.3.2 se indican más detalles sobre las funciones señaladas. En este contenedor se tiene una imagen de *PostgreSQL* versión 16.1 extraída del “docker hub” a la cual no le fue realizada ninguna modificación distinta a la configuración para la definición de usuarios y contraseña junto con la definición de un volumen compartido con el *host* para garantizar que se tengan “datos persistentes”. Este contenedor recibe consultas del contenedor “*Shiny Web App*” y escritura-lectura desde el contenedor “R imagen Servicios” teniendo habilitado el puerto 5432.
6. **“R Imagen Servicios”**: En este contenedor se creó una imagen con todos los servicios necesarios para realizar el *web crawling*, el procesamiento de textos y la descarga de los archivos desde Saber UCV para realizar la clasificación de las Tesis y demás trabajos. Al iniciar la configuración del Sistema, contiene las funcionalidades que permiten realizar la creación de la base de datos, las tablas y el poblado de estas. Periódicamente es invocado un script mediante un “cron job” para realizar los procesos de incorporación de aquellos documentos nuevos que se detecte que están disponibles en Saber UCV. La imagen base que se usa es la del proyecto Rocker (Boettiger and Eddelbuettel, 2017), la cual es una versión ampliamente probada y optimizada por la comunidad de usuarios de R.

Procesos:

1. **Poblado base de datos**: se ejecutan los procesos para hacer el poblado inicial de base de dato así como a la creación del indexado de la base de datos en el contenedor “**PostgreSQL**”. En la fase de especulación de este ciclo se presentó un diagrama que describe el esquema de “extracción y clasificación de los datos”, ver figura 5.31.

5.3. CICLOS DE DESARROLLO:

2. **Descarga de datos:** proceso que fue abordado en el “Ciclo Conformación del Conjunto de Datos”.
3. **Text Mining y NLP:** en el “Ciclo Prototipo SCSU”, ver 5.3.2, en la iteración “Preparación del Corpus”, ver 5.3.2.1, se detallan los procesamientos a los textos que ahora son ejecutados en este contenedor. La diferencia es que para usar “spacyr”, al depender esta librería de una ambiente virtual en python , es necesario configurar otro contenedor con las dependencias y librerías que permitan el llamado al etiquetado del discurso. El contenedor que se integra para realizar estos procesos es “**Python Spacy**”.
4. **Generación de recomendaciones:** se corresponde con detallado en el “Ciclo Prototipo de SCSU” en la iteración “Recomendación de Documentos”, ver 5.3.2.2.

En el Apéndice 7 se listan las librerías que usa este contenedor

7. **“Python Spacy”:** Se creó una imagen que contiene un “Ubuntu 22” con “python 3.10”, la librería “spacy v.3” y el modelo de Spacy “es_core_news_lg” . Su función es que mediante un volumen compartido pueda ser invocado desde el contenedor “R Imagen Servicios” para así realizar los procesamientos de NLP descritos.

5.3.3.0.2.2 Orquestador: Todos los contenedores mencionados es necesario que actúen de forma coordinada, compartiendo recursos como una red y volúmenes de almacenamiento. Para lograr esto se acude a la herramienta “Docker Compose” que se instala en el *host* y permite simplificar y automatizar el despliegue de aplicaciones compuestas por múltiples servicios y contenedores en entornos de desarrollo y producción.

Se define la infraestructura, configuración y gestión de una aplicación a través de un archivo YAML, denominado “docker-compose.yml” donde se detallan los servicios, contenedores, redes y volúmenes necesarios para la aplicación, así como las relaciones y configuraciones entre ellos, simplificando la orquestación de los contenedores.

También la adopción de Docker Compose facilita la replicación del entorno de desarrollo en diferentes máquinas, mejorando la consistencia en el desarrollo.

La responsabilidad del administrador queda reflejada en la definición de distintas variables de entorno contenidas en el archivo docker compose, como lo es la definición del período de actualización de los textos.

5.3.3.0.2.3 Mantenimiento del SCSU: De acuerdo a la implementación del Sistema, los procesos de mantenimiento que corresponde ejecutar al administrador, son los que se detallan a continuación:

1. Respaldos regulares de la base de datos:

- Se estableció un “*cron job*” en el contenedor “**R Imagen Servicios**” para realizar respaldos automáticos de la base de datos PostgreSQL.

2. Monitoreo del rendimiento:

- Se utilizan herramientas de monitoreo para vigilar el rendimiento del contenedor PostgreSQL, Shinyproxy y Shiny.

3. Logs y registro de eventos:

- Registro de logs de contenedores en un volumen compartido con el *host*.

4. Gestión de dependencias:

- Se documentan las dependencias específicas de la versión de PostgreSQL, Shinyproxy y Shiny.

5. Seguridad de red:

- Se configuró el *firewall* y las reglas de red para limitar el acceso no autorizado a los contenedores.

6. Escalabilidad:

- Se realizaron pruebas de carga para identificar posibles cuellos de botella.

Eventualmente en caso del Sistema quedar en un entorno de Producción permanente, sería necesario agregar los siguientes procesos al Mantenimiento:

5.3. CICLOS DE DESARROLLO:

1. Actualizaciones de seguridad:

- Supervisar actualizaciones de seguridad para PostgreSQL y la imagen de Shiny.

2. Control de versiones:

- Utilizar un sistema de control de versiones para el código del SCSU.

3. Documentación:

- Mantener actualizada la documentación del sistema, incluyendo la configuración, dependencias y procedimientos de recuperación ante desastres.

5.3.3.0.2.4 Requerimientos mínimos de hardware: Para un concurrencia de 4 usuarios simultáneos, estos son los requerimientos de hardware:

- 2 CPU virtual
- 4 GB de memoria RAM
- 50 GB de disco duro

5.3.3.0.3 Aprender: Este Ciclo trajo un profundo y largo proceso de aprendizaje ya que el desarrollo implicó integrar distintos elementos abordados previamente de forma aislada, más el reto de encontrar componentes que fuesen compatibles y adaptados a los requerimientos funcionales y no funcionales planteados.

La utilidad de esta integración también representa en que se cuenta con un *framework* que permite desplegar otro tipo de aplicaciones que pueden ser fácilmente escalables en entornos de producción o añadir módulos distintos al SCSU como pudiera ser una aplicación para la administración del sistema o consultar estadísticas de uso.

Como la cantidad de ciclos e iteraciones ejecutados previamente había sido extenso, en esta Ciclo, más allá del reto de la integración y evaluación de componentes, no fue de distinta índole el aprendizaje adquirido, sin embargo se decidió incorporar en la interfaz del usuario un gráfico que muestre la frecuencia de aparición de *query* en el tiempo así como una representación estática de los

“mapas de conocimiento” ya que en esta versión se presenta con una mejor perspectiva la coocurrencia de términos, a diferencia de la versión interactiva que está diseñada para inspeccionar en detalle una determinada aparición de términos.

5.3.3.0.3.1 Objetivos alcanzados:

- Se implementó el Sistema que cumple con el Objetivo General 2.4.1.
- Se dispone de una versión del Sistema que es reproducible.
- Se integraron con éxito los distintos componentes del Sistema.

5.3. CICLOS DE DESARROLLO:

5.3.4 Ciclo Incorporación de otras investigaciones:

En este ciclo se aborda la incorporación al “Sistema Complementario Saber UCV” publicaciones distintas a las investigaciones de pregrado y postgrado de la Universidad Central de Venezuela.

Lo propuesto en este Ciclo únicamente tiene como finalidad evaluar si el SCSU se puede usar para alojar revistas de investigación producidas dentro de la Universidad Central de Venezuela o fuera de ella.

5.3.4.0.1 Especulación: Para los centros de estudio que generen investigaciones que sean alojadas en un repositorio de datos y se cumpla la estructura donde se disponga de un título para la publicación, una fecha, un autor, palabras claves (opcional) y un texto, es posible replicar los métodos descritos anteriormente para obtener los datos e incorporarlos al SCSU.

Las publicaciones que se incorporarán son:

1. Archivo histórico de la Revista “**Gestión I+D**” editada dentro de la Universidad Central de Venezuela por el Postgrado en Gestión de Investigación y Desarrollo de la Facultad de Ciencias Económicas y Sociales.
2. Archivo histórico de la Revista “**Episteme NS**” editada dentro de la Universidad Central de Venezuela por el Instituto de Filosofía de la Facultad de Humanidades y Educación.
3. Archivo histórico de la Revista “**Observador del Conocimiento**” editada por el Observatorio Nacional de Ciencia, Tecnología e Innovación, adscrito al Ministerio del Poder Popular para Ciencia y Tecnología.
4. Artículo “**El Dorado Revisitado**” del Boletín Atropológico editado por la Universidad de los Andes.

5.3.4.0.2 Colaboración: Siguiendo las técnicas descritas en “Iteración-”Extracción de Datos web Saber UCV” 5.3.1.1, se hizo la descarga de los datos y se introdujeron a la base de datos sin alterar la estructura que se había propuesto en el “Ciclo de

Integración de los Componentes” 5.3.3 lo que motiva a que no se entre en detalles sobre los métodos aplicados para incorporar este nuevo lote de investigaciones.

Luego de hacer la descarga y procesamiento de los datos se obtuvieron la siguiente cantidad de artículos por Revista:

1. Gestión I+D (UCV): 129
2. Revista Episteme (UCV): 68
3. Revista Observador del Conocimiento (ONCTI): 197
4. Boletín Antropológico (ULA): 1

5.3.4.0.3 Aprender: En caso de querer expandir el Sistema a otro tipo de publicaciones o universidades puede resultar de utilidad crear un módulo para la incorporación de estas donde se determine previamente la estructura de las etiquetas *css* o *html* , y posteriormente ser realice el scrapy y la asignación de categorías, no obstante no con este método no se resolvería, en caso de ser necesario, el proceso de tener que realizar clasificaciones por área de conocimiento como se hizo *ad hoc* para Saber UCV.

En los históricos de algunas publicaciones existen atributos que no cumplen con las etiquetas *css* identificadas para hacer la descarga y se presentan algunas fallas, no obstante es para un mínimo de artículos y se considera que sí es viable realizar las incorporaciones bajo el método propuesto.

5.3.5 Ciclo Buscador Semántico:

Si bien en el “Ciclo de Integración de Componentes de Software” se cumplió con el objetivo general propuesto en esta investigación, en este ciclo se evalúa que el Sistema incorpore la búsqueda semántica 3.5.5.

Los procesos incorporados en este Ciclo se hacen con fines experimentales y no serán sometidos a las distintas pruebas que se efectuarán más adelante, ni tampoco se harán los procesos de ingeniería de software para detallar funcionalidades o modificaciones al “diagrama entidad relación”, sino se expondrá el método adoptado para añadir al Sistema este tipo de búsqueda que para esta fecha constituye el estado del arte en los sistemas de recuperación de información.

5.3.5.0.1 Especulación: La búsqueda semántica trata de mejorar la precisión de la búsqueda entendiendo el contenido de la consulta. A diferencia de los motores de búsqueda tradicionales, que sólo encuentran documentos a partir de coincidencias léxicas, la búsqueda semántica también puede encontrar sinónimos.

Adicionalmente se puede hacer el *reranking* con modelos de *machine learning* entrenados para tales fines (Gökçe et al., 2020), (Nogueira and Cho, 2019) pudiendo mejorar el ordenamiento con criterios de relevancia distintos a los vistos en el creado por la función “ts_rank” de postgresQL en 5.3.2.3.

Para incorporar estas funcionalidades se propone crear una “API” que sea implementada mediante el microframework “FastAPI” donde se permitirán recibir peticiones para:

1. **Registrar Embedding:** Con el texto resumen de cada investigación, convertirlo en distintos trozos de texto, generar embeddings para cada trozo y que estos sean almacenados en una tabla de *embeddings* dentro de la base de datos, asociando el código de identificación del documento.
2. **Consultar:** Recibir el texto del query, convertirlo en un *embedding*, buscar los trozos de texto que presenten mayor similitud coseno, recuperar los identificadores de esos documentos, hacer un proceso de *reranking* con un modelo de *machine learning* y dar como resultado los identificadores de los veinte documentos más relevantes.

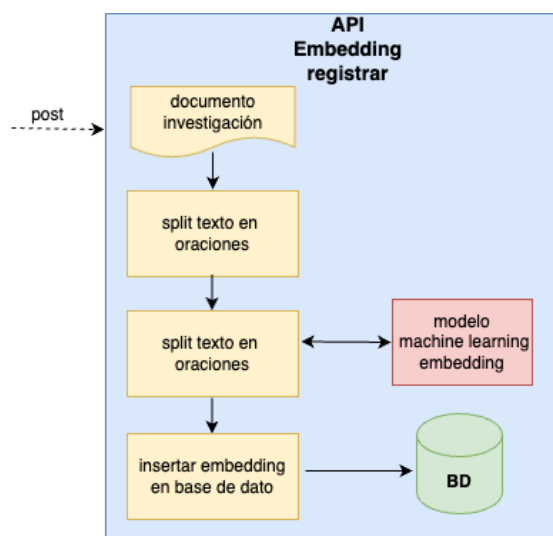


Figura 5.40: Diagrama API- Registrar Embedding

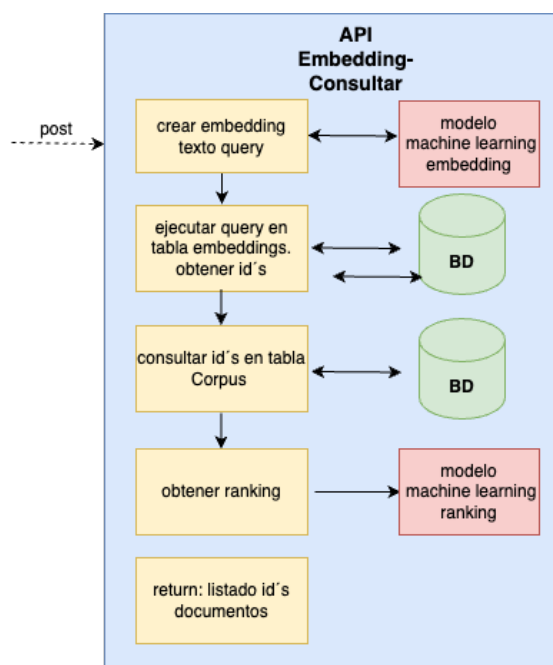


Figura 5.41: Diagrama API- Consultar Embedding

5.3. CICLOS DE DESARROLLO:

5.3.5.0.2 Colaboración: La API es una implementación realizada con el lenguaje de programación Python mediante el *microframework* “FastAPI”. Como ya se había implementado una versión orquestada en contenedores se añadió como un contenedor adicional este componente.

Se hizo la evaluación de varios modelos de aprendizaje automático para generar los *embeddings* los cuales se encuentran de libre acceso en el repositorio “Hugging Face” (hfm, 2023).

Para realizar el proceso del “splitting” del texto resumen se usaron distintos modelos preentrenados para segmentar el texto acorde a las distintas ideas representadas, no necesariamente basándose en signos de puntuación. Dentro de los modelos revisados está “What it’s the point” (Minixhofer et al., 2023) , “Text to Sentence Splitter” (sen, 2018) seleccionando este último. Este proceso es necesario ejecutarlo porque se busca segmentar el texto en ideas, que queden registradas en *embeddings* para que en el momento de hacer el *query* se busque el trozo de texto que presente una mayor similitud con el *embedding* generado a partir del *query*.

Para poder registrar los vectores de *embeddings* y realizar la búsqueda por similaridad en el gestor de base de datos PostgreSQL fue necesario añadir la extensión “pgvector” (pgv, 2023).

El modelo para crear los *embeddings* es el “intfloat/multilingual-e5-base” (Wang et al., 2022), alojado en el repositorio de hugging face (e5b, 2022), el cual limita la cantidad de tokens a 512, equivalente a 360 palabras por *embedding* y tiene un tamaño de 768 atributos. Previo a ser seleccionado se hizo una evaluación de otros modelos que tienen la misma funcionalidad como “dccuchile/bert-base-spanish-wwm-cased” (Canete et al., 2020). El modelo “e5-base” presenta la mejor relación entre cantidad de descargas y peso del modelo. Teniendo presente que la cantidad de recursos computacionales que se disponen para hacer el *deploy* de todo el Sistema es limitado se decidió usar este modelo que ocupa aproximadamente 1 GB para estar en producción.

Para hacer los proceso de *reranking* se evaluaron dos modelos que son el “ameroad/bert-multilingual-passage-reranking-msmarco” (era, 2022) y el “IIC/roberta-base-bne-ranker” (hfr, 2023), siendo seleccionado segundo por presentar un menor peso, tener mayor cantidad de descargas y ayudar a minimizar el consumo de recurso dentro del Sistema.

Es necesario señalar que los modelos de machine learning evaluados y seleccionados están entrenados para funcionar con textos en el idioma español.

En la aplicación Shiny se hicieron las modificaciones para hacer el llamado a la API y obtener de ella el listado de id's de documentos seleccionados y posteriormente añadirlo a la lista de documentos recuperados mediante el método de índice invertido. Con la finalidad de evitar que se dupliquen documentos que sean obtenidos mediante el índice invertido y la búsqueda semántica se procedió a remover los documentos repetidos. Igualmente en la interfaz de usuario en la tabla que muestra los resultados se añadió una columna para señalar el método con el que fue recuperado el documento.

5.3.5.0.3 Aprender: Usar la “búsqueda semántica” amplia considerablemente las posibilidades de obtener documentos que resulten de interés para el investigador, ya que no sólo se queda circunscrita la recuperación de información a términos exactos, sino se extraen documentos que pueden estar altamente relacionados al tema de búsqueda. En la figura se presentan los resultados ante el texto “problemas que enfrentan las mujeres”.

1	2015-01-29	Rasgos de personalidad en sujetos tatuados	tatuajes, rasgos de personalidad, piel, tattoos, personality traits, skin.	Yendys Martínez, Osmay del Valle	facultad de medicina	especialización en psiquiatría	Tamara Gonzalez Osmay	semantic search
2	2013-08-06	El proceso de integración Mercosur y los Derechos Humanos	el proceso de integración mercado común del sur, mercosur, democracia, the southern common market, democracy, transition of human rights	Alvarado, Marino	facultad de ciencias jurídicas y políticas	especialización en derecho internacional económico y de la integración	Raul Arrieta	semantic search
3	2012-09-24	Relaciones de objeto y codependencia en madres de sujetos adictos a las drogas	relaciones de objeto, codependencia, relación madre - hijo, psicodiagnóstico de roschach, adicciones y drogas	Moncada, Libertad	facultad de humanidades y educación	maestría en información y comunicación para el desarrollo	sin información	semantic search
4	2023-09-12	Mujeres: Desigualdades presentes en diferentes espacios		Glenda Yamileth Trejo-Magaña	Gestión I+D	julio - diciembre - 2019		semantic search
5	2016-12-01	Ansiedad y Depresión en Mujeres infértiles	ansiedad, depresión, infertilidad, anxiety, depression, infertility	Martínez Paredes, Carla Eugenia Rangel Ibarra, Venus Vianey	facultad de medicina	especialización en obstetricia y ginecología	Simon Pineda	semantic search
6	2019-01-07	Relaciones objetales y estrategias de afrontamiento en mujeres en situación de violencia de pareja.	psicología, psicología clínica dinámica, relaciones objetales, estrategias de afrontamiento, violencia de pareja	Castillo, Victor Hermoso, Kimberlyn	facultad de humanidades y educación	escuela de psicología	sin información	semantic search
7	2016-03-28	Centralidad y flujo de información en el control de redes complejas	redes, redes complejas, estrategias de control	Vieira, Raquel	facultad de ciencias	escuela de física	Pedro García	semantic search

Figura 5.42: Resultados de Búsqueda Semántica

En este query se extrajeron 25 documentos y sino se hubiese usado esta técnica hubiesen ascendido a solo 5 los documentos extraídos.

5.3. CICLOS DE DESARROLLO:

Un elemento a tener en cuenta es que la búsqueda semántica y la ejecución del reranking, incrementa considerablemente el tiempo de ejecución del *query*, no obstante, es necesario señalar que el método que implementa PostgreSQL para hacer la comparación vectorial no es parte del estado del arte en la materia.

El uso de PostgreSQL para hacer el proceso se hizo para simplificar la instalación de componentes adicionales dentro del Sistema, pero en caso de ser adoptada la búsqueda semántica en otra versión del SCSU, es necesario encontrar métodos que proporcionen un mejor desempeño en lo referente a los tiempos de ejecución del *query*.

En esta implementación se evaluaron algunos modelos para crear los *embeddings* sin entrar en consideración sobre cuáles presentan mejores resultados al evaluar la relevancia de los documentos extraídos. Este es un punto en que en futuras investigaciones es necesario ahondar para comparar cuáles pueden ser de mayor utilidad.

Igualmente es necesario en futuros trabajos establecer un umbral de similitud o cantidad de documentos a recuperar, óptimo, ya que al no hacerlo y simplemente reordenar los documentos por similitud, se extraerá todo el Corpus y se afectará directamente las métricas de precisión y exactitud (recall).

5.4 Pruebas de aceptación:

La fase de “Pruebas de Aceptación” en el desarrollo del Sistema Complementario Saber UCV representa el punto culminante en los ciclos de desarrollo abordados ya que se mide si las expectativas y necesidades del usuario final fueron satisfechas con la solución propuesta. Estas pruebas verifican la conformidad del sistema con los requisitos previamente establecidos y evalúa la capacidad para satisfacer las demandas del entorno operativo.

En esta sección, se muestran los resultados de las distintas pruebas que fueron ejecutadas, diseñadas para validar no solo la funcionalidad técnica de la aplicación, sino también su capacidad para integrarse al contexto de uso previsto.

5.4.1 Funcionales:

Las Pruebas Funcionales constituyen la columna vertebral en la validación del software, evaluando su comportamiento según los requisitos que fueron especificados.

En la tabla 5.13 se muestran las pruebas de caja negra realizadas para evaluar si el Sistema se comporta según lo que fue definido para cada caso de uso sin evaluar los procesos, sino en de una forma concreta si la salida creada por el Sistema se corresponde a lo descrito en los Casos de Uso del “Ciclo Integración de Componentes de Software” 5.3.3. En la tabla 5.14 se muestran las pruebas de caja negra realizadas para evaluar si el Sistema cumple con todos los requerimientos funcionales también definidos en Ciclo precitado.

5.4. PRUEBAS DE ACEPTACIÓN:

Cuadro 5.13: Pruebas de Caja Negra: Casos de Uso

Caso de Uso	Se realiza el comportamiento esperado
UC.1: Realizar proceso de recuperación de información (query)	verdad
UC.1. Nivel 2: Realizar proceso de recuperación de información (query) aplicando filtros	verdad
UC.2: Realizar Inspección de Recomendaciones	verdad
UC.3: Realizar inspección de "Mapas de Conocimiento"	verdad
UC.3. Nivel 2.1: Realizar inspección "Mapa de Conocimiento"	verdad
UC.3. Nivel 2.2: modificar cantidad de coocurrencias en "Mapa de Conocimiento"	verdad

Cuadro 5.14: Pruebas de Caja Negra:
Requerimientos Funcionales

Requerimiento Funcional	Se realiza el comportamiento esperado
UC.1: Realizar proceso de recuperación de información (query)	verdad
UC.1. Nivel 2: Realizar proceso de recuperación de información (query) aplicando filtros	verdad
UC.2: Realizar Inspección de Recomendaciones	verdad
UC.3: Realizar inspección de "Mapas de Conocimiento"	verdad
UC.3. Nivel 2.1: Realizar inspección "Mapa de Conocimiento"	verdad
UC.3. Nivel 2.2: modificar cantidad de coocurrencias en "Mapa de Conocimiento"	verdad

5.4.2 Rendimiento:

La primera prueba de rendimiento que se realizó fue mediante la librería Shinytest (Chang et al., 2023c), la cual está diseñada para facilitar a los desarrolladores verificar el comportamiento de sus aplicaciones de manera sistemática y reproducible mediante pruebas automatizadas en aplicaciones Shiny. Las pruebas se hacen con un driver que simula la interacción del usuario con el sistema definiendo previamente cuáles son los comportamientos que se quieren simular. Igualmente se define en qué momento se quiere tomar una "snapshot" para posteriormente evaluar si se adapta al comportamiento esperado.

Concretamente se evaluó en esta prueba si al hacer una simulación de *query* con cada uno de los textos de los títulos de las investigaciones, se generaba al menos la extracción de un documento, lo cual demostraría que todos los documentos que

5.4. PRUEBAS DE ACEPTACIÓN:

conforman el Corpus son extraídos. El resultado obtenido fue el esperado con al menos un documento por *query*.

La segunda prueba aplicada fue la que se ejecuta con el paquete “Shinyloadtest” (Schloerke et al., 2021) el cual es un paquete que permite evaluar el rendimiento y la escalabilidad de aplicaciones Shiny ya que permite simular múltiples usuarios interactuando simultáneamente con una aplicación y así evaluar el comportamiento bajo carga. Adicionalmente este paquete genera informes detallados sobre el rendimiento, identificando posibles cuellos de botella y proporcionando métricas clave, como tiempos de respuesta y tasas de error.

Al realizar el shinytest no se detectaron cuellos de botella ni tiempos de carga que estuviesen fuera de lo esperado.

Pruebas de usabilidad: mencionar disponibilidad de menús de ayuda y pantalla de bienvenida

5.4.3 Relevancia:

Para hacer las mediciones de Relevancia se adoptó el siguiente método:

1. Definir Conjunto de Datos de Prueba:

- Se seleccionaron aleatoriamente 50 documentos de trabajos de pregrado y postgrado asociados a la Escuela de Computación y el postgrado en Ciencias de la Computación. Se denomina conjunto A.
- Se seleccionaron aleatoriamente 50 documentos de trabajos de pregrado y postgrado de documentos que no pertenezcan a la Escuela de Computación ni al postgrado en Ciencias de la Computación. Se denomina conjunto B.

2. Crear Consultas de Prueba:

- Se diseñaron 10 consultas con términos asociados a los documentos del conjunto A.
- Se diseñaron 10 consultas con términos asociados a los documentos del conjunto B.

3. Ejecutar Consultas en el Sistema:

- Se hicieron las consultas tanto para el conjunto A como para el B y se guardaron los resultados junto con el término del query.

4. Definir Relevancia Esperada por expertos:

- A un grupo de expertos sobre el conjunto de documentos A se les pidió que indicaran para cada consulta, de las 10 definidas, cuáles son los 5 (o un número menor) documentos más relevantes sin tomar en cuenta el ranking.

5. Recopilar Resultados:

- Para cada consulta evaluada por un experto se comparó si el Sistema introducía dentro de los 10 primeros lugares los cinco (o menos) documentos señalados por el experto. En esta evaluación no se toma en consideración el orden ni del sistema ni de los expertos.

6. Calcular Métricas de Evaluación:

- Con los resultados se calcularon las métricas *precisión* y *recall* y F1 ya que ellas son claves para la evaluación de sistemas de recuperación de información.
 - **Precisión (Precision):** Número de documentos relevantes recuperados dividido por el número total de documentos recuperados.
 - **Recall:** Número de documentos relevantes recuperados dividido por el número total de documentos relevantes en la colección.

Los resultados obtenidos fueron:

- **Precisión:**
- **Recall:**
- **F1:**

Con esta prueba se da por concluido el Desarrollo de la Solución.

Capítulo 6

Conclusiones:

Este es el capítulo de las Conclusiones

Repaso por el problema que representó la generación del Corpus

Implementación del Sistema Complementario Saber

6.1 Contribución:

Contar con un sistema que permite clasificar los trabajos alojados en el repositorio Saber UCV mientras no se solucione de raíz la asignación de las investigaciones a su área de conocimiento en el momento en que son registradas dentro del sistema por las unidades responsables. Otra contribución que representa la investigación realizada es que se extraen los nombres de los tutores con lo cual se permite tener noción de cuáles son las áreas de investigaciones que manejan los profesores de la Universidad Central de Venezuela.

El Sistema Complementario Saber UCV puede servir de base para el desarrollo de softwares que permitan incorporar métodos distintos para la recuperación y representación de los resultados obtenidos.

Usar los *embeddings* dentro del Sistema abre las puertas para poder realizar complementos y mejoras al proceso de Recuperación de Información y en concreto expande la posibilidades con que contarán los investigadores que hagan búsquedas sobre el Corpus conformado.

6.2 Trabajos Futuros:

La investigación que se presentó en este Trabajo de Grado deja grandes inquietudes para inspeccionar nuevos métodos que ayudan a la gestión de la investigación. A continuación se muestran algunas de las áreas sobre las cuales se puede ahondar en Trabajos Futuros:

1. Incorporar otros saberes (Universidad de los Andes, Universidad de Carabobo, Universidad Católica) o repositorios en la base de dato de un sistema similar al propuesto, para poder contar con un repositorio donde se integren las distintas investigaciones que son realizadas a nivel nacional.
2. Distribuir mediante técnicas de “map reduce” los datos en un sistema distribuido y paralelo para evaluar métodos que permitan mejorar los tiempos de búsqueda, sobre todo en caso de que se contase con más documentos.
3. Mediante el uso de embeddings:
 1. Integrar largos modelos de lenguaje para crear un sistema de “preguntas y respuestas” sobre los documentos alojados en el Sistema.
 2. Evaluar distintos modelos para realizar el *rerank* a todos los resultados obtenidos.
 3. Evaluar distintos modelos de generación de *embeddings* para evaluar el comportamiento en el proceso de recuperación de información, incluso usando modelos que están entrenados sobre áreas de conocimiento específicas como lo es “SciBERT” (Beltagy et al., 2019).
 4. Usar distintos métodos para medición de similitud, como el de vecinos cercanos (KNN) (Bernhardsson, 2023) o de máquinas de soporte vectorial (SVM) (Karpathy, 2023).

Capítulo 7

Apéndice: Librerías usadas y créditos

1. Las librerías de R que usa la aplicación web son :

1. **shinydashboard**: sirve para gener un formato de *dashboard* en la aplicación (Chang and Borges Ribeiro, 2021).
2. **shinydashboardPlus**: sirve para asistir a “shinydashboard” en configuraciones y funcionalidades adicionales (Granjon, 2021).
3. **shinycssloaders**: muestra un *loader* cuando la aplicación de está cargando (Sali and Attali, 2020).
4. **pool**: gestor de conexión mediante un *pool* a la base de datos (Cheng et al., 2023).
5. **RPostgres**: carga el protocolo de conexión a postgresSQL (Wickham et al., 2023c).
6. **DBI**: permite la definición en R de *querys* en lenguaje SQL (et al., 2022).
7. **dplyr**: es una herramienta que facilita la manipulación de las *dataframes* que sirven de insumo para representar los resultados (Wickham et al., 2023a).
8. **tidyr**: manipulación de la *dataframe* que contiene los resultados (Wickham et al., 2023d).
9. **dbplyr**: bajo la sintáxis de “dplyr” permite estructurar los querys (Wickham et al., 2023b).
10. **DT**: *wrapper* de la librería “DataTables” de Javascript que permite generar tablas interactivas en html que muestran los resultados de los querys (Xie et al., 2023b).

11. **udpipe**: paquete que permite manejar textos bajo los principios del *framework* “Universal Dependencies” y en la aplicación se usa para generar la estructura de datos que representa la coocurrencias (Wijffels, 2023c).
 12. **ggraph**: permite realizar la visualización estática mediante grafos de las coocurrencias (Pedersen, 2022b).
 13. **visNetwork**: permite realizar la visualización dinámica mediante grafos de las coocurrencias. Es un *wrapper* de la librería Vis.js.
 14. **apexcharter**: permite generar gráficos interactivos y es un *wrapper* de la librería “ApexCharts.js” (Perrier and Meyer, 2023).
 15. **scales**: se usa para reescalar los arcos que unen los nodos en las coocurrencias (Wickham and Seidel, 2022).
 16. **fontawesome**: se usa para mostrar íconos en los selectores y ventanas (Iannone, 2023).
 17. **stringr**: manipulación y manejo de textos (Wickham, 2022c).
 18. **shinyBS**: permite añadir los *tooltips* (Bailey, 2022).
 19. **shinyalert**: crea un *popup* al ingresar a la aplicación.
 20. **shinyWidgets**: herramienta para manejar interactividades (Perrier et al., 2023).
 21. **config**: configurar variables globales (Allaire, 2020).
2. Librerías usadas por el Contenedor “R Servicios”:
1. **httr2**: ejecutar consultas HTTP y procesar las respuestas (Wickham, 2023).
 2. **pool**: gestor de conexión mediante un *pool* a la base de datos (Cheng et al., 2023).
 3. **RPostgres**: carga el protocolo de conexión a postgresSQL (Wickham et al., 2023c).
 4. **DBI**: permite la definición en R de *queries* en lenguaje SQL (et al., 2022).
 5. **odbc**: soporte para conexiones a manejadores de base de datos (Hester et al., 2023).

6. **dplyr**: es una herramienta que facilita la manipulación de las *dataframes* que sirven de insumo para representar los resultados (Wickham et al., 2023a).
 7. **tidyr**: manipulación de la *dataframe* que contiene los resultados (Wickham et al., 2023d).
 8. **dbplyr**: bajo la sintaxis de “dplyr” permite estructurar los queries (Wickham et al., 2023b).
 9. **stringr**: manipulación y manejo de textos (Wickham, 2022c).
 10. **rvest**: manipulación y extracción de datos de páginas web (Wickham, 2022b).
 11. **rlang**: caja de herramientas para trabajar con estructuras de R base (Henry and Wickham, 2023).
 12. **purrr**: aplicar programación funcional (Wickham and Henry, 2023).
 13. **readtext**: importar archivos en formato “word” o “pdf” (Benoit and Obeng, 2023) .
 14. **text.alignment**: aplicar el algoritmo “Smith-Waterman” (Wijffels, 2022).
 15. **spacyr**: wrapper para usar el framework Spacy (Benoit and Matsuo, 2020b).
 16. **config**: configurar variables globales (Allaire, 2020).
 17. **tm**: ejecutar procesos de minería de texto (Feinerer and Hornik, 2023).
 18. **quanteda**: análisis cuantitativo de textos (Benoit et al., 2018b).
3. Para realizar los diagramas y esquemas representados en esta investigación se uso el software “draw.io” (dra, 2023).
 4. Este documento, tanto en su versión PDF como en HTML se generó con la librería Bookdown (Xie, 2023).

Bibliografía

- (2016). *World Development Report 2016: Digital Dividends*. World Bank, Washington, DC. OCLC: ocn915488955.
- (2018). Text to sentence splitter using heuristic algorithm by philipp koehn and josh schroeder. <https://github.com/mediacloud/sentence-splitter>.
- (2022). Passage reranking multilingual bert. <https://huggingface.co/ambroad/bert-multilingual-passage-reranking-msmarco>.
- (2022). Text embeddings by weakly-supervised contrastive pre training. <https://huggingface.co/intfloat/multilingual-e5-base>.
- (2023). Hugging face models. <https://huggingface.co/models>.
- (2023). Open-source vector similarity search for postgres. <https://github.com/pgvector/pgvector-python>.
- (2023). roberta-base-bne-ranker. <https://huggingface.co/IIC/roberta-base-bne-ranker>.
- (2023). Security-first diagramming for teams. <https://www.drawio.com>.
- (2023). Shinyproxy. <https://shinyproxy.io>.
- , R., Wickham, H., and Müller, K. (2022). Dbi: R database interface.
- Aggarwal, C. C. (2018a). *Machine Learning for Text*. Springer International Publishing : Imprint: Springer, Cham, 1st ed. 2018 edition. DOI: 10.1007/978-3-319-73531-3.
- Aggarwal, C. C. (2018b). *Machine Learning for Text*. Springer International Publishing : Imprint: Springer, Cham, 1st ed. 2018 edition. DOI: 10.1007/978-3-319-73531-3.

- Aggarwal, C. C. and Zhai, C., editors (2012). *Mining text data*. Springer, New York Heidelberg. DOI: 10.1007/978-1-4614-3223-4.
- Alammar, J. (2019). The illustrated word2vec. <https://jalammar.github.io/illustrated-word2vec>. Acceso: 18 el octubre, 2023.
- Allaire, J. (2020). config: Manage environment specific configuration values.
- Bailey, E. (2022). shinybs: Twitter bootstrap components for shiny.
- Balog, K. (2018). *Entity-Oriented Search*. Number 39 in The Information Retrieval Series. Springer International Publishing : Imprint: Springer, Cham, 1st ed. 2018 edition. DOI: 10.1007/978-3-319-93935-3.
- Beltagy, I., Lo, K., and Cohan, A. (2019). Scibert: A pretrained language model for scientific text. *arXiv*.
- Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C. (2003). A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155.
- Benoit, K. and Matsuo, A. (2020a). spacyr: Wrapper to the 'spacy' 'nlp' library.
- Benoit, K. and Matsuo, A. (2020b). spacyr: Wrapper to the 'spacy' 'nlp' library.
- Benoit, K. and Obeng, A. (2023). readtext: Import and handling for plain and formatted text files.
- Benoit, K., Watanabe, K., Wang, H., Nulty, P., Obeng, A., Müller, S., and Matsuo, A. (2018a). quanteda: An r package for the quantitative analysis of textual data. 3:774.
- Benoit, K., Watanabe, K., Wang, H., Nulty, P., Obeng, A., Müller, S., and Matsuo, A. (2018b). quanteda: An r package for the quantitative analysis of textual data. 3:774.
- Bernhardsson, E. (2023). Annoy (approximate nearest neighbors oh yeah). <https://github.com/spotify/annoy>.
- Boettiger, C. and Eddelbuettel, D. (2017). An introduction to rocker: Docker containers for r. *The R Journal*, 9(2):527–536.
- Boleda, G. (2020). Distributional semantics and linguistic theory. *Annual Review of Linguistics*, 6(1):213–234.

- Brin, S. and Page, L. (1998). The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30(1-7):107–117.
- Büttcher, S., Clarke, C. L. A., and Cormack, G. V. (2010a). *Information retrieval: implementing and evaluating search engines*. MIT Press, Cambridge, Mass. OCLC: ocn473652398.
- Büttcher, S., Clarke, C. L. A., and Cormack, G. V. (2010b). *Information retrieval: implementing and evaluating search engines*. MIT Press, Cambridge, Mass. OCLC: ocn473652398.
- Canete, J., Chaperon, G., and Fuentes (2020). Spanish pre trained bert model and evaluation data.
- Cañete, J., Chaperon, G., Fuentes, R., Ho, J.-H., Kang, H., and Pérez, J. (2020). Spanish pre-trained bert model and evaluation data. In *PML4DC at ICLR 2020*.
- Chang, W. and Borges Ribeiro, B. (2021). shinydashboard: Create dashboards with 'shiny'.
- Chang, W., Cheng, J., Allaire, J., Sievert, C., Schloerke, B., Xie, Y., Allen, J., McPherson, J., Dipert, A., and Borges, B. (2023a). shiny: Web application framework for r.
- Chang, W., Cheng, J., Allaire, J., Sievert, C., Schloerke, B., Xie, Y., Allen, J., McPherson, J., Dipert, A., and Borges, B. (2023b). shiny: Web application framework for r.
- Chang, W., Csárdi, G., and Wickham, H. (2023c). shinytest: Test shiny apps.
- Chen, D. and Manning, C. (2014a). A fast and accurate dependency parser using neural networks. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.
- Chen, D. and Manning, C. D. (2014b). A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 740–750.
- Cheng, J., Borges, B., and Wickham, H. (2023). pool: Object pooling.

- Clarke, C. L., Cormack, G. V., and Tudhope, E. A. (2000). Relevance ranking for one to three term queries. *Information Processing and Management*, 36(2):291–311.
- Cook, J. (2017). *Docker for Data Science*. Apress.
- Coulouris, G. F., editor (2012). *Distributed systems: concepts and design*. Addison-Wesley, Boston, 5th ed edition.
- Czaja, L. (2018). *Introduction to Distributed Computer Systems: Principles and Features*. Number 27 in Lecture Notes in Networks and Systems. Springer International Publishing : Imprint: Springer, Cham, 1st ed. 2018 edition. DOI: 10.1007/978-3-319-72023-4.
- de Marneffe, M.-C., Manning, C. D., Nivre, J., and Zeman, D. (2021). Universal dependencies. *Computational Linguistics*, pages 1–54.
- Desagulier, G. (2017). *Corpus Linguistics and Statistics with R*. Springer International Publishing.
- Dettmers, T., Pagnoni, A., Holtzman, A., and Zettlemoyer, L. (2023). Qlora: Efficient finetuning of quantized llms.
- Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. Publisher: arXiv Version Number: 2.
- Dueñas, M., Rojas, D., and Morales, M. (2011). Propuesta metodológica para realizar mapas de conocimiento. *Revista Facultad de Ciencias Económicas*, 20(1):77–90.
- Edward Beeching, Nathan Habib, S. H. (2023). Open llm leaderboard. <https://huggingface.co/open-llm-leaderboard>.
- Eisenstein, J. (2019). *Introduction to natural language processing*. Adaptive computation and machine learning. The MIT Press, Cambridge, Massachusetts.
- Feinerer, I. and Hornik, K. (2023). tm: Text mining package.
- Fredkin, E. (1960). Trie memory. *Communications of the ACM*, 3(9):490–499.

- Frej, J., Schwab, D., and Chevallet, J.-P. (2020). WIKIR: A python toolkit for building a large-scale Wikipedia-based English information retrieval dataset. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 1926–1933, Marseille, France. European Language Resources Association.
- Goodrich, M. T., Tamassia, R., and Goldwasser, M. H. (2013). *Data structures and algorithms in Python*. Wiley, Hoboken, NJ.
- Granjon, D. (2021). shinydashboardplus: Add more 'adminlte2' components to 'shinydashboard'.
- Gökçe, O., Prada, J., Nikolov, N. I., Gu, N., and Hahnloser, R. H. (2020). Embedding-based scientific literature discovery in a text editor application. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*.
- Harman, D. (2011). *Information retrieval evaluation*. Number 19 in Synthesis lectures on information concepts, retrieval, and services. Morgan Claypool, San Rafael, Calif.
- Henry, L. and Wickham, H. (2023). rlang: Functions for base types and core r and 'tidyverse' features.
- Hester, J., Wickham, H., and Gjoneski, O. (2023). odbc: Connect to odbc compatible databases (using the dbi interface).
- Heydari, M. and Teimourpour, B. (2020). Analysis of researchgate, a community detection approach. In *2020 6th International Conference on Web Research (ICWR)*, pages 319–324. IEEE.
- Highsmith, J. A. (2000). *Adaptive software development: a collaborative approach to managing complex systems*. Dorset House Pub, New York.
- Honnibal, M., Montani, I., Van Landeghem, S., and Boyd, A. (2020). spacy: Industrial-strength natural language processing in python. *Zenodo*.
- Howard, J. and Ruder, S. (2018). Universal language model fine-tuning for text classification.
- Iannone, R. (2023). fontawesome: Easily work with 'font awesome' icons.

- Jurafsky, D. and Martin, J. H. (2009). *Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition*. Prentice Hall, Upper Saddle River, NJ, 2. ed. [nachdr.] edition.
- Karpathy, A. (2023). lookups on embeddings. https://github.com/karpathy/randomfun/blob/master/knn_vs_svm.ipynb.
- Knuth, D. E. (1997). *The art of computer programming*. Addison-Wesley, Reading, Mass, 3rd ed edition.
- Kraft, D. H. and Colvin, E. (2017). Fuzzy information retrieval. *Synthesis Lectures on Information Concepts, Retrieval, and Services*, 9(1):i–63.
- Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., Riedel, S., and Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks.
- Li, H. (2018). A visual query system for scholar networks.
- Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach.
- Lv, K., Yang, Y., Liu, T., Gao, Q., Guo, Q., and Qiu, X. (2023). Full parameter fine-tuning for large language models with limited resources.
- Mahapatra, A. K. and Biswas, S. (2011). Inverted indexes: Types and techniques. *International Journal of Computer Science Issues*, 8.
- Manning, C., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S., and McClosky, D. (2014). The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Baltimore, Maryland. Association for Computational Linguistics.
- Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to information retrieval*. Cambridge University Press, New York. OCLC: ocn190786122.
- Martín de Santa Olalla Sánchez, A. (1994). Una propuesta de codificación morfosintáctica para corpus de referencia en lengua española.

- Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. Publisher: arXiv Version Number: 3.
- Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. Publisher: arXiv Version Number: 1.
- Minixhofer, B., Pfeiffer, J., and Vulić, I. (2023). Where’s the point? self-supervised multilingual punctuation-agnostic sentence segmentation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7215–7235, Toronto, Canada. Association for Computational Linguistics.
- Muennighoff, N. (2022). Sgpt: Gpt sentence embeddings for semantic search.
- Muennighoff, N., Tazi, N., Magne, L., and Reimers, N. (2022). Mteb: Massive text embedding benchmark.
- Nogueira, R. and Cho, K. (2019). Passage re-ranking with bert.
- Nüst, D., Sochat, V., Marwick, B., Eglen, S. J., Head, T., Hirst, T., and Evans, B. D. (2020). Ten simple rules for writing dockerfiles for reproducible data science. *PLOS Computational Biology*, 16(11):e1008316.
- Padró, L. and Stanilovsky, E. (2012). Freeling 3.0: Towards wider multilinguality. In *Proceedings of the Language Resources and Evaluation Conference (LREC 2012)*, Istanbul, Turkey. ELRA.
- Pang, L., Lan, Y., Guo, J., Xu, J., Xu, J., and Cheng, X. (2017). Deeprank: A new deep architecture for relevance ranking in information retrieval. *arXiv*.
- Pedersen, T. L. (2022a). ggraph: An implementation of grammar of graphics for graphs and networks.
- Pedersen, T. L. (2022b). ggraph: An implementation of grammar of graphics for graphs and networks.
- Penedo, G., Malartic, Q., Hesslow, D., Cojocar, R., Cappelli, A., Alobeidli, H., Pannier, B., Almazrouei, E., and Launay, J. (2023). The refinedweb dataset for falcon llm: Outperforming curated corpora with web data, and web data only. Publisher: arXiv Version Number: 1.

- Pennington, J., Socher, R., and Manning, C. (2014). Proceedings of the 2014 conference on empirical methods in natural language processing (emnlp). pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.
- Perrier, V. and Meyer, F. (2023). apexcharter: Create interactive chart with the javascript 'apexcharts' library.
- Perrier, V., Meyer, F., and Granjon, D. (2023). shinywidgets: Custom inputs widgets for shiny.
- R Core Team (2023). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.
- Reimers, N. and Gurevych, I. (2019a). Sentence-bert: Sentence embeddings using siamese bert-networks.
- Reimers, N. and Gurevych, I. (2019b). Sentence-bert: Sentence embeddings using siamese bert-networks.
- Reimers, N. and Gurevych, I. (2020). Making monolingual sentence embeddings multilingual using knowledge distillation.
- Risch, J., Möller, T., Gutsch, J., and Pietsch, M. (2021). Semantic answer similarity for evaluating question answering models. *arXiv*.
- Robertson, S. and Zaragoza, H. (2009). The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.
- Ruiz Fabo, P. and Bermúdez-Sabel, H. (2019). Navegación de corpus a través de anotaciones lingüísticas automáticas obtenidas por procesamiento del lenguaje natural: de anecdótico a ecdótico.
- Salakhutdinov, R. and Hinton, G. (2009). Semantic hashing. *International Journal of Approximate Reasoning*, 50(7):969–978.
- Sali, A. and Attali, D. (2020). shinycssloaders: Add loading animations to a 'shiny' output while it's recalculating.
- Schloerke, B., Dipert, A., and Borges, B. (2021). shinyloadtest: Load test shiny applications.

- Segev, E. (2021). *Semantic Network Analysis in Social Sciences*. Routledge.
- Smith, T. and Waterman, M. (1981). Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1):195–197.
- Stephens, R. (2015). *Beginning software engineering*. Wrox, A Wiley Brand, Indianapolis, IN.
- Straka, M. and Straková, J. (2017). Tokenizing, pos tagging, lemmatizing and parsing ud 2.0 with udpipe. *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*.
- Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Ferrer, C. C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., Fuller, B., Gao, C., Goswami, V., Goyal, N., Hartshorn, A., Hosseini, S., Hou, R., Inan, H., Kardas, M., Kerkez, V., Khabsa, M., Kloumann, I., Korenev, A., Koura, P. S., Lachaux, M.-A., Lavril, T., Lee, J., Liskovich, D., Lu, Y., Mao, Y., Martinet, X., Mihaylov, T., Mishra, P., Molybog, I., Nie, Y., Poulton, A., Reizenstein, J., Rungta, R., Saladi, K., Schelten, A., Silva, R., Smith, E. M., Subramanian, R., Tan, X. E., Tang, B., Taylor, R., Williams, A., Kuan, J. X., Xu, P., Yan, Z., Zarov, I., Zhang, Y., Fan, A., Kambadur, M., Narang, S., Rodriguez, A., Stojnic, R., Edunov, S., and Scialom, T. (2023). Llama 2: Open foundation and fine-tuned chat models. Publisher: arXiv Version Number: 2.
- Trotman, A., Puurula, A., and Burgess, B. (2014). Improvements to bm25 and language models examined. *Proceedings of the 2014 Australasian Document Computing Symposium*.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. Publisher: arXiv Version Number: 7.
- Voorhees, E. M., Harman, D. K., Voorhees, E. M., of Standards, N. I., and Technology, editors (2005). *TREC: experiment and evaluation in information retrieval*. Digital libraries and electronic publishing. MIT, Cambridge, Mass. London.
- Wang, L., Yang, N., Huang, X., Jiao, B., Yang, L., Jiang, D., Majumder, R., and Wei, F. (2022). Text embeddings by weakly-supervised contrastive pre-training.

- Wickham, H. (2021). *Mastering Shiny: build interactive apps, reports, and dashboards powered by R*. O'Reilly Media, Inc., Sebastopol, CA, first edition edition. OCLC: 1249505228.
- Wickham, H. (2022a). rvest: Easily harvest (scrape) web pages.
- Wickham, H. (2022b). rvest: Easily harvest (scrape) web pages.
- Wickham, H. (2022c). stringr: Simple, consistent wrappers for common string operations.
- Wickham, H. (2023). httr2: Perform http requests and process the responses.
- Wickham, H., François, R., Henry, L., Müller, K., and Vaughan, D. (2023a). dplyr: A grammar of data manipulation.
- Wickham, H., Girlich, M., and Ruiz, E. (2023b). dbplyr: A 'dplyr' back end for databases.
- Wickham, H. and Henry, L. (2023). purrr: Functional programming tools.
- Wickham, H., Ooms, J., and Müller, K. (2023c). Rpostgres: Rcpp interface to postgresql.
- Wickham, H. and Seidel, D. (2022). scales: Scale functions for visualization.
- Wickham, H., Vaughan, D., and Girlich, M. (2023d). tidyr: Tidy messy data.
- Wijffels, J. (2022). text.alignment: Text alignment with smith-waterman.
- Wijffels, J. (2023a). udpipe: Tokenization, parts of speech tagging, lemmatization and dependency parsing with the 'udpipe' 'nlp' toolkit.
- Wijffels, J. (2023b). udpipe: Tokenization, parts of speech tagging, lemmatization and dependency parsing with the 'udpipe' 'nlp' toolkit.
- Wijffels, J. (2023c). udpipe: Tokenization, parts of speech tagging, lemmatization and dependency parsing with the 'udpipe' 'nlp' toolkit.
- Willett, P. (2006). The porter stemming algorithm: then and now. *Program*, 40(3):219–223.

- Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T. L., Gugger, S., Drame, M., Lhoest, Q., and Rush, A. M. (2019). Huggingface’s transformers: State-of-the-art natural language processing.
- Xie, Y. (2023). bookdown: Authoring books and technical documents with r markdown.
- Xie, Y., Cheng, J., and Tan, X. (2023a). Dt: A wrapper of the javascript library ‘datatables’.
- Xie, Y., Cheng, J., and Tan, X. (2023b). Dt: A wrapper of the javascript library ‘datatables’.
- Zhai, C. and Massung, S. (2016). *Text data management and analysis: a practical introduction to information retrieval and text mining*. Number #12 in ACM Books. ACM Books, New York, first edition edition. OCLC: ocn957355971.
- Zhou, J., Zhou, Y., and Xu, Y. (2018). Analogy search engine: Finding analogies in cross-domain research papers.