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 e implementación de un sistema para hacer la *Recuperación, Extracción y Clasificación de Información de SABER UCV*. El sistema ejecuta procesos de clasificación, almacenamiento y recuperación de información sobre los Resúmenes, tanto de las tesis como de los trabajos especiales de grado (TEG) que se encuentran publicados en el repositorio institucional Saber UCV y permite al usuario hacer procesos de búsqueda mediante una aplicación web.

A los Resúmenes con los cuales se conforma el corpus, se le aplican técnicas de Procesamiento de Lenguaje Natural (PLN), de Minería de Texto, de indexación en base de datos y se usan modelos preentrenados de inteligencia artificial para generar *embeddings* y soportar procesos de búsqueda semántica.

Mediante la búsqueda de palabras o frases, aplicando filtros como fechas de publicación, área en la cual se generó la investigación y/o el nivel académico, el usuario genera un *query* con el que se pueden recuperar los trabajos de mayor relevancia en que se encuentran contenidas tales palabras, o palabras similares y a partir de ahí enriquecer la experiencia del usuario con la presentación de la información extraída en tablas interactivas, gráficos, grafos de coocurrencia de palabras y recomendaciones de textos que puedan ser de interés para el investigador.

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

También se implementa una rutina que permite clasificar las tesis y los TEG, según el área académica donde cursó estudios el autor del correspondiente trabajo y así se solventa la carencia que actualmente presenta Saber UCV, donde no están disponibles estas clasificaciones.

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

**Abstract:**

The research and implementation of a system for the Retrieval, Extraction, and Classification of Information from SABER UCV (UCV's Institutional Repository) is presented. The system performs processes of classification, storage, and retrieval of information about Abstracts, both from Theses and Special Degree Works (TEG), that are published in the institutional repository Saber UCV, allowing users to perform search processes through a web application.

Natural Language Processing (NLP) techniques, Text Mining, database indexing, and pre-trained artificial intelligence models are applied to the Abstracts forming the corpus. These techniques are used to generate embeddings and support semantic search processes.

By searching for words or phrases and applying filters such as publication dates, research area, and/or academic level, the user generates a query with which the most relevant works containing those words or similar words can be retrieved. From there, the user experience is enhanced by presenting the extracted information in interactive tables, graphs, word co-occurrence graphs, and recommendations of texts that might be of interest to the researcher.

The application is implemented as a distributed system under the client-server architecture and relies on the use of orchestrated containers.

Additionally, a routine is implemented that allows the classification of Theses and TEGs based on the academic area where the corresponding work's author studied, thus addressing the current deficiency of Saber UCV, where these classifications are not available.

**Keywords:** Information Retrieval, Natural Language Processing, Text Mining, Relevance, Full Text Search, Artificial Intelligence, Embeddings, Semantic Search, Knowledge Maps.

# *Dedicatoria:*

*A mi hijo Cassiel y*

*mi esposa Waleska.*

# *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 Oddysey 2.

- A mi tía Mercedes Infante.

- A Cesar Alejandro García por todas las ayudas.

- A mi hermano David por su 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. Nuri Hurtado Villasana por tomarse el tiempo de escucharme y brindarme sugerencias en la elaboración de este trabajo.

- A todo el personal del Postgrado: sus buenos días, por tener a mano la llave, por ayudar a mantener viva la Academia.

- 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 toda la comunidad de creadores de software libre y open sourcem en especial a los #useRs por motivarme a adentrarme al mundo de las ciencias de la computación.

- A Alexandra Asanovna Elbakyan, sin ella serían mínimas las citas bibliográficas de esta Investigación.

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

# Índice de figuras

# Índice de cuadros

# 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, libros, páginas de internet especializadas en algún tema o los repositorios digitales de documentos. 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 documentos que puedan resultar de mayor relevancia.

Una de las herramientas con que cuentan los investigadores, para la búsqueda de documentos que se encuentran en formato digital, son los Sistemas de Recuperación de Información, donde ante una búsqueda, a la cual denominaremos el ***query***, serán recuperados del conjunto de documentos, conjunto al que denominaremos ***Corpus***, aquellos que cumplan ciertos parámetros (**?**). Los documentos que el Sistema logre recuperar se deben adaptar a las necesidades del investigador y en caso de existir múltiples documentos, el sistema debe poder jerarquizar aquellos que puedan resultar de mayor relevancia.

En el caso de la Universidad Central de Venezuela existe un repositorio digital de documentos denominado SABER.UCV al cual se puede acceder desde la dirección web saber.ucv.ve el cual permite realizar búsquedas de información sobre documentos generados dentro de la comunidad de investigadores de la Universidad.

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)**, el cual complementa las funcionalidades de búsqueda de información que actualmente tiene el repositorio oficial de la Universidad Central de Venezuela.

La implementación del SCSU hace un aporte a los investigadores que necesitan realizar búsquedas de información sobre el subconjunto de tesis doctorales, de trabajos de grado de maestría y de pregrado, que se encuentran disponibles en SABER.UCV . Adicionalmente, la representación de los resultados de los *querys* son enriquecidas con gráficos interactivos que muestran “mapas del conocimiento” (**?**).

Gran parte de las funcionalidades que incorpora el SCSU se sustentan en los procesos de extracción y clasificación de información de datos que originalmente no se encuentran estructurados en el

repositorio oficial, como lo son los nombres de las carreras de pregrado o del postrgrado donde fue generada cada una de las investigaciones o el nombre del tutor que acompañó el desarrollo de cada investigación, así que el Sistema acá descrito contribuye efectivamente a que estos datos puedan estar disponibles y afinar con una mayor cantidad de parámetros las búsquedas de información que se deban realizar y se incorporan técnicas de búsqueda indexada mediante un índice inverso que establece los criterios de relevancia al momento de jerarquizar y mostrar los resultados de una búsqueda.

## 1.1 Estructura:

En el Capítulo 2 **El Problema** se hace una exposición de cuáles son las funcionalidades de SABER.UCV que pueden ser complementadas por el Sistema Complementario Saber UCV mientras que en 2.2 se establecen las delimitaciones en el proceso de investigación abordado. En 2.3 se justifica la necesidad de realizar este estudio y en 2.5 se presenta la Solución propuesta junto con el 2.5.1 Objetivo General y los 2.5.2 Objetivos Específicos . En **??** se enuncia el alcance del proceso de recuperación, extracción y clasificación de Información de SABER UCV y en **??** se enumeran los aportes que brinda esta investigación.

En el Capítulo 3 **Marco Teórico** se hace una reseña histórica 3.1 sobre el problema computacional que es la "búsqueda", entendido de manera genérica. En la sección 3.2 "Recuperación de Información" se mencionan los conceptos para comprender los procesos de búsqueda de texto dentro de un Corpus y también se introducen a los "Sistemas" 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 en 3.3 Procesamiento de Texto, como los son el 3.3.1 Procesamiento del Lenguaje Natural y 3.3.2 la Minería de Texto . Seguidamente en 3.4 se conceptualiza a los "Sistemas Distribuidos", motivado a que la implementación realizada del Software 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 **?**(mm) **Marco Metodológico** se presenta la metodología de trabajo Kanban 4.1 que sirvió para realizar la planificación de la investigación así como también el Desarrollo Adaptable de Software –Adaptive Software Development (ASD)- 4.2 que fue la metodología adaptada para realizar los ciclos de desarrollo del SCSU.

En el Capítulo 5 **Desarrollo** se presenta la 5.1 Descripción general de la Solución y en 5.2 se presenta la Arquitectura de la misma. Luego en **??** Análisis y Diseño se muestran las consideraciones adoptadas para dicho proceso. En 5.3 Ciclos de Desarrollo se muestran las iteraciones que se hicieron para crear el SCSU; en un primer ciclo para la 5.3.1 Obtención del Conjunto de Datos , en un segundo ciclo 5.3.2 para la Extracción y Clasificación de Información del Conjunto de Datos , en el tercero 5.3.3 para crear el Prototipo de Buscador y en el cuarto 5.3.4 se realizó la Integración de Componentes del Software . En este Capítulo adicionalmente se realizan las distintas 5.4 Pruebas tanto 5.4.1 Funcionales, 5.4.2 de Rendimiento y en 5.4.3 las pruebas de Relevancia sobre la información recuperada al momento de hacer una búsqueda.

Para concluir esta investigación, en el Capítulo 6 **Conclusiones** se exponen las 6.1 Contribuciones

y los 6.2 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 se introduce la 2.1 **Descripción del Problema** que se intenta solucionar con esta Investigación. En 2.2 se hace la **Delimitación del Problema** se hacen algunas consideraciones al respecto. En 2.3 **Justificación e Importancia** se menciona la razón principal para haber llevado a cabo este trabajo. En 2.4 se listan los **Aportes** que se logró hacer en el transcurso de la investigación. En 2.5 **Descripción de la Solución** se hace una aproximación a una descripción con algún nivel de detalle.

En 2.5.1 se menciona el **Objetivo Principal** que se trazó, mientras que en 2.5.2 se enumeran los **Objetivos Específicos**.

## 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 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).

También pueden realizar la búsqueda en el texto que incluye el nombre 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 algunos criterios adicionales, como lo es el área de académica donde fue realizada la investigación, en particular, si fue hecha en una escuela determinada, o en una facultad, o postgrado.

Aplicar el filtrado descrito en el párrafo anterior es donde se presenta una de las limitaciones principales al hacer búsquedas en Saber UCV, ya que el sistema no dispone de esos criterios para la recuperación de información.

De manera similar ocurre con el nombre del tutor de los trabajos de grado y tesis, al no encontrarse registro de esta información.

Motivado a que es viable realizar procesos para extraer y clasificar la información faltante, se realiza esta Investigación que mediante la implementación de un software denominado **Sistema Complementario Saber UCV (SCSU)**, extrae y clasifica los datos del área académica donde se realizaron las distintas investigaciones de tesis y trabajos de grado que están alojadas en la página precitada.

Adicionalmente, se detectó que es posible generar una propuesta alterna de implementación de criterios de relevancia para realizar la recuperación de información. Este problema será descrito con mayor detalle en 5.3.3 correspondiente al tercer ciclo de desarrollo de la Solución.

## 2.2 Delimitación del Problema:

El corpus con el que se trabajará sólo es el subconjunto de los trabajos de grado de pregrado y maestría junto con las tesis doctorales, no abarcando otro tipo de documentos que se encuentran registrados en el repositorio. No obstante en la implementación del ciclo final 5.3.4 se incluyen dos revistas: una científica (Gestión I+D) y la otra del área de filosofía (Episteme) de la Universidad Central de Venezuela. También se añade una publicación del repositorio digital de la Universidad de los Andes Saber U.L.A. para mostrar posibles ampliaciones en la ingesta de distintos tipos de publicaciones y fuentes dentro del SCSU.

Sin embargo, posibles incorporaciones de otras fuentes de datos implican procesos de revisión en las estructuras de los datos, ya que por los momentos el Sistema no está diseñado para incorporaciones de datos adicionales, sin la modificación de los códigos con los que se extraen los datos desde las páginas web donde se alojan los documentos.

## 2.3 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 amplian los criterios de búsqueda disponibles para investigadores que necesiten realizar consultas sobre los documentos disponibles en el repositorio.

## 2.4 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.

- Se crean visualizaciones y representaciones de Mapas del Conocimiento.
- Se implementa la búsqueda de texto completo con una función de relevancia distinta.
- Se implementa un "sistema de recomendación" de documentos que presenten similitudes con los textos recuperados.
- Se obtiene mejores métricas de desempeño en la recuperación de información.
- La experiencia del usuario se enriquece mediante gráficos que muestran la evolución de aparición de los términos buscados en el período establecido para la búsqueda de la información.
- El Sistema fue diseñado para estar en producción y todos los componentes son de código abierto, libres de algún pago de licencia.
- Se puede implementar fácilmente el Sistema en un servidor ya que está diseñado de manera que con los archivos "docker compose" y "dockerfiles", se puedan desplegar la aplicación, adicionalmente alcanzando elevados niveles de reproducibilidad en todos los procesos que conforman la Solución.
- El tipo de componentes hace que el Sistema sea escalable y adaptable a la demanda de accesos.
- El Sistema está diseñado para poder actualizarse cada vez que sea necesario o 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 una aproximación a la implementación de "búsquedas semánticas".

## 2.5 Descripción de la Solución:

La Solución que se propone es un Sistema de Recuperación de Información implementado para que mediante técnicas de extracción de datos de archivos html´s se puedan obtener datos estructurados de los trabajos mencionados en 2.2 que están disponibles en Saber UCV. Los datos que se extraen de cada documento son el título de la investigación, el nombre del investigador, las palabras claves, la fecha de publicación y el resumen.

Posteriormente se descarga el documento en formato word o pdf, anexo a la investigación, que es donde se encuentran, generalmente en las primeras páginas, los datos de la facultad, escuela y/o postgrado donde se generó la investigación así como el nombre del tutor.

Con todos los datos recopilados se conforma un corpus, al cual se le aplican distintas técnicas de procesamiento del lenguaje natural y de minería de texto, que permiten obtener un corpus anotado y un índice invertido sobre los textos, que posteriormente alimentan a un sistema gestor de base de datos.

Una vez que se tienen estructurados los datos, se cuenta con una aplicación web que se soporta en un sistema distribuido conformado por contenedores que son gestionados por un orquestador.

A esta aplicación web está diseñada para permitir la exploración exhaustiva. El usuario puede ingresar para realizar consultas de texto, aplicando distintos filtros como la selección del área académica donde se generó la investigación, bien sea en la escuela, postgrado o facultad, así como la selección de un rango de fechas y las palabras a buscar. La aplicación funciona bajo la arquitectura cliente servidor.

Posterior a realizar la búsqueda el sistema recuperará los documentos que cumplan con las condiciones establecidas y les será aplicada una función de relevancia para mostrar de primero aquellos que puedan resultar de mayor interés para el usuario.

Adicionalmente para cada texto que se muestre en los resultados, se incluirán recomendaciones de otros documentos que sean similares. Igualmente se presentará un selector, que al estar activado, hará que se generé una representación de "mapas del conocimiento" según una adaptación *ad-hoc* hecha para la implementación del SCSU basada en el trabajo de (**?**) donde mediante una representación en grafos en formato interactivo, se podrán aplicar filtros sobre palabras coocurrentes en los resultados de la búsqueda.

La Solución contempla procesos de actualización de los datos ante la incorporación de nuevas investigaciones en el repositorio Saber UCV.

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

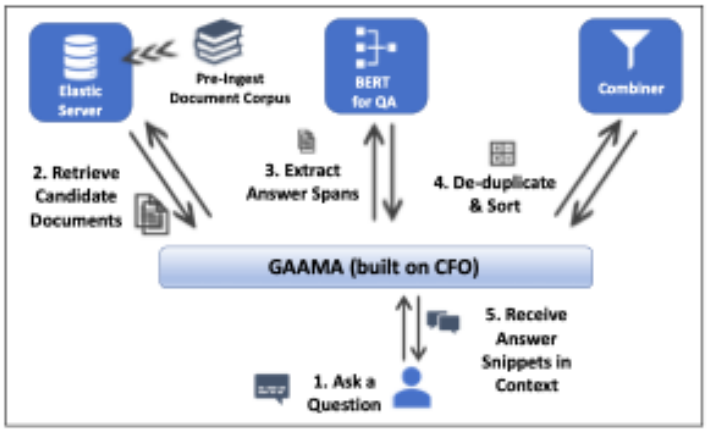


Figura 2.1: Arquitectura del Sistema-Este diagrama se va a traducir y adaptar al SCSU. Se coloca ya que es muy similar a la Solución Implementada

### 2.5.1 Objetivo General:

Implementar un Sistema que pueda 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.

### 2.5.2 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 que permita realizar las “búsquedas de texto completo” sobre el corpus conformado.
4. Generar recomendaciones de investigaciones que presenten similitud con los documentos recuperados por el Sistema.

# Capítulo 3

# Marco Teórico-Referencial:

En este capítulo se exponen los fundamentos teóricos para los procesos y métodos que sustentan la investigación 1 **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 Natura**l 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 este 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)** y 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* (**?**, 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*" (**?**).

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.

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 expuesto en 5.3.2.

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 ser resueltos.

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*** (**?**). 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 5.3.2 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* (**?**) 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 (***query***) de texto. El investigador Charu Aggarwal en su libro sobre Minería de Texto (**?**) 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)" (**?**).

A los efectos de delimitar el espacio de búsqueda que se realiza ante la acción de la consulta, el *query*, se define al **Corpus** como el conjunto cerrado de documentos codificados electrónicamente que se encuentra integrado en un sistema de almacenamiento y conformará el agregado (**?**) sobre el cual se realizará el proceso de recuperación de información. Entendido desde otra perspectiva, es el conjunto de datos sobre el cuál se hará la búsqueda.

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 comprendido en un subconjunto del corpus, que es a lo que se denomina **búsqueda múlti atributo**.

La información que se recolecte en una búsqueda tendrá distintos aspectos que aportarán peso en el orden en que sea presentada al usuario y no sólo vendrá dado por la aparición de las palabras sino también por otros elementos como lo puede ser la aparición de la frase del ***query*** dentro del título, la proximidad (la cercanía entre dos palabras) que tengan los términos que conforman el ***query*** o por otra parte la frecuencia que una palabra, o varias, se repitan dentro un determinado documento que compone el *corpus*. 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 documento, similar a la propuesta del algoritmo ***PageRank*** (**?**), 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* (**?**).

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 también añadiendo a los resultados, documentos 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*** (**?**). En 3.3.3 **Similitud de Documentos** y en 3.5.1 **Embeddings,** igualmente se mencionan y especifican algunas de las técnicas con las cuales se incorpora este otro lote de documentos en los resultados de 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 (**?**), 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 (**?**) 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 (**?**) . 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 querys, 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 coomo IRS sobre corpus de investigaciones científicas.

1. Arvix 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 *Asociation Computery Machine* (ACM) alojado en https://dl.acm.org incorpora un motor de búsqueda con particulares características ya que los resultados de una búsqueda son acompañados de distintas representaciones gráficas que le dan un valor adicional a la representación de los resultados. En la figura 3.1 se ve una de estas representaciones que incluye la frecuencia de aparición del *query* en el tiempo.

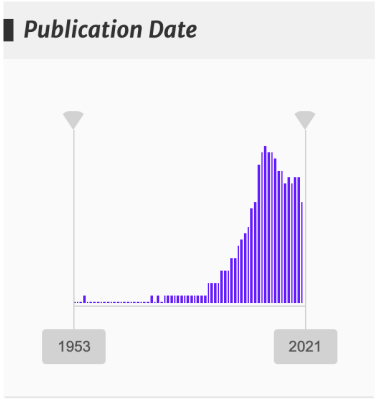


Figura 3.1: Gráfico que acompaña resultados de búsqueda de un término en la biblioteca digital de la Association for Computing Machinery (https://dl.acm.org/)

### 3.2.3 Modelos de Recuperación de Información:

#### 3.2.3.1 Recuperación boleana:

Ante una búsqueda de información se recorre linealmente todo el documento para retornar un valor boleano indicando la presencia o no del término buscado. Es uno de los primeros modelos que se uso y está asociado a técnicas de *grepping* (**?**). 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 *querys* 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 boleana si bien representa una buena aproximación a la generación de *querys* 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 *querys* 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 5.3.3 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 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 (**?**).

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 *querys*. Algunos ejemplos de estos índices son el *Generalized Inverted Index* (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 junto con el valor del costo en espacio de almacenamiento.

[1] Precision: la fracción, o porcentaje, de los documentos recuperados que son relevantes en la búsqueda efectuada.

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

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

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 (**?**) se encuentran detalles adicionales sobre este tipo de índices.

En 3.5 se muestra el **Estado del Arte** en los Sistemas de Recuperación de Información al incorporar representaciones de **Embeddings** (**?**) 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 (**?**). 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.

a. Incluir tema de relevancia. Trecs. Métodos- recorrido histórico. Subjetividad.

### 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. Estos algoritmos utilizan técnicas de aprendizaje automático para modelar la relevancia de los documentos basándose en características específicas (**?**). 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.

#### 3.2.5.2 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 de búsqueda (**?**) 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 (**?**) de documentos TREC (**?**). 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 (**?**), 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:

Teniendo las siguientes medidas 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.

Enunciado el concepto de estas tres medidas es necesario determinar el proceso con que se puede determinar la relevancia de los documentos recuperados y para esto se expone brevemente cómo fue que se crearon.

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”(**?**), 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 y el concepto para ese momento es similar al de precisión que recién definimos.

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 formulasen una pregunta que satisfactoriamente iba a ser respondida en el documento de su autoría (**?**).

Hoy en día se han construido distintos conjuntos de datos constituidos por la dupla "pregunta-documento", o índice del documento más relevante, 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 expression binaria: "relevante" o "no relevante" para la dupla query- documento. Con esto podremos 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 distintos documentos y temas donde expertos anotaron juicios de relevancia para indicar cuáles eran 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.

El problema que presenta este enfoque es que ante métodos de recuperación de documentos más avanzados como la búsqueda semántica, que será presentado más adelante, y ante el incremento de documentos digitales, cada vez 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 (**?**).

## 3.3 Procesamientos de texto:

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 nuestro sistema de recuperación de información, 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"* (**?**), como es el caso del "coreNLP" de la Universidad de Stanford (**?**) 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)* (**?**) o el reconocimiento de entidades nombradas (*Named Entity Recognigtion)*, 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 AnCora [4] . También

[4] **AnCora** es un corpus del **catalán (AnCora-CA)** y del **español (AnCora-ES)** con diferentes niveles de anotación como lema y categoría

la Universidad Politécnica de Cataluña creó la herramienta FreeLing [5] que implementó para el español, y catalán, algunas de las funcionalidades con que sí contaba para el idioma inglés el "coreNLP" descritas en el párrafo anterior. No obstante, su integración en cadenas de trabajo y la actualización de sus modelos de entrenamiento, presentan rezagos en comparación a otros modelos que actualmente se están usando, basados en el uso del aprendizaje mediante redes neuronales (**?**) 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):

Son las 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 (**?**). 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 signficado a las cuales se le llaman *tokens* (**?**). 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.

#### 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 (**?**) 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

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)

[5] https://nlp.lsi.upc.edu/freeling/node/1

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 la coocurrencia de palabras, que es una de las formas en que se representan los resultados de los *querys* 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 (**?**). 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 (**?**). 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 reduce el espacio de búsqueda en los documentos.

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 esta tarea en varios de los modelos de aprendizaje automático preentrenados para realizarla, no obstante no se disponen datos puntuales para la precisión que se alcanza en 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 (**?**).

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

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

#### 3.3.2.1 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 *"Sparce Term-Document Matrix)"* de tamaño *n X 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 evaluan 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,…,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 (**?**).

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.

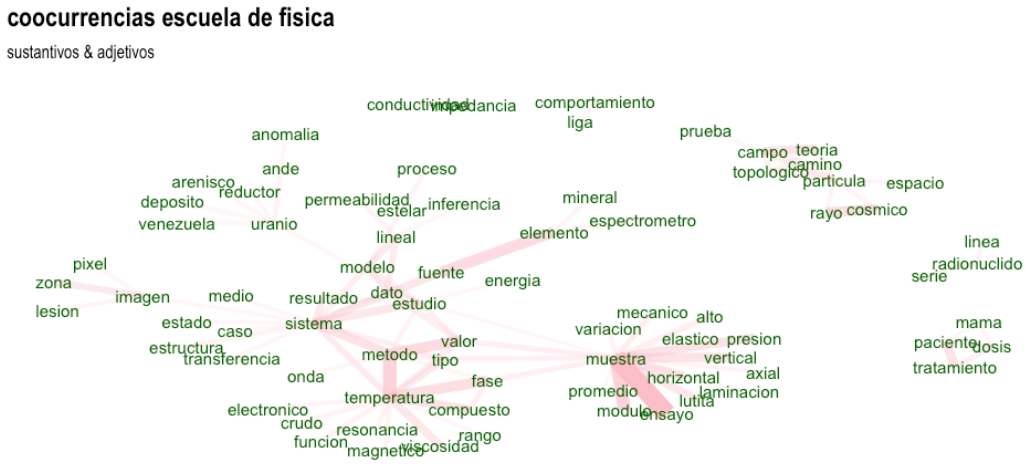


Figura 3.2: Coocurrencia de Palabras

La representación gráfica y el método de extracción de sustantivos y adjetivos, resulta similar a la propuesta metodológica realizada por (**?**) para crear Mapas del Conocimiento con "las palabras claves obtenidas a través de búsquedas recurrentes y relacionadas". En esta Investigación se simplificará la obtención y representación de los Mapas del Conocimiento, 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 (**?**) . Un ejemplo de esta técnica es el uso de la similitud coseno que se explica con esta fórmula.

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

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 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 cual 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 sofware se encuentran localizados en una red de computadores y estos coordinan sus acciones sólo mediante el pase de mensajes" (**?**).

Algunas de las principales características que tienen los sistemas distribuidos es la tolerancia a fallos, compartir recursos, concurrencia, ser escalables (**?**) 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 (**?**).

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

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 (**?**). 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 la Propuesta Técnica 5.3.4 serán expuestas las funcionalidades de cada contenedor y se apreciará la integración que brindará el 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* ) (**?**), 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, el corpus, 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 Frecuency (TDF), para representar en un documento la aparición de una determinada palabra, se colocaba en la matríz el valor "1" en la posición *i,j* , o se colocaba un "0" en su ausencia, correspondo la *i* al índice del documento y *j* al índicede 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 o corpus, 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 Frecuency (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

Cuadro 3.1: Embedding bidimensional para representar palabras

| Dimensión.1 | Dimensión.2 |
|---|---|
| 0.71038 | 1.76058 |
| 0.43679 | 1.93841 |
| 1.77337 | 0.00012 |

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* (**?**), 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", "estudent"* y"*wealth"*:

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

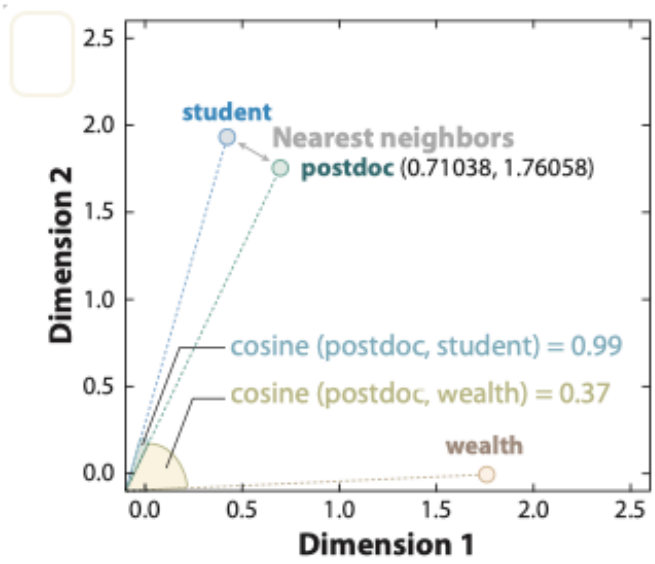


Figura 3.3: Representación de palabras en un plano

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 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, expandiendo su capacidad y llegando a ser un motor de respuestas (*answering engines*).

Los *embeddings* son representaciones numéricas densas de las palabras contenidas en un vocabulario. A diferencia de los modelos de tipo “one hot encodding (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” (**?**), se tenían 100 componentes, cifra considerablemente menor a las 20 mil que del modelo de “one hot encodding”, evitando así la alta esparcidad. Otro cambio sustancial que se introdujo con este tipo de representación es que los 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 el la figura 3.4 donde se mapean los números a una paleta de colores.

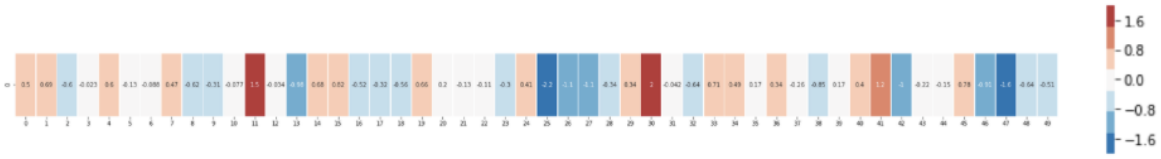


Figura 3.4: Representación de palabra ”king” mediante el modelo GloVe

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 corresponde al divulgador Alammar,J. (2019). The Illustrated Word2vec. Blog. https://jalammar.github.io/illustrated-word2vec/ (acceso 18 el octubre,2023). 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. Otro hito

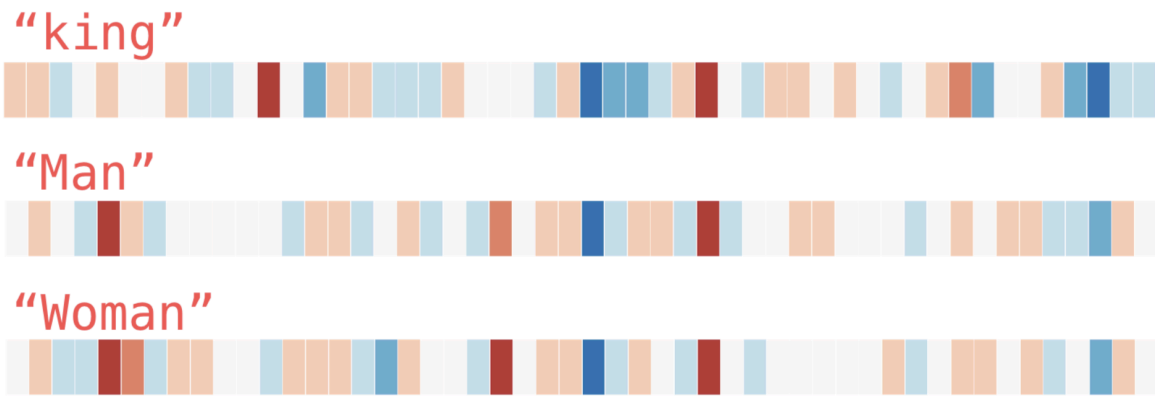


Figura 3.5: Representación de palabras mediante el modelo GloVe

fue la investigación “Semantic Hashing” (**?**) donde usaron redes neuronales para transformar datos de alta dimensionalidad en representaciones binarias de baja dimensionalidad. Esa investigación añadió como aporte que se empiezarán a usar técnicas de aprendizaje no supervisado para entrenar las redes neuronales, deshaciéndose de los cuellos de botella que introducían previamente los procesos de etiquetado, necesarios en métodos supervisados.

Ante la ampliación de capacidades de computo 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” (**?**) 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) (**?**) donde se hicieron representaciones vectoriales distribuidas para frases, mejorando la capacidad de capturar significados contextuales y relaciones sintácticas en un nivel más alto, siendo esto un avance importante hacia la comprensión de la semántica en un contexto más amplio, lo cual resultó crucial para aplicaciones como la recuperación de información y también para la traducción automática.

La investigación GloVe (**?**) 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 (**?**).

### 3.5.2 Arquitectura de Redes Neuronales *Transformers*:

En el año 2017 se publica "Attention Is All You Need" (**?**) 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 (**?**) 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" (**?**) 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 {#LLM} la sección referida a los Largos Modelos del Lenguaje.

Otro modelo basado en la arquitectura de Transformers, necesario referir, ya que a un componente del SCSU le da soporte, es el que se publicó bajo el título "Sentence-BERT: Sentence Embeddings" (**?**) 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 (**?**). 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 era necesario resolver era el lograr tener representaciones de embeddings para distintos idiomas, ya que inicialmente estaban entrenados con textos en idioma ingles y uno de las investigaciones que permitió avanzar en hacia modelos multilingües fue "Making Monolingual Sentence Embeddings Multilingual" (**?**), 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" (**?**), 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 tantos los modelos, como la evaluación de ellos.

[6] Los 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.

Para obtener información sobre los modelos de *embeddings* que presentan una elevada precision 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" (**?**) al que se puede acceder en el enlace https://huggingface.co/spaces/mteb/leaderboard y muestra información sobre 140 modelos preentrenados en el área del lenguaje.

### 3.5.3 Largos Modelos de Lenguaje:

Con la aparición de la arquitectura Tranformers se abrió el camino para la aparición de los Largos Modelos de Lenguaje *(Large Language Modes -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 increméntales que se han dado desde la publicación del modelo basado en los Transformers en el 2017.

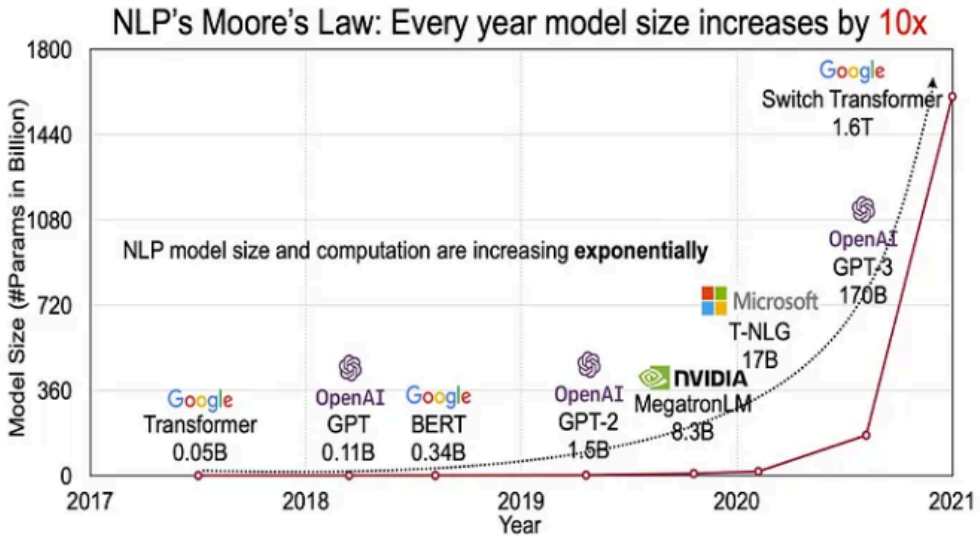


Figura 3.6: Evolución en la Cantidad de parámetros en los LLM

El crédito a la visualización 3.6 corresponde a Harishdatala (2023). Unveiling the Power of Large Language Models (LLMs). https://medium.com/@harishdatalab/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

tunning", 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 a manera de conversación a modo de *chat*. 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 gigantes características.

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 basto cúmulo de textos. Si hacemos otra abstracción de un nivel más 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 (**?**) o el modelo OpenLlama2 (**?**) [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 (**?**), 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 (**?**) 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 tunning o cuantizaciones, se puede ver en el enlace 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.

### 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** (**?**): esta técnica permite que un LLM que fue entrenado con un determinado corpus, pueda activar la extracción de información desde

[7] la 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.

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** (**?**): 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" (**?**), 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 búsquedas fáciles, 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.

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(**?**) . Igualmente mediante redes neuronales se ha buscado simular el comportamiento humano para jerarquizar los resultados obtenidos en procesos de búsqueda (**?**).

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 (**?**).

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 los modelos los querys, que permitan medir el grado de precisión que alcanza un determinado sistema (**?**).

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:

Para el desarrollo del Sistema se adoptó la metodología Kanban por considerarse idónea ante los retos propuestos. En ella se fomenta una cultura de mejora continua, incremental, al identificar cuellos de botella, la limitación del trabajo en curso y la mejora continua 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" y se basa en la visualización del flujo de trabajo mediante un tablero, que en el caso de los proyectos de desarrollo de software, facilita la toma de decisiones informadas y promueve la transparencia al proporcionar una visualización clara de las tareas y actividades involucradas (**?**).

En el tablero que se aprecia en la figura 4.1 se puede visualizar y facilitar la gestión del flujo de trabajo, desde la concepción de una idea, hasta su implementación y entrega. 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 se querían definir períodos fijos para alguna fase en particular sino más bien para el desarrollo general.

## 4.2 Desarrollo Adaptable de Software:

El **Adaptive Software Development (ASD)** (**?**) 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 la flexibilidad para abordar los desafíos cambiantes del desarrollo de software. A diferencia de los enfoques tradicionales, ASD reconoce la naturaleza impredecible del desarrollo de software y se adapta continuamente para satisfacer las necesidades

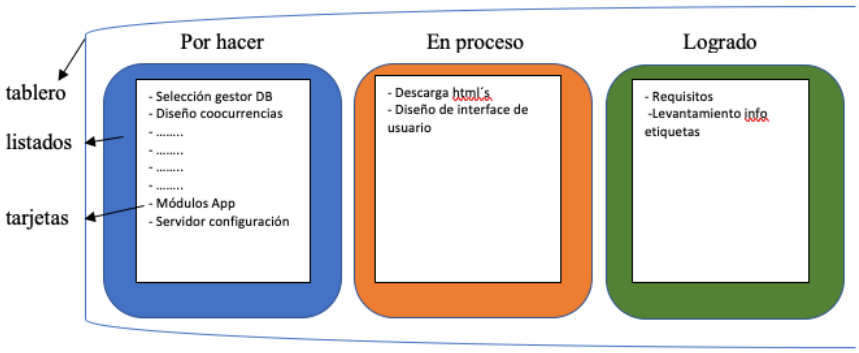


Figura 4.1: Representación de un tablero según la metodología Kanban

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.

**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.

#### 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.

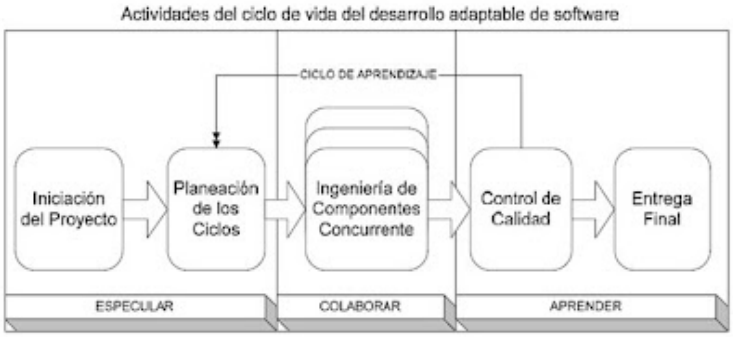


Figura 4.2: Ciclo ASD

#### 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.

Para el desarrollo del SCSU se hizo un proceso iterativo donde en cada ciclo se abordó cada una de las fases descritas y así se fueron añadiendo funcionalidades al Sistema, y también en otros casos desechándolas, ya que no se adaptaban de forma correcta a los objetivos propuestos. Como esta metodología se enfoca en construir el software en pequeñas y frecuentes partes incrementales, añadiendo funcionalidades con cada iteración, es válido asumir que este modelo es representativo para la aproximación de desarrollo realizada.

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 evolucionan. 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.

# Capítulo 5

# Desarrollo de la Solución:

En este Capítulo en **??** se presenta la **Descripción General de la Solución**. Posteriormente se muestra la 5.2 **Arquitectura de la Solución** con el "Modelo-Vista-Controlador". En 5.3 se exponen los cinco **Ciclos de Desarrollo** que se efectuaron, mientras que en 5.4 **Pruebas** se muestran las distintas pruebas que fueron ejecutadas.

## 5.1 Descripción General de la Solución:

La propuesta consiste en implementar un Sistema de Recuperación de Información para extraer datos estructurados de los trabajos disponibles en Saber UCV. Utilizando técnicas de extracción de datos de archivos HTML, se obtienen detalles como el título, el nombre del investigador, palabras clave, fecha de publicación y el resumen de cada investigación.

Posteriormente, el sistema descarga los documentos anexos en formato Word o PDF, que contienen la investigación, da lectura y extráe de la información sobre la facultad, escuela o postgrado, así como el nombre del tutor.

Todos los datos obtenidos se integran en un corpus, el cual es sometido a técnicas avanzadas de Procesamiento del Lenguaje Natural y Minería de Texto. Esto incluye la creación de un corpus anotado y un índice invertido y una tabla con los vectores de *embeddings* (vector database), esenciales para un eficiente manejo de la base de datos.

La aplicación resultante es una solución web distribuida, basada en una arquitectura cliente-servidor la cual permite a los usuarios explorar extensivamente el corpus anotado, realizando consultas de texto y aplicando varios filtros como área académica, rango de fechas y palabras clave. La relevancia de los resultados recuperados se determina mediante funciones de ponderación, y se presenta 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 presenta una herramienta interactiva de visualización, que permite la representación gráfica de "mapas del conocimiento". Estos mapas, basados en técnicas adaptadas de minería de texto, proporcionan una representación visual intuitiva de las palabras coocurrentes en los resultados de búsqueda.

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

## 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.

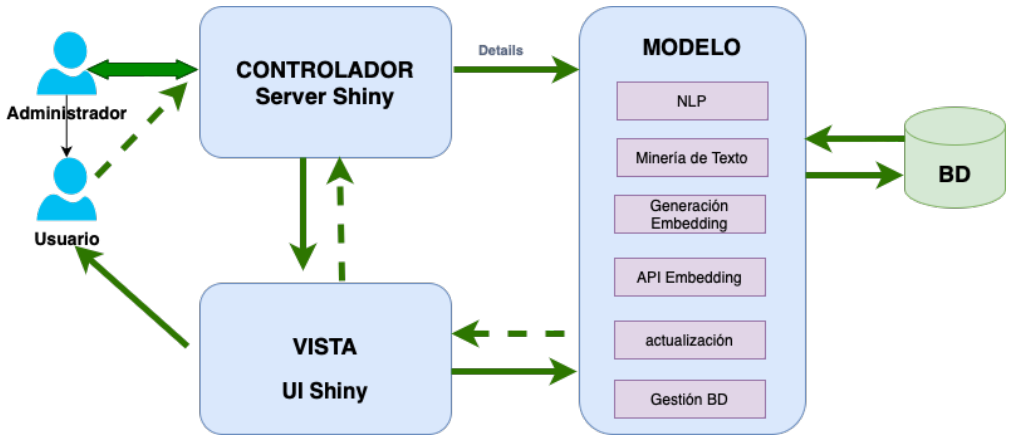


Figura 5.1: Modelo de Arquitectura MVC

### 5.2.1 1. Modelo:

El Modelo en el contexto de esta propuesta es la parte del Sistema que se ocupa de la manipulación y gestión de los datos mediante el gestor de base de datos PostgreSQL. Esto incluye el Procesamiento del Lenguaje Natural, la Minería de Texto y la creación del índice invertido. Además, el Modelo se encarga de hacer la lectura de los datos en html que se encuentran en el repositorio Saber UCV y extrae datos como el título, nombre del investigador, palabras clave, fecha de publicación y resumen. También gestiona la lectura y la extracción de datos de los documentos descargados en formatos word o PDF. Esta parte del Sistema también incluye la lógica para generar los *embeddings* y manejar el corpus anotado. Es el encargado de realizar las tareas de actualizar el corpus periódicamente y actualizar las recomendaciones de documentos a medida que se agreguen nuevos textos. En 5.3.4 el cuarto y 5.3.5 y quinto ciclo de iteración se expondrán con detalle los componentes del Modelo.

Estos son los principales procesos que contiene el Modelo:

1. **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.
2. **Minería de Texto:**

- POS: etiquetado del Part of Speach.
- Análisis de Frecuencia: Determinar la frecuencia de ocurrencia de palabras o frases.

3. **Generación de Embeddings:**
    - Utilización de 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.
4. **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.
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 descritos anteriormente son ejecutados y/o actualizados periódicamente.
    - Se realiza la validación de la integridad de datose para asegurar que los nuevos datos se integren correctamente sin errores o inconsistencias, eliminando posibles duplicados o datos incorrectos.

### 5.2.2 2. Vista:

La Vista se implementa mediante el framework para crear aplicaciones web interactivas Shiny (**?**) [1] que tiene un componente de Interface de Usuario (UI) donse se puede indicar el texto con el que se hará la bíusqueda y también permite a los usuarios aplicar filtros como área académica y rango de fechas durante. Posterior a la definición de los criterios del query, la Vista muestra las tablas con los resultados de las búsqueda, las representaciones visuales como los “mapas del conocimiento”, así como las recomendaciones de documentos similares.

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

### 5.2.3 3. Controlador:

El Controlador se implementa mediante el el framework para crear aplicaciones web interactivas Shiny (**?**) que tiene un componente denominado Server que es el responsable de manejar las interacciones del usuario, procesar 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 hace el llamado a la API donde se genera el embedding del query y se determina la relevancia de los documentos recuperados y en paralelo remite el query a la base de datos. El Controlador se encarga de aplicar el re ordenamiento para mostrar primero los resultados más relevantes. En el Controlador se genera la estructura de datos necesaria para representar los mapas del conocimiento.

## 5.3 Ciclos de Desarrollo:

### 5.3.1 Obtención del Conjunto de Datos:

Recuperación

### 5.3.2 Extracción y Clasificación de Información del Conjunto de Datos:

Extracción y clasificación

### 5.3.3 Prototipo de buscador:

### 5.3.4 Integración de componentes del software:

### 5.3.5 Buscador semántico:

## 5.4 Pruebas:

### 5.4.1 Funcionales:

### 5.4.2 Rendimiento:

### 5.4.3 Relevancia:

# Capítulo 6

# Conclusiones:

Este es el capítulo de las Conclusiones

## 6.1 Contribución:

## 6.2 Trabajos Futuros:

Finalizar con que este libro en sus dos versiones pdf y html se generó con bookdown

# Bibliografía


(2016). *World Development Report 2016: Digital Dividends*. World Bank, Washington, DC. OCLC: ocn915488955.

Aggarwal, C. C. (2018). *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.

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.

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.

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., Cheng, J., Allaire, J., Sievert, C., Schloerke, B., Xie, Y., Allen, J., McPherson, J., Dipert, A., and Borges, B. (2023). shiny: Web application framework for r.

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.

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.

Eisenstein, J. (2019). *Introduction to natural language processing*. Adaptive computation and machine learning. The MIT Press, Cambridge, Massachusetts.

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.

Harman, D. (2011). *Information retrieval evaluation*. Number 19 in Synthesis lectures on information concepts, retrieval, and services. Morgan Claypool, San Rafael, Calif.

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.

Howard, J. and Ruder, S. (2018). Universal language model fine-tuning for text classification.

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.

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.

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.

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*.

Penedo, G., Malartic, Q., Hesslow, D., Cojocaru, 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.

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.

Salakhutdinov, R. and Hinton, G. (2009). Semantic hashing. *International Journal of Approximate Reasoning*, 50(7):969–978.

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.

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.

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.