**INSTITUTO POLITÉCNICO NACIONAL**
ESCUELA SUPERIOR DE INGENIERÍA MECÁNICA Y ELÉCTRICA
UNIDAD ZACATENCO

# TESIS

**"RECUPERACIÓN DE INFORMACIÓN EN TEXTOS EN ESPAÑOL"**

**Que para obtener el título de:**

**INGENIERO EN COMUNICACIONES Y ELECTRÓNICA**

**P R E S E N T AN:**

**DANIEL ENRIQUE VALENCIA FLORES**
**OMAR ZÚÑIGA HERNÁNDEZ**

**ASESORES:**

**M. EN C. JUAN PABLO FRANCISCO POSADAS DURAN**
**M. EN C. GABRIELA POSADAS DURAN**
**M. EN C. BEATRIZ ADRIANA JAIME FONSECA**

**México, D. F. Junio del 2014.**

# INSTITUTO POLITÉCNICO NACIONAL
## ESCUELA SUPERIOR DE INGENIERÍA MECÁNICA Y ELÉCTRICA
## UNIDAD PROFESIONAL "ADOLFO LÓPEZ MATEOS"

# TEMA DE TESIS

**QUE PARA OBTENER EL TITULO DE** INGENIERO EN COMUNICACIONES Y ELECTRÓNICA

**POR LA OPCIÓN DE TITULACIÓN** TESIS COLECTIVA Y EXAMEN ORAL INDIVIDUAL

**DEBERA (N) DESARROLLAR** C. DANIEL ENRIQUE VALENCIA FLORES
C. OMAR ZUÑIGA HERNANDEZ

**"RECUPERACIÓN DE INFORMACIÓN DE TEXTOS EN ESPAÑOL"**

DESARROLLAR UNA METODOLOGÍA PARA LA RECUPERACIÓN DE INFORMACIÓN PARA TEXTOS EN ESPAÑOL, EMPLEANDO EL MODELO ESPACIO VECTORIAL CON MEDIDA TF-IDF USANDO HERRAMIENTAS DE PROCESAMIENTO DE LENGUAJE NATURAL.

- INTRODUCCIÓN
- MARCO TEÓRICO
- ESTADO DEL ARTE
- METODOLOGÍA
- PRUEBAS Y RESULTADOS
- CONCLUSIONES

**MÉXICO D.F. A 16 DE ABRIL DE 2015**

**A S E S O R E S**

**M. EN C. JUAN PABLO FRANCISCO POSADAS DURAN**

**ING. BEATRIZ ADRIANA JAIME FONSECA**

Posadas Durán G.

**M. EN C. GABRIELA POSADAS DURAN**

RAMIREZ LORENA

**ING. PATRICIA LORENA RAMÍREZ RANGEL**
**JEFE DEL DEPARTAMENTO DE**
**INGENIERÍA EN COMUNICACIONES Y ELECTRONICA**

## AGRADECIMIENTOS

*Los logros de una organización son el resultado de los esfuerzos combinados de cada individuo.-Vince Lombardi.*

Agradecemos al M. en C. Juan Pablo Francisco Posadas Duran por su apoyo, atención, enseñanzas, disposición y dedicación en el desarrollo de este trabajo.

Agradecemos a la M. en C. Beatriz Adriana Jaime Fonseca por su dedicación, apoyo y enseñanzas en el curso "Tópicos Selectos de Ingeniería".

Agradecemos a la Dra. María Elena Acevedo Mosqueda por el tiempo, dedicación y enseñanzas en el curso "Tópicos Selectos de Ingeniería".

Agradecemos a los sinodales quienes estudiaron nuestra tesis y la aprobaron.

Agradecemos al Instituto Politécnico Nacional y a la Escuela Superior de Ingeniería Mecánica y Eléctrica Unidad Zacatenco por todos los recursos otorgados en el desarrollo de nuestra formación como profesionistas.

*Un poco más de persistencia, un poco más de esfuerzo y lo que parecía un fracaso sin esperanza se podría convertir en un glorioso éxito.-Elbert Hubbard*

Quiero expresar mi más profundo agradecimiento a mis padres: J. Daniel Alberto Valencia Solórzano y Teresa Flores Molina, por todo su cariño, amor, entrega, consejos, apoyo económico y moral durante toda mi formación académica.

Le doy las gracias a mis hermanos: José Alberto Valencia Flores y Alexis Valencia Flores por su apoyo incondicional.

Agradezco a mi abuelita Angela Molina de la O por su cariño, amor y todo el apoyo que me ha brindado toda mi vida.

Le doy gracias a Erika Navarro por brindarme su amistad, apoyo, consejos y palabras de aliento cuando estaba a punto de desistir.

Agradezco al Ing. Juan Díaz Díaz y a la Lic. María Elena Robles Molina por todos sus consejos y apoyo durante mi formación como profesionista.

Agradezco a mis maestros quienes nunca renunciaron al enseñarme, aun sin importar que muchas veces no ponía atención en clase o no entendía, a quienes siempre me indujeron en el mundo del saber.

Agradezco a mis tías: Gela, Sonia, Elizabeth, a  mis primos en especial a Ricardo Díaz y Ana Karen Buendía por el apoyo brindado a lo largo de este trabajo.

*Daniel Enrique Valencia Flores.*

*"Hay una fuerza motriz más poderosa que el vapor, la electricidad y la energía atómica: la voluntad."*
*Albert Einstein*

Yace 6 años que entre al Instituto Politécnico Nacional, todo un mundo nuevo que habría que conocer y por lo tanto también que agradecer.

Quiero dar las gracias a mis padres José Francisco Zúñiga Santos y María del Carmen Hernández Barrera por ofrecerme un mundo nuevo a través de la educación, su cariño, apoyo moral, consejos incondicionales y preocupación para llegar a ser una persona y profesionista de bien. Gracias por haberme dado lo más grande que puede haber.

Agradecer a mis hermanos, Nacho, Delia, Judith ,Paco y familia que me apoyan a seguir fuerte en mis convicciones y a mi sueño, su deseo de mi desarrollo profesional y motivarme a seguir adelante en los momentos más difíciles, ayudarme a mantenerme fuerte en el duro camino del estudio de la ingeniería.

Agradecer a todos mis amigos de clase, de trabajo y amigos de infancia, por su compresión, compartir sus experiencias conmigo y su ayuda.

Agradezco a mi compañera Jovana, por tenerme paciencia, ofrecerme sus palabras de aliento y apoyarme a seguir mi sueño.

También agradezco a los profesores con los que he convivido durante mi estancia en el Instituto Politécnico Nacional, compartir sus sabios consejos, orientación y su atención hacia mi persona.

Agradezco todos los problemas que surgieron en mi camino, considero la mejor forma de superarse uno mismo.

*Omar Zúñiga Hernández*

# RESUMEN

La integración de los equipos de cómputo y el internet a nuestra a vida cotidiana ha generado sin duda beneficios, destacado algunos como proporcionar la capacidad de compartir información a cualquier parte del mundo.

Hoy en día tener acceso a las grandes cantidades de información no es difícil, solo con entrar a internet o una biblioteca virtual, podemos conseguir la información cual fuera que sea, sin embargo, la constante creación y el aumento de información contenida en documentos, artículos y textos, han hecho que la recuperación de información sea compleja y laboriosa. La pregunta actual a la que debemos responder es ¿Cómo acceder al documento que necesito?

En este trabajo de tesis se plantea una metodología para la Recuperación de Información en textos en español utilizando herramientas de Procesamiento de Lenguaje Natural, la metodología consiste en analizar el contenido de un conjuntos de textos, así como la petición realizada por el usuario, al ser comprados ambas tramas , la metodología ofrecerá una serie de documentos presentados de manera relevante indicando los posibles textos que en su contenido pueda satisfacer la necesidad del usuario.

El sistema fue probado en textos obtenidos de la red, tales ejemplos son, noticias y artículos. La metodología es evaluada a través de *Baseline*, así como medidas de precisión, *Recall*, para conocer la capacidad de recuperación de la metodología propuesta.

## ABSTRACT

The integration of computer equipment and the Internet into our daily lives has certainly generated benefits,
Susch as providing the ability to share information anywhere in the world.

Today access lo large a mounts of information is not difficult, because only access the internet with a virtual library or we can gel any information; however the constant development and enhancement of information contained in documents, articles and books have made the recovery of information is complex and laborious. Current question we must answer is how to access the document I need?.

This thesis presents a methodology for information retrieval tools texts in Spanish using natural language processing , the methodology is to analyze the contents of a set of texts as well as the request made by the user , being compared both plots , the methodology will offer a number of documents relevant way possible texts indicate that its contents can meet the user's needs.

The system was tested on texts obtained from the network, such examples are news and articles. The methodology is evaluated through baseline and measures precision , recall , and f1 to know the capacity of the proposed methodology.

# GLOSARIO

**RI**: Recuperación de Información.
**IR:** Information Retreival (RI).
**PLN:** Procesamiento de lenguaje natural.
***Corpus*:** Colección, base de datos o textual.
**Indexación:** Es una construcción de estructuras de datos (índices), que almacena y soporta búsquedas eficientes.
**Interface**: Es el medio con que el usuario puede comunicarse con una máquina, un equipo o una computadora, y comprende todos los puntos de contacto entre el usuario y el equipo.
**SRI:** Sistema de Recuperación de Información.
***Query*:** Consulta del usuario.
**Recuperación *ad-hoc*:** Búsqueda donde el usuario formula una consulta en un lenguaje, el sistema le evalúa y responde.
**Perfil:** Es un entorno personalizado específicamente para un usuario.
**Lema:** es la reducción de una palabra a su raíz.
**Similitud:** Es la medida de la interrelación existente entre dos palabras cualesquiera en un texto.

**Tokenizacion**: Es encargado de segmentar la frase en palabras simples o compuestas.
Sintagma: Es una unidad gramatical formada por palabras que ejercen una función sintáctica determinada (nominal, adverbial, adjetival, etc.).
**Algoritmo de búsqueda**: Es un conjunto de instrucciones orientadas para localizar un elemento con ciertas propiedades dentro de una estructura de datos.
**Análisis sintáctico**: Relaciones de concordancia y jerarquía que guardan las palabras agrupándose entre sí en sintagmas, oraciones simples y compuestas.
**Sustantivo:** *Es una categoría gramatical que sirve para nombrar a todo tipo de sujeto u objeto.*
***Semántica:** Se ocupa del significado de las palabras, los enunciados y los textos.*
***Palabras vacías**: Son palabras funcionales o de bajo contenido semántico (menor significado).*
***Morfología:** Se ocupa de clasificar y explicar el funcionamiento y significado de las variaciones de forma en las palabras dentro de la estructura de la lengua.*
***Texto plano**: Es un archivo informático compuesto únicamente por texto sin formato (negritas, cursivas, imágenes, etc.) sólo caracteres, lo que lo hace también legible por humanos.*
***Python**: Es un lenguaje independiente de plataforma (Windows, Linux) y orientado a objetos, preparado para realizar cualquier tipo de programa.*
***Diccionario:** Contenedor de almacenamiento de claves-valor. Un diccionario es muy diferente a una típica lista en Python ya que el índice no rellena automáticamente para cualquier elemento dado en el diccionario.*
***Vector:** Es una zona de almacenamiento continuo, que contiene una serie de elementos del mismo tipo, los elementos de una matriz. Se puede ver como un conjunto de elementos ordenados en fila.*
***Ponderación:** Es el peso o la relevancia que tiene un objeto.*

## CONTENIDO

## CAPÍTULO 1. INTRODUCCIÓN



## CAPÍTULO 2. MARCO TEORICO



## CAPITULO 3. ESTADO DEL ARTE

**CAPITULO 4.METODOLOGÍA**



**CAPITULO 5. PRUEBAS Y RESULTADOS**



**CAPITULO 6. CONCLUSIONES**

## CONTENIDO DE FIGURAS

# CONTENIDO DE TABLAS

# CAPÍTULO I. INTRODUCCIÓN

## 1.1 ANTECEDENTES.

Durante años, el ser humano ha tenido la necesidad describir el mundo que lo rodea y plasmar esta descripción en un formato que le permita ser trascendida. En este sentido, la escritura ha sido uno de los mecanismos que le ha permitido alcanzar este objetivo.

Con el paso de los siglos la escritura ha sido característica fundamental para el desarrollo de las culturas civilizadas, prueba de esto, es el conocimiento que se ha creado y sigue presente en libros, artículos, notas, documentos, noticias todos ellos almacenados en bibliotecas o colecciones de archivos, que ahora con la incorporación de internet, es posible encontrar información en formatos electrónicos y bibliotecas digitales.

Actualmente se han desarrollado grandes avances tecnológicos, científicos y sociales; que a su vez ha hecho que la cantidad de información crezca de una manera acelerada. Además la tecnología ha hecho posible compartir ésta información a nivel mundial.

Hoy en día existe un fenómeno llamado **sobrecarga de información**, el cual consiste en un estado en el que la cantidad de información es tan grande que su análisis y administración sea difícil de realizarla, impactando de manera negativa la eficiencia de recuperación de información textual, es decir, obtener la información que se solicita de manera rápida y que cumpla con los requerimientos de la búsqueda, por lo tanto, que típicamente requiere de mayor esfuerzo en el procesamiento de la información, dedicando más tiempo y mayor concentración de parte de la persona para localizar la información necesaria de una gran gama de documentos disponibles con la finalidad de cumplir su necesidad.

Ante este panorama, una estrategia para administrar eficientemente la información es fundamental, en este sentido, para ofrecen soluciones a través de métodos en Recuperación de Información asistido por Procesamiento de Lenguaje Natural.

El Procesamiento en Lenguaje Natural aplicado en métodos de Recuperación de Información proporciona los elementos necesarios para que la computadora "comprenda" el lenguaje natural empleado en los textos, en la formulación de la consulta y en la comparación de la petición del usuario con el texto, la maquina decidirá que documentos son mejores y que puedan ser ofrecidos como solución a la necesidad de información reduciendo la intervención analítica del usuario.

Las aplicaciones y desarrollos que se le puede dar al modelo propuesto son variadas, tales como: clasificadores automáticos de información, elaboración automática de resúmenes más confiables, filtros para información no deseada, detección de patrones psicológicos en textos, etc. Son algunos ejemplos de los beneficios y alcances que sin duda puede llegar a adquirirse.

## 1.2 PLANTEAMIENTO DEL PROBLEMA

La capacidad de comunicarse que tienen los seres humanos está ligada a su capacidad cognitiva (su inteligencia), comparten experiencias, conocimientos desarrollados a partir de razonamientos, acciones y sentimientos, implícitamente asociados con el funcionamiento de su cerebro; también depende de fenómenos sociales e históricos.

La lectura de los textos es la mejor forma de obtener información, el usuario interviene directamente en su búsqueda, localizando en los textos la información que requiere y cubra su necesidad, sin embargo, la capacidad para manejar demasiada información está acotada por factores físicos, mencionar lugares y situaciones en el que no es muy conveniente hacer los análisis tradicionales, por ejemplo: clasificar grandes cantidades de libros para una biblioteca; desde un principio cabe mencionar ,que la persona que realice esta tarea debe saber intuitivamente que información necesita, debe estar capacitada con técnicas de lectura, técnicas de resumen, así como conocimiento de lenguaje documental y tener referencia de las características necesarias para tomar una decisión acertada en dicha clasificación, apoyándose sobre el contenido de un texto , cabe destacar que la actividad no

es imposible de realizar, pero al tratar de procesar tal cantidad de información de la que podemos hace ardua esta tarea y utiliza más tiempo del que disponemos.

Actualmente el almacenamiento de información en las computadoras, así como su capacidad de procesar es vital y de gran ayuda, los avances tecnológicos permiten que grandes cantidades de documentos sean almacenados electrónicamente, con el objetivo de que una persona con la necesidad de información, tenga la facilidad de encontrar y acceder a un conjunto extenso de textos variado en temas previamente almacenados idóneos para cubrir necesidades. Actualmente el problema radica en a partir de esas vastas colecciones contestar las siguientes preguntas: ¿Cómo accedo a la información que necesito?, ¿Qué documentos tienen la información que busco?

La recuperación de información es descrito como un conjunto de tareas que ayudan al usuario a localizar y acceder a la información que le es pertinente para la resolución de un problema; RI nos ayuda a resolver las preguntas anteriormente mencionadas. Aunque existen métodos automatizados basados en coincidencias de términos, el resultado ofrecido en la recuperación de información se ha vuelto ineficaz por características propias de los textos, los cuales, muchos de ellos no se presentan en forma estructurada, la información mostrada a veces es redundante, los textos se actualizan en cada momento, la calidad de las publicaciones es variable, puede presentar errores ortográficos; causando que la recuperación de información en estos métodos sean limitados y ocasionando que el usuario tome más responsabilidad en su búsqueda.

Por otra parte, es importante mencionar que los resultados obtenidos dependen de factores relacionados con el usuario, entre estos, se encuentra la dificultad de expresar correctamente su necesidad de información y que los métodos automatizados de Recuperación de Información, les son complicados de asimilar, a pesar de que ambos se expresan en lenguaje natural, por tal motivo la introducción del Procesamiento del Lenguaje Natural mejora el entendimiento hombre-máquina, que el cual a través del lenguaje empleado, la máquina entienda la expresión y no solo como una representación binaria. Otro factor es que los resultados son subjetivos, ya que estos variaran de usuario a usuario.

El enfoque clásico de la Recuperación de información se basa en el uso de modelos probabilísticos, booleanos o vectoriales, siendo procesos meramente matemáticos, presentando desventajas sobre el contenido de la información, expresada en lenguaje natural.

## 1.3 JUSTIFICACIÓN

Vivimos en un mundo donde la comunicación es imprescindible, y donde el principal objetivo es transmitir, recibir y procesar información. Pero nos encontramos con el hecho de que los sistemas de información son incapaces de procesar todo el contenido en lenguaje natural que es el utilizado para representar el conocimiento, haciendo difícil el procesamiento y búsqueda de información en las computadoras de una manera concreta.

Por tal motivo, se propone un método que sea capaz de recuperar información de una colección de textos en español, con apoyo de herramientas en Procesamiento de Lenguaje Natural, con el propósito de resolver una petición de información por parte de un usuario de manera más eficiente en comparación con los métodos estadísticos tradicionales

El método a desarrollar está dirigido a textos no estructurados, por no presentar restricciones y formas rígidas, este puede contener redundancia, ambigüedad, errores ortográficos, gramaticales, por lo cual es necesario proporcionar un tratamiento previo al texto.

El usuario al efectuar una búsqueda acerca de un tema determinado, intuye los conceptos que le son relevantes, que hacen que el contenido de la información le sea útil, sin embargo, los términos que se manejan son subjetivos. El metodología asimila la petición del usuario, de tal modo, que esta pueda procesar su necesidad de una manera más eficiente y le presente resultados que le sean relevantes de acuerdo a su consulta.

A largo plazo, la metodología proporcionará elementos que sirvan como base a más herramientas enfocadas a Recuperación de Información en textos en español, donde la información contenida sea analizada y satisfaga la necesidad del usuario.

## 1.4 HIPÓTESIS

Mediante herramientas para el uso Procesamiento de Lenguaje Natural es posible proporcionar tratamiento a la información contenida en textos en español, representando los términos de acuerdo a sus cargas sintácticas así como su frecuencia, indicando la esencia informativa de un texto haciendo comparable la relevancia entre los mismos, provocando que aumente la efectividad de búsqueda de información útil en la metodología propuesta, capaz de cubrir la necesidad de Recuperación de Información del usuario.

## 1.5 OBJETIVOS

### 1.5.1 OBJETIVO GENERAL

Desarrollar una metodología para la recuperación de información para textos en español, empleando el modelo espacio vectorial con medida tf-idf usando herramientas de Procesamiento de Lenguaje Natural.

### 1.5.2 OBJETIVOS PARTICULARES

- Pre procesar el texto en español.
- Aplicar análisis sintáctico a las palabras que conforman el texto.
- Discriminación de términos de interés (Sustantivos).
- Realizar pruebas del sistema con diferentes textos.

## 1.6 ALCANCES

En esta investigación la metodología propuesta pretende que el sistema de Recuperación de Información, e implicando técnicas de procesamiento de lenguaje natural, logre mejorar los resultados en la obtención de documentos relacionados a las consultas hechas por un usuario y por lo tanto es usuario pueda satisfacer su necesidad.

Para la propuesta de investigación, se desarrollara un Sistema de recuperación de información de textos en español basado en el modelo vectorial usando medida tf-idf, así como procesamiento de lenguaje natural. El modelo vectorial provee las operaciones de presentación, comparación y resultados enlistados en orden relevante de los textos; el mecanismo PLN provee las técnicas de análisis del contenido textual que por sus características requiere de análisis profundos.

La metodología que se propone puede ser utilizada como herramienta de comparación para futuros métodos similares.

## 1.7 ESTRUCTURA DEL TRABAJO

En el capítulo uno establece una pequeña introducción a la recuperación de información de textos, se mencionan algunas problemáticas actuales para procesar la información así como su obtención; en este apartado se establecen los principios de desarrollo del presente trabajo, planteamiento de problema, justificación, hipótesis, objetivos. Por consiguiente en el capítulo dos, hace reseña al campo de recuperación de información, presentado conceptos básicos, para entender el funcionamiento y objetivos de la recuperación de información, se muestran los modelos utilizados conocidos como clásicos y su operación, tales como la comparación ,presentación de las consultas y resultados. Se describen algunas medidas de evaluación. De misma manera en esta sección se proporcionan conceptos en Procesamiento de Lenguaje Natural, comprendiendo su objetivo y sus aplicaciones.

La sección número tres está dedicada a presentar algunos ejemplos de programas actuales recuperadores de información, se describe algunas de sus características funcionales, haciendo hincapié en el método de comparación y la obtención de similitud entre el documento y la petición realizada por el usuario; por otra parte el capítulo número cuatro, consta de mostrar el desarrollo y elaboración para obtener la metodología propuesta, un sistema de recuperación de información de textos en español, basado en modelo vectorial con medida tf-idf, y uso de herramientas de procesamiento de lenguaje natural, es utilizado Python, como lenguaje de programación para crear el algoritmo del sistema propuesto.

En el capítulo final se presentan los resultados del método recuperación de información en textos en español, se presentan pruebas del sistema funcionando, así como las conclusiones del trabajo desarrollado.

# CAPÍTULO 2. MARCO TEÓRICO

## 2.1 RECUPERACIÓN DE INFORMACIÓN

La Recuperación de Información (*Information Retrieval, en inglés)* no es un área nueva, ha pasado por un proceso de largo desarrollo e innovación [Kowalski, 2011], girando alrededor de un elemento importante, el valor de la información. Por lo tanto, para un usuario es de gran importancia disponer del contenido de un texto de manera precisa y que cubra la necesidad de este en el menor tiempo posible.

Actualmente se dispone de varios métodos y modelos para recuperar información, tales como, Modelo Booleano, Modelo Probabilístico, Modelo Vectorial [Baeza, 1999], aplicados ampliamente en bases de datos documentales, bibliotecas, páginas web o todo aquel lugar donde haya grandes bases de información y por lo consiguiente gran necesidad de procesamiento y búsqueda.

Hoy en día la Recuperación de Información apoya en diferentes áreas y se enfoca a una gran gama de aplicaciones como son: extracción de información, búsquedas Web, respuestas a preguntas, organización automática (*clustering),* elaboración de resúmenes (sumarization), clasificación, detección de novedades, entre otros, ofreciendo diferentes soluciones y desarrollando nuevas herramientas para cada situación [Bordignon, 2007].

El proceso de recuperación se ha regido mediante los siguientes conceptos fundamentales: almacenamiento, organización, acceso, búsqueda, recuperación, necesidad, precisión y relevancia, entonces la Recuperación de Información puede ser concretada como un conjunto de tareas por las cuales un usuario puede localizar, acceder y recuperar los recursos informativos que previamente han sido almacenados [Crotf, 1987], que son útiles y favorezcan a solucionar un determinado planteamiento o necesidad.

Finalmente mencionar que el objetivo principal de un sistema de recuperación, consiste en presentar al usuario la información que satisfaga su necesidad, satisfacer no necesariamente se refiere a mostrar toda la información sobre un tema especificado, sino mostrar lo suficientes textos que pueden ser relevantes y útiles para el usuario.

### 2.1.1 CONCEPTOS BÁSICOS

Algunos conceptos que se utilizan en el ámbito de Recuperación de Información (RI) es necesario entenderlos, ya que son definiciones de uso común para este trabajo.

En RI se define como *corpus* todo aquel documento que pertenece a una colección o base de datos textual o documental [Bordignon, 2007]. Algunas características notables que tienen estos documentos y nombrados como no estructurados es no poseer atributos ni dominios conocidos, son textos libres de cualquier forma rígida, no clara y difícil de crear [Baeza, 1999], por ejemplo: artículos, noticias, ensayos, notas, libros y documentos, elementos los cuales trabaja Recuperación de Información, en cambio se puede diferenciar de las bases de datos, que son fuertemente estructurados, dominios conocidos y son expresados comúnmente en tablas o matrices, objetos concernientes a la Recuperación de Datos . En la tabla 2.1 se logra observar las diferencias entre Recuperación de Información y Recuperación de Datos.

Tabla 2.1.Comparación entre Recuperación de Datos y Recuperación de Información.

| Características | Recuperación de Datos | Recuperación de Información |
|---|---|---|
| Acierto | Exacto | El mejor |
| Modelo | Determinista | Probabilístico |
| Lenguaje de consulta | Estructurado | Natural |
| Especificación para consultar | Precisa | Imprecisa |
| Error en resultados | Sensible | Insensible |

Mientras tanto, un Sistema de Recuperación de Información o SRI es un programa de software que almacena y gestiona documentos. Es descrito como un conjunto de documentos expresados en forma de texto, un conjunto de peticiones o consultas (*query*, en inglés) y un mecanismo que principalmente ayude a determinar la necesidad de información expresada por un usuario a partir de una petición, además que el sistema

retome y ofrezca los documentos relevantes, que son aquellos que pudieran contener la información requerida por el usuario.

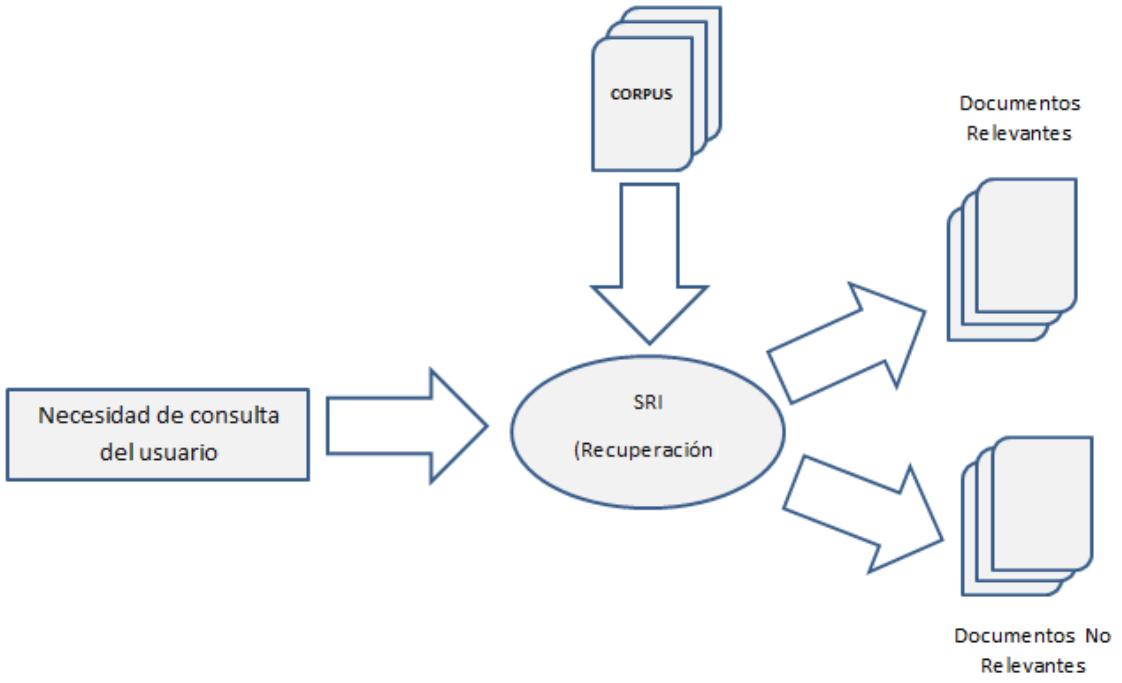



Figura 2.1. Esquema de un Sistema de Recuperación de Información.

Dentro de los principios fundamentales en el contexto en RI es de obtener la información en base a la similitud de la consulta, es decir, proporcionar los documentos que contengan los términos clave más parecidos y cercanos a los de la consulta propuesta por el usuario, para ello el SRI toma ciertos parámetros para mejorar la calidad de la respuesta. Uno de ellos es la relevancia, este término nos define que tan importante o significativo es un documento perteneciente a un campo o materia, que es acotado por los términos de la consulta del usuario. Un documento relevante será aquel que en su contenido sea capaz de satisfacer la necesidad o que tanta información de valor tiene para el usuario, normalmente los documentos son presentados en listas de acuerdo a su relevancia. Cabe mencionar que este término es subjetivo [Rijsbergen, 1979], es decir, el usuario de acuerdo a su percepción y grado de necesidad, definirá que tan conveniente y útil le es un documento que ha sido recuperado de un SRI (Figura 2.1). En cambio definimos como pertinencia, aquellos documentos que proporcionan al usuario la información que se ajusta a su necesidad informativa.

La precisión es una medida apoyada de la relevancia, nos mide el porcentaje de documentos recuperados que resultan relevantes de acuerdo a la consulta, se expresa como la razón del total de documentos relevantes recuperados entre el total de documentos recuperados [Sagayam, Srinivasan, Roshni,2012], la precisión será máxima (valor de 1) cuando solo se recuperan documentos relevantes. Cuantos más documentos relevantes y útiles tenga un conjunto más preciso será. Dentro del campo en RSI también se emplea el término de exhaustividad conocido como *recall* en inglés, se entiende como la división de los documentos relevantes recuperados entre los documentos relevantes [Sagayam, et al,2012] para que un documento sea exhaustivo los términos clave que describe un texto debe estar en muchos documentos, comparando con la precisión, en este las palabras clave aparecen en pocos documentos y con nula en el resto del corpus. La proximidad se utiliza para restringir la distancia permitida en un elemento entre dos términos de búsqueda, entre más estrechos se encuentran los términos en un texto es más probable que se parezcan en la descripción del mismo, es decir si las mismos términos clave aparecen en ambos textos, los documentos pueden ser parecidos en contenido. La precisión, relevancia, proximidad así como exhaustividad son utilizados como medidas para mejorar la calidad un SRI, de misma manera indican el grado en que los documentos pueden ser recuperados o rechazados a partir de la comparación de la consulta y el texto.

Por ultimo cabe mencionar la medida F1[Sidorov, 2013], que combina la precisión y *Recall*, se le ese nombre porque se le da el mismo peso a la precisión y *Recall* igualándolos a 1.

En la siguiente tabla se muestra las fórmulas para calcular precisión y *recall*.

Tabla 2.2. Fórmulas de Precisión y Recall.

| Concepto | Expresión Matemática |
|---|---|
| Precisión | $P = \frac{Numero\ de\ Documentos\ Relevantes}{Numero\ de\ documentos\ Recuperados}$ |
| Recall (exhautividad) | $R = \frac{Numero\ de\ Docuemntos\ Recuperados}{Numero\ de\ Documentos\ Relevantes}$ |
| F1 | $F1 = \frac{2(P)(R)}{P+R}$ |


Figura 2.2. Recuperación de documentos de acuerdo a la relevancia.

Además de la medidas mencionadas, la evaluación del sistema se hace a través de corpus especiales llamados colecciones de prueba, en este caso, el corpus es enfocado a sistemas de recuperación de información ad-hoc.

Las colecciones de prueba se encuentran estandarizadas por organizaciones, que son enlistadas a continuación en la siguiente tabla.

Tabla 2.3 Lista de colecciones de ensayos de prueba y características.

| Colección de ensayos | Características |
|---|---|
| Colección de Cranfield | Pionero en usar ensayos de prueba, permite medidas precisas, pero hoy en día este sistema no funcionan y ocupadas para pruebas piloto más elementales; contiene 1938 artículos acerca de aerodinámica y un conjunto de 225 consultas |
| TREC (*Text Retrieval Conference*) | Evalúa desde 1952,las más conocidas comprenden entre 1992 y 1999, contiene 6 CD´s que contiene 1,890,000 , 450 juicios de necesidades de relevancia, para colecciones de prueba individuales se utiliza subconjuntos de estos temas (artículos de periódicos) |
| GOV2 | Incluye una colección de 25 millones de páginas web |
| NTCIR | Ha construido varias colecciones de ensayo de dimensiones similares a las colecciones de TREC, centrándose en idiomas de Asia Oriental y Recuperación de información entre lenguajes |
| CLEF | Para la clasificación de texto, la colección de prueba más utilizada ha sido la colección Reuters-21578 de 21578 artículos de agencia de noticias; más recientemente publica corpus volumen 1 que consta de 806.791 documentos, su escala y rica anotación hace que sea una mejor base para futuras investigaciones. |
| 20 NEWSGROUPS | Esta es otra colección de clasificación de texto ampliamente usado, recogido por Ken Lang. Consta de 1.000 artículos de cada uno de 20 grupos de noticias Usenet (el nombre del grupo de noticias que se considerará la categoría). Después de la eliminación de artículos duplicados, ya que se utiliza por lo general, contiene 18.941 artículos. |

La interacción del usuario con un SRI es fundamental en la recuperación de información, ya que la necesidad puede ser planteada de diferentes formas, una correcta interacción con el SRI puede brindar facilidades al usuario. La tarea de recuperar información puede dividirse como Recuperación Inmediata y Recuperación Diferida [Bordignon, 2007]. En la primera describe en que el usuario establece su consulta en busca de información y el sistema arroja documentos considerándolos como relevantes. Hay dos modalidades para esta técnica, el primero conocido como Búsqueda o *ad-hoc* ,modo en el que un usuario expresa en su lenguaje su necesidad, el SRI lo acepta, evalúa y responde con documentos seleccionados de un *corpus*, en este procedimiento el usuario sabe con certeza su necesidad y como expresarla; el otro modo conocido como Navegación o *Browsing* , la interfaz del

SRI dispone de campos relacionados al tema, por los cuales el usuario navega a través de su estructura, busca y obtiene referencia de documentos que le sean útiles .En esta técnica el usuario no sabe con certeza lo que busca o cómo definir su necesidad, incluso durante el *query* esta puede ir definiendo su necesidad. El segundo método es Recuperación Diferida, en esta opción el usuario especifica su necesidad y el SRI entregará documentos continuos que concuerden a su consulta, se emplea el ruteo y filtrado que junto con la necesidad del usuario actúan para generar un perfil de los documentos, cada vez que arribe un documento este es comparado con el perfil y si son similares es considerado relevante y por lo tanto se muestra al usuario, si no concuerda con el perfil el documento es desechado.

En la actualidad encontramos problemas de software y hardware asociados a la recuperación y presentación de los documentos, es sustancial diseñar y construir sistemas que mejoren el rendimiento del procesamiento de las consultas hechas por el usuario que son expresadas en su leguaje y desarrollar procedimientos en el que un SRI pueda "comprender" la necesidad del usuario, garantizando mayor calidad de los objetos recuperados.

## 2.2 SISTEMAS DE RECUPERACIÓN DE INFORMACIÓN

El progreso de la Recuperación de la información ha estado ligado a la evolución de la potencia de procesamiento de las computadoras, ocasionando que la demanda y expectativas de los usuarios para encontrar rápidamente cualquier información aumenten, impulsando el análisis teórico y desarrollo de nuevas tecnologías para satisfacer esa necesidad.

Un Sistema de recuperación de información tiene como objetivo la búsqueda de información relevante o un documento que pueda cubrir la necesidad de un usuario disminuyendo tiempo, sin embargo, se presentan algunos problemas como es, la diferencia que existe en el lenguaje del usuario y el que usa el autor para expresar su obra documental [Kowalski,2011]; el usuario tratara de exponer su necesidad de información en el lenguaje que utiliza mencionándolo en forma de consulta, además de la diferentes formas de

declarase la necesidad, ocasionan problemas para que un SRI pueda asimilar el vocabulario utilizado en ambas partes. El desempeño de recuperación depende de factores humanos, esencialmente del lenguaje y su expresión. En la Figura 2.3 se observa la comparación entre los conjuntos de vocabularios utilizados.

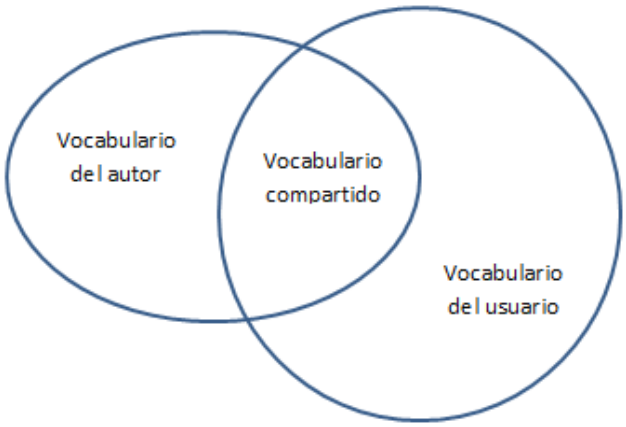



Figura 2.3 Dominios de Vocabularios

Para alcanzar el objetivo de un RI se implementan los siguientes pasos [Sharma, Patel, 2013]:

1) En el proceso de indexación se presentan de manera resumida los documentos, se hace fuera de línea, por lo que el usuario no está involucrado.
2) En el proceso de filtrado todas las palabras comunes y vacías, se quitan.
3) La búsqueda es el punto fundamental de un SRI, mediante modelos para recuperar información se obtienen los documentos relevantes que el usuario necesita.

Un SRI se apoya en tres procesos básicos capaces de mejorar el procesamiento de la información, búsqueda y recuperación. Estos procesos son: la presentación del contenido, consistiendo en la muestra lógica de los documentos; representar correctamente la necesidad del usuario, establecer la expresión de la necesidad de información en forma de consulta y por último, comparación de las representaciones, evalúa e indica que documentos son relevantes con respecto a la consulta. En la Figura 2.4 se observa el marco

general de un SRI, las formas cuadradas representan datos, mientras las redondeadas indican procesos.

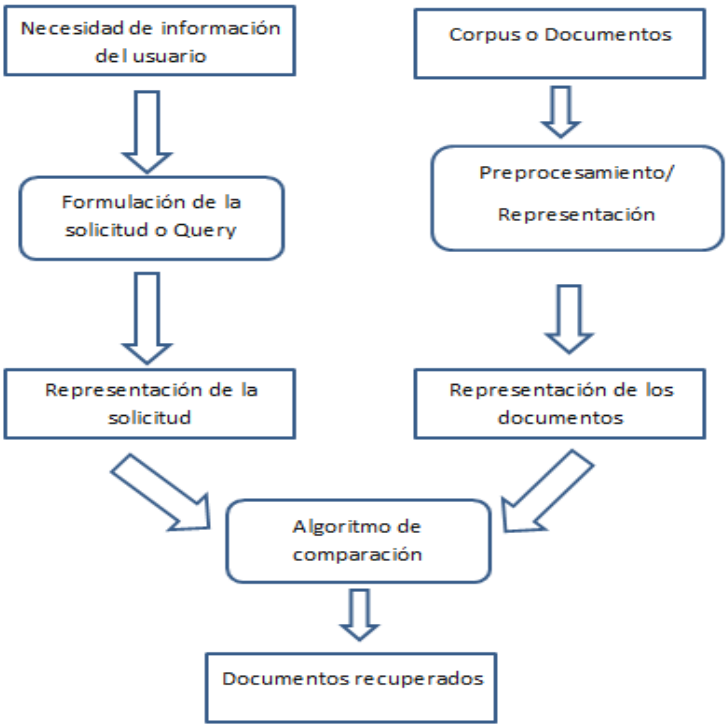



Figura 2.4. Marco general de un SRI

En muchos casos el texto es presentado como una secuencia lineal de caracteres, por lo tanto en muchos Sistemas de Recuperación de Información, por medio de algoritmos y técnicas, es común normalizar las secuencias de caracteres a formas más pequeñas, tales son el caso de las frases, posteriormente la frase es segmentado en palabras, discriminando signos ortográficos, espacios entre oraciones y palabras vacías, por ejemplo: conjunciones. Las palabras usadas para expresar dicha idea se determinan como unidades mínimas lingüísticas, agrupándolas como unidades semánticas útiles o de acuerdo a su frecuencia de aparición, posteriormente son almacenadas para ser usadas como referencia. Este conjunto de términos indican la mejor presentación del contenido de un texto.

Obtenidos los argumentos del texto en la forma necesaria de acuerdo al RSI, se representa el contenido con el propósito de que el sistema pueda procesarlo todo o algunos de los

elementos de dicho texto para luego ejecutar recuperación. En las etapas de preprocesamiento y procesamiento, el usuario no está en contacto con estas acciones, por eso es conocido como fuera de línea, el usuario no está directamente involucrado. En preprocesamiento y procesamiento son operaciones que se llevan a cabo con la finalidad de representar la información de tal modo que el proceso de recuperación sea óptimo creando índices para mejorar el almacenamiento y consulta para futuras exploraciones. También es conocido como Indexación.

La formulación de la solicitud de información, consiste en representar correctamente la necesidad del usuario de tal modo que pueda ser entendible para el SRI y pueda cumplir con su objetivo; cumplir la necesidad de información ,al término de la formulación se tiene la solicitud que de igual manera es una representación para el SRI .Esta tarea simboliza la interacción del usuario con el sistema, además de proporcionar beneficios como ayudar al usuario a formular idóneamente su solicitud, continuamente se ofrece una mejor compresión de la necesidad de información por parte de este [Ferreira, Atkinson, 2009].

El nivel de Recuperación basa su funcionamiento en modelos que hacen la comparación de las representaciones de la solicitud elaborada por parte del usuario con la representaciones de los textos previamente procesados, surgiendo de esta operación una serie de elementos ordenados jerárquicamente de manera descendente de acuerdo a su contenido, el cual el usuario recorre en búsqueda de la información que necesite.

El SRI arroja una serie de documentos ordenados de acuerdo a su relevancia, los textos colocados al principio de la lista son considerados más relevantes, mientras descienda la posición de los elementos de la lista disminuirá su relevancia. El SRI ayuda a disminuir el tiempo de lectura del usuario ya que solo revisara los documentos colocados en las primeras posiciones, estos serán los textos más relevantes y pertinentes devueltos por el sistema.

La base del funcionamiento de la Recuperación de Información esta descrito por los modelos que puedan ser utilizados, puede ser sencillos, complejos y eficaces, ayudando a representar y comparar de la mejor manera la necesidad del usuario y un documento.

## 2.3 MODELOS DE RECUPERACIÓN DE INFORMACIÓN

Son muchos enfoques que se han desarrollado para mejorar el objetivo de la Recuperación de Información, se ha utilizado desde el elemento más básico y vías más simples como es el Modelo Booleano hasta el empleo de técnicas apoyadas en inteligencia artificial, como es Redes Neuronales, Algoritmos Genéticos y Procesamiento de Lenguaje Natural, para efectos de este trabajo se analizan los modelos de Recuperación de Información.

Un modelo de Recuperación de información es un método que específica los detalles de la representación del documento, la representación de la consulta [Martínez, 2004] de la necesidad del usuario y lo eficaz de la recuperación en base a la comparación de las representaciones de cada elemento, evaluándose la similitud entre estas.

La estructura que presentan los textos indican que modelos se pueden llegar a utilizar, en este caso se procesa información no estructurada, caracterizados por ser textos libres y sin formato, por lo tanto se destina a utilizar los tres modelos más empleados para recuperar información documental y conocidos como clásicos, el modelo Booleano, el Modelo Probabilístico y el Modelo Vectorial [Baeza, 1999] (ver Figura 2.5). Los tres modelos manejan como unidad de referencia las palabras o términos clave que representan el contenido del texto, y utilizados como términos de indexación, la diferencia entre cada modelo radica en la forma en cómo será representado los términos y en la forma en que puedan ser relacionados.

La comprobación de las consultas y como se ajusta a un documento, también hacen diferenciar los modelos, se distinguen dos modos, los modelos de "ajuste exacto ",en el que el documento debe ser exactamente igual a la consulta, o el modo del "mejor ajuste", en el que el documento cumple de manera parcial con la consulta .En el primer modo es arrojado como solución un conjunto de documentos que más satisfacen la búsqueda, en tanto que en el segundo, se presentan los documentos enlistados ordenadamente de acuerdo a que tanto se ajusta la consulta al escrito. Los modelos probabilísticos y vectoriales utilizan el mejor ajuste, concediendo facilidades de uso y notoriamente más eficaces, mientras el modelo booleano utiliza ajuste exacto, también son muy eficientes, fáciles de

comprender y funcionan muy bien cuando el usuario sabe con certeza la información que necesita y como expresar su consulta.

Probablemente se han dado a conocer más los modelos probabilísticos y el modelo vectorial a comparación del Modelo de Boole.

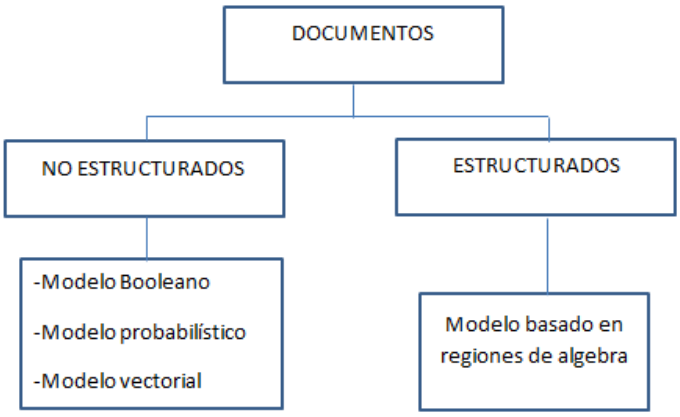



Figura 2.5. Clasificación de los modelos de Recuperación de Información.

### 2.3.1 MODELO BOOLEANO

El modelo de recuperación booleano es muy importante debido a su relativa sencillez para expresar el lenguaje de consulta, la facilidad con la que se puede entender y por lo tanto su puesta en práctica, este modelo considera a un texto como un conjunto de palabras. El uso más simple consiste en la declaración de las palabras clave que pueden estar presentes o no en el documento que se quiere recuperar, tanto los documentos como las consultas están representados por palabras clave, además de que estas a su vez están armadas por medio de combinaciones de los operadores de Boole (AND, OR, NOT) [Manning, Raghavan, Schütze, 2009]. Los resultados ofrecidos por este modelo serán todos aquellos documentos cuya representación satisface la similitud y aparición de las palabras clave indicados en la expresión y la condición de la búsqueda, todos los textos recuperados tiene la misma relevancia y no hay un listado de importancia. Esto ocasiono un problema ocasionando que

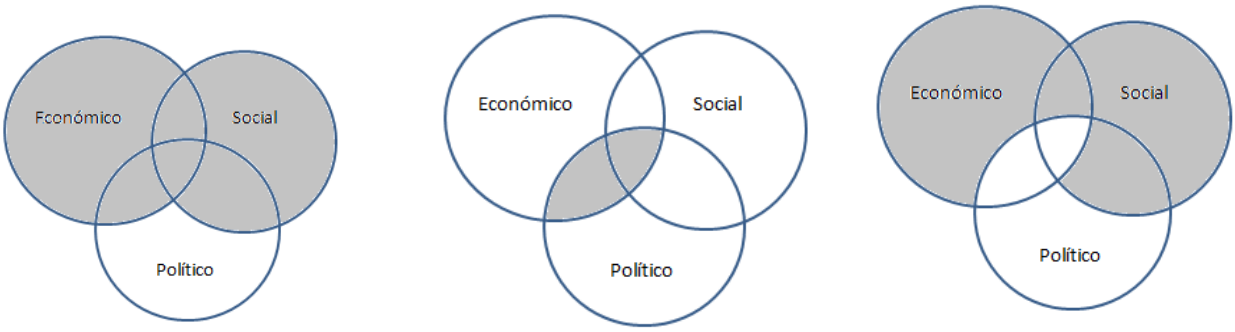
se le hicieran ciertas variaciones para mejorar el proceso de recuperación al modelo original fundamentalmente la capacidad de presentar los documentos de acuerdo a su relevancia y proximidad, por lo tanto es un modelo que opera bajo la teoría de probabilidad de conjuntos, Algebra de Boole y se conoce como modelo booleano extendido [Waller, Kraft, 1979].

Las ventajas que ofrece este modelo es su sencillez y eficiencia, contrastado por sus desventajas como es la dificultad de formular exactamente las necesidades de información empleando algebra de Boole y términos clave.

El funcionamiento del modelo se explica a continuación.

La consulta se formula con una combinación de términos utilizando los operadores AND, OR y NOT, por ejemplo: se tiene una consulta que es elaborada con términos clave "$t_1$ **y (AND)** $t_2$", que solo puede ser satisfecha por un documento $D_1$ y solo un documento $D_1$ que contenga la condición establecida, de manera similar tenemos una consulta formada por "$t_1$ **o (or)** $t_2$" que es satisfecho por un documento $D_1$, y solamente este texto contiene $t_1$ o $t_2$ o ambos. La consulta "$t_1$ **y no (AND NOT)** $t_2$" se satisface si un documento $D_1$ y solo ese documento contiene $t_1$ y no contiene a $t_2$. En la Figura 6 se observa el funcionamiento del modelo booleano y sus operadores.

Consultas más complejas pueden ser elaboradas fuera de los operadores, con más términos y evaluados usando las reglas de Algebra de Boole por ejemplo: $t_1$ OR $t_2$ OR … OR ($t_i$ AND $t_j$) OR ($t_l$ AND $t_m$ AND $t_n$) … OR $t_t$ ; pero la consulta Booleana puede ser falsa o verdadera, ya sea si un documento satisface o no la satisface la necesidad.




Social **o** Económico | Económico **y** Político | Social **o** Económico **y no** (Político **y** Económico)

Figura 2.6. Posibles combinaciones de los operadores Booleanos.

### 2.3.2 MODELO PROBABILÍSTICO

El modelo probabilístico tiene como objeto representar la información desde el punto de vista de las probabilidades y define la recuperación como un proceso de clasificación que esta descrita por dos clases, la clase de los documentos relevantes y los documentos no relevantes [Manning, 2009], conocidos ambos se puede estimar la probabilidad de que aparezca un término *t* en un texto, condición que puede ser usada para determinar la relevancia en los documentos o no. Este método utiliza técnicas para determinar las probabilidades de las palabras, hay diversos modos, pero el más habitual es emplear la frecuencia de los términos clave en cada documento y la frecuencia del total de los documentos que permite distinguir los documentos relevantes de los no relevantes, de esta manera se utilizan los términos que puedan exponer de mejor manera el contenido de un texto y ser usados para la consulta, desechando los términos menos aptos, obteniendo mayor calidad en las respuestas.[Martínez, 2006]

El modelo probabilístico actúa sobre los términos clave que simbolizan la consulta del usuario, asignándoles un peso a cada una de las palabras, originando ciertamente una desventaja la cuál es que el usuario no sabe con seguridad que términos puedan representar lo mejor posible su consulta, ya que las diversas formas harán variar los pesos de cada

palabra. La notación que determina si un documento es relevante es, si la probabilidad de que pertenezca a la clase de los documentos relevantes es mayor que la probabilidad de la clase de los no relevantes, el documento definitivamente es relevante para la consulta (ver Ecuación 1). En la Figura 2.7 se puede observar los subconjuntos de cómo está conformado un modelo probabilístico; REL (Documentos Relevantes), RR (Documentos Recuperados Relevantes), REC (Documentos Recuperados), NN (Documentos No Recuperados No Relevantes).Una recuperación ideal se da cuando el conjunto REL es igual al conjunto REC.

$$P(R|D) > P(NR|D) \quad \text{..... (1)}$$

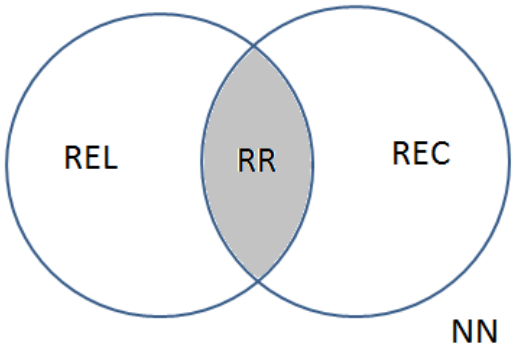



Figura 2.7. Esquema del Modelo Probabilístico.

En este modelo también ofrece esquemas de relevancia en la recuperación de información, por lo cual es determinado a través de la probabilidad de los términos que mejor describen un texto [Rijsbergen,1989], lo cual genera algunos inconvenientes como es calcular la probabilidad de esa palabra para ser un buen o mal descriptor sea para la consulta y en el contenido de la información, este modelo es capaz de calcular la similitud entre el texto y la consulta, arrojando como resultados documentos enlistados de acuerdo a su relevancia

Una característica notoria de estos modelos es el mejoramiento del sistema por medio de retroalimentación, que refina los resultados de la búsqueda con apoyo de la información

arrojados por una primera búsqueda, el sistema pide al usuario evaluar los documentos de acuerdo a su relevancia. Con estos datos el SRI crea perfiles, en donde se establece nuevos valores probabilísticos a las palabras de consulta que el usuario considera importantes, obteniéndose nuevas respuestas de acuerdo a la nueva clasificación, ofreciendo nuevamente listados de los documentos relevantes, solo que ahora mejorado, ya que se ha introducido información de acuerdo a la evaluación del usuario.

### 2.3.3 MODELO VECTORIAL

Hoy en día el método más utilizado para Recuperación de Información es el modelo vectorial o también conocido como espacio vectorial, en este apartado se nombraran las características funcionales de este modelo y su forma para ponderar los términos de los documentos, conocido como *tf-idf* [Martinez,2004] . Este modelo es el soporte teórico para el desarrollo de la metodología propuesta en este trabajo.

El modelo vectorial surge a partir de cubrir los inconvenientes generados por el Modelo Booleano, optimizando y facilitando la formulación de la consulta, así como su representación, por ejemplo: no es necesario hacer uso de los operadores AND, OR y NOT, ofreciendo la ventaja de que en el modelo vectorial, los textos recuperados arrojados por el sistema ya están ordenados de acuerdo a su grado de relevancia con respecto a la consulta elaborada [Salton, 1983].

El modelo vectorial funciona en base a los términos clave [Salton, 1975] contenidas en el documento, palabras cuya característica es describir y representar de manera general el contenido de un texto, así como la consulta de información elaborada por el usuario.

El método consiste en colocar estos términos diferentes entre ellos, en un vector de tamaño igual al número de palabras que hay en cada documento, donde en este vector a cada término se le asigna un valor, indicando el número de frecuencia de aparición en los textos.

La frecuencia de términos considera dos aspectos para la asignación de los valores, el primero expresa que entre más frecuente sea un término en un texto, mayor es su valor, el posible representando el posible contenido del texto, sin embargo, el segundo aspecto

indica que si ese término es muy frecuente en ese texto, pero además aparece frecuentemente en los demás, señala que ese término en particular no podría servir para discriminar los textos por aparecer constantemente en los textos.

El modelo vectorial intenta clasificar los documentos de acuerdo a la similitud entre la consulta y cada documento, ambos representados en forma de vectores sobre un plano vectorial y separado entre sí por un ángulo conocido como Angulo de similitud, la fórmula para calcular la similitud entre la consulta y un documento se llama función Coseno de Similitud. Ver Ecuación 2 y Figura 2.8.

$$Sim_{D_n q} = cos_{D_n q} \frac{D_n \cdot q}{\|D_n\| \|q\|} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (2)$$

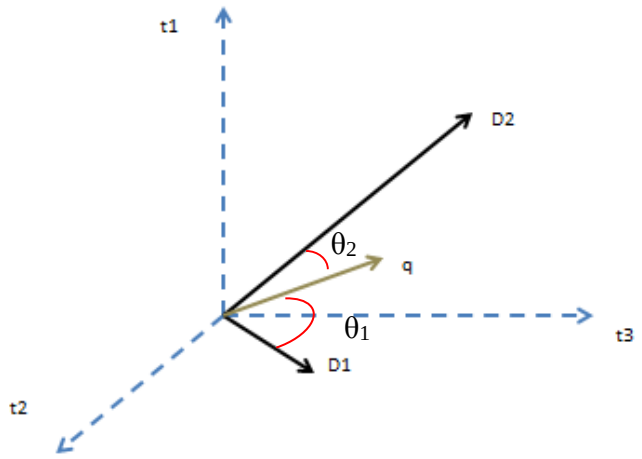



Figura 2.8. Representación del Modelo Vectorial.

A continuación se explica en un panorama general el funcionamiento del Modelo Vectorial.

En la figura anterior están representados vectorialmente los documentos (D1 y D2) así como la consulta (q), se observa que el documento D2 es más relevante para la consulta (q), ya que el ángulo ($\theta_2$) formado entre los vectores es menor a comparación del documento (D1), que notablemente el ángulo ($\theta_1$) es mayor y menos similar a q. Entre menor sea el ángulo entre el vector del documento y el vector consulta, más similares serán ambos vectores, mostrando que el contenido del documento (D2) es similar y posiblemente de mejor respuesta al vector consulta (q). En la ecuación para calcular la similitud, se pueden

obtener resultados de la comparación entre un documento y la consulta , obteniéndose valores oscilados entre 0 a 1, cero significa que los vectores son perpendiculares y por lo tanto el documento no satisface para nada a la solicitud hecha por el usuario, en cambio si el resultado son aproximaciones a uno, los vectores son casi paralelos indicando que el documento puede satisfacer en gran medida a la búsqueda del usuario y por lo tanto es clasificado como un documento relevante.

Cabe señalar que el modelo vectorial está basado en tres principios. [Baeza, 1999]

- Equiparación parcial: el sistema puede ordenar los resultados de una búsqueda, basados en el grado de similitud entre un documento y la consulta.
- Ponderación de los términos en los documentos: Señala la presencian o ausencia de los términos asignándole a cada uno de ellos un valor que refleje su importancia en el documento.
- Ponderación de los términos de consulta: El usuario puede asignar pesos a los términos de la consulta, adjuntándoles valores que reflejen el valor de la consulta.

La ponderación de los términos contenidos en los textos que ha sido más eficiente, utilizada y probada es denominada la medida tf-idf.

#### 2.3.3.1 MEDIDA TF-IDF.

Las unidades por la cual se trabaja un texto son las palabras, específicamente de acuerdo a la frecuencia aparición de estas en los textos. Las palabras identificadas como claves serán asignadas con mayor peso de acuerdo al número de apariciones que tenga esa palabra en un documento que intuitivamente los términos claves son capaces de representar eficientemente el contenido de un documento.

La principal premisa para ponderar palabras indica que entre mayor sea la frecuencia de un término en un documento, este será de mayor importancia para el texto [Martínez 2009].
La frecuencia de una palabra es denominada como *tf* (*term frequency*), es decir, frecuencia de la palabra (término), esta medida significa cuántas veces la palabra ocurre en un

documento. Más específicamente se denomina *tfij*, es decir, cuántas veces la palabra *i* aparece en el documento *j*. Se describe *tf* de la siguiente manera, Ecuación (3).

$$tf\ (t) = \frac{Numero\ de\ veces\ que\ aparce\ t\ en\ un\ documento}{Numero\ total\ de\ terminos\ en\ el\ mismo\ \ documento} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (3)$$

Normalmente la frecuencia de una palabra se combina con otra medida, que se llama *idf* (en inglés, *inverse document frequency*, frecuencia inversa de documentos). La intuición detrás del *idf* se relaciona con lo siguiente: si una palabra se encuentra en todos los documentos de nuestra colección, entonces esta palabra es incapaz de distinguir entre los documentos, y por lo tanto no nos sirve. Y al contrario, si una palabra se encuentra exactamente en un documento en nuestra colección, es una palabra muy útil para cálculos de similitud (o, por ejemplo, recuperación de información, que, como ya mencionamos, es un caso particular de cálculo de similitud). El *idf* se calcula para cada palabra en una colección dada, es decir, depende de la colección, pero no depende de un documento específico en la colección.
La fórmula para el cálculo del idf es la siguiente Ecuación (4):

$$idf_i = log\frac{N}{DF_i} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (4)$$

Donde N es el número total de documentos en la colección de documentos, DFi es el número de documentos donde la palabra *i* se encuentra por lo menos una vez.
En caso general se recomienda combinar *tf* e *idf* de una palabra dada en cada documento: normalmente se multiplican. La medida que combina esas dos características se llama *tf-idf* (se pronuncia "*te-efe-i-de-efe*") y muy a menudo se utiliza como el valor que tienen las palabras como características en el modelo de espacio vectorial para comparar los documentos. A continuación se ilustra la fórmula para calcular tf*id, ver Ecuación (5).

$$tf - idf = tf_{ij} * idf_i \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots..(5)$$

## 2.4 PROCESAMIENTO DEL LENGUAJE NATURAL

En el transcurso de la historia la información ha sido de gran relevancia y ahora de mayor trascendencia con la introducción de herramientas tecnológicas como la computadora y la creación de internet, ampliando el conocimiento y aumentando considerablemente las creaciones textuales como artículos, documentos, libros, etc.[Gelbukh, Sidorov, 2010] Grandes colecciones documentales son expresadas en forma de lenguaje natural almacenadas en formatos digitales, que con el uso de las computadoras han facilitado y ayudado el procesamiento del conocimiento.

Una persona puede comprender y analizar el contenido de un documento arrojado por una computadora porque esta expresado en lenguaje natural o lenguaje humano, recuperando conocimiento de dicho texto, sin embargo para una computadora el texto solo representa archivos, secuencias de caracteres y solo eso, resultando su capacidad limitada al no poder hacer inferencia lógicas sobre el contenido, resumir o contestar preguntas, las computadoras no tienen la capacidad de entender los textos.

Con el argumento anterior podemos definir al procesamiento de lenguaje natural como la disciplina encargada de posibilitar la comunicación entre hombre y computadora por medio del lenguaje natural, producir sistemas informáticos capaces de entender y procesar un texto en base a su sentido y no solo como un archivo binario. [Gelbukh et al. 2010]

En algunos de los sistemas que involucran el procesamiento del lenguaje, generalmente maneja un esquema de tres pasos. El primer paso indica que el texto no se procesa directamente, pasa a través de una transformación o preprocesamiento el cual se obtiene una presentación formal manteniendo las características relevantes del mismo; en el segundo paso, se trabaja con la representación obtenida mediante un programa principal encargado de manipular la representación , buscando subestructuras que sean consideradas necesarias para realizar una tarea específica y por último en el tercer paso, la representación formal se transforma a lenguaje natural. Todo esto es posible gracias a las herramientas computacionales, capacidad de procesamiento, y por lo tanto convirtiendo a la computadora en un procesador de textos.

El procesamiento del Lenguaje Natural así como conocimiento lingüístico nos proporciona elementos que son empleados para la manipulación de textos. Algunas de las aplicaciones que se han desarrollado son las siguientes:

- Ortografía: la detección de un error ortográfico es algo complejo, involucra conocer todas las palabras de un idioma dado, conocer las formas de las palabras, y sobre todo analizar el contexto de la palabra. Por ejemplo: yo fui *haya* a visitar a mi tía. En estos análisis la computadora necesita en cierto grado entender el texto.
- Gramática: Detecta las estructuras en las oraciones que se encuentren incorrectas, aunque las palabras mencionadas están bien escritas. Ejemplo: El profesor de física, se fue. Es difícil localizar estos errores ya que hay gran cantidad de estructuras que están permitidas, es el conjunto de reglas de combinación de palabras y su orden o posición dentro de las oraciones.
- Estilo: Consiste en detectar problemas en el texto, palabras que son bien escritas, bien estructuradas, pero poco entendibles, ambiguas, inconsistentes en el uso de las palabras de diferentes estilos.
- Hechos y coherencia lógica: son herramientas que serán capaces de analizar textos como el que sigue: *ya estoy aquí, pero también me fui.*

### 2.4.1 USO DEL PROCESAMIENTO DEL LENGUAJE NATURAL PARA RECUPERACION DE INFORMACION.

Hoy en día la cantidad de información es demasiada, proveniente de muchas fuentes, ocasionando que la información sea inútil al no poder ser encontrada fácilmente, una de las soluciones propuestas es jerarquizar esa enorme cantidad de documentos otorgándoles cierto grado de relevancia a unos textos más que otros, para lograr esta tarea es necesario medir la relevancia del documento y proporcionarle al usuario primero el documento más relevante, si no le es útil, utilizara el segundo documento relevante y así sucesivamente.

El problema actual en Recuperación de Información consiste en representar de manera adecuada la necesidad real del usuario, además de que no existe un lenguaje formal para formular claramente su necesidad. El uso de lenguaje natural resuelve este problema, la

mayoría de las técnicas empleadas en sistemas de recuperación implican la búsqueda de información por palabras clave, es decir se buscan los textos relacionados a las palabras que el usuario teclea. La representación formal más usada y conocida está dada por el uso de las cadenas de palabras, que son asociadas a la frecuencia o número de ocurrencia de los términos en los textos, mejorando la búsqueda de información con ayuda de herramientas matemáticas y modelos probabilísticos.

El uso de las palabras sin duda mejora el proceso de recuperación, sin embargo sigue siendo limitado, ya que al ser usados los términos siguen siendo representaciones de cadenas de caracteres y no se involucra como tal el contenido del texto, afectando la tarea de cumplir la necesidad del usuario.
Algunas operaciones pueden mejorar la búsqueda de información, interviniendo el contenido de un texto, se expresan a continuación:

- Coincidencia de las formas morfológicas: buscar la palabra *comer* y encontrar *comérselo*. En el idioma español es un problema complejo ya que es extensa la variedad en las formas de las palabras, El procesamiento del lenguaje natural se encarga del modelado de las formas morfológicas. [Gelbukh et al. 2010]

- Coincidencia de los sinónimos: este elemento se ejemplifica de la siguiente manera, si buscamos *perro* es posible que encontremos *can, mascota, cachorro,* etc.

- Considerar la relación entre las palabras de la petición del usuario y los documentos: para realizar esta operación es necesario reconocer la estructura del texto, capaz de mejorar la comparación entre la consulta y un texto.[Gelbukh et al. 2010]

Algunos de los sistemas que son presentados en el mercado, no son capaces de solucionar los problemas mencionados, en mayoría de los casos el nombre de un archivo o un título es insuficiente para representar el contenido de un texto.

## 2.4.2 NIVELES DEL LENGUAJE

En la procesamiento de lenguaje natural trata de establecer un modelo del desarrollo completo del lenguaje, muy difícilmente es posible lograrlo, sin cambio es más fácil desarrollar modelos más pequeños, para ello se ha dividido en partes el lenguaje, tradicionalmente el lenguaje se divide en 6 niveles.

1. Fonética/ Fonología
2. Morfología
3. Sintaxis
4. Semántica
5. Pragmática
6. Discurso

La división de los niveles puede variar, ya que no existen criterios hechos para la separación de cada nivel. A continuación se describirá a grandes rasgos cada nivel del lenguaje.

1) Fonética/Fonología

Parte de la lingüística que indaga sobre las características del sonido, parte fundamental del lenguaje. Este nivel está orientado a desarrollos de métodos fundamentalmente físicos, tales como reconocimiento de voz y síntesis del habla. La fonología dirige su enfoque a la posición del sonido en relación a otros de algún idioma.

2) Morfología

Se encarga de analizar la estructura interna de las palabras y mecanismos de formación, como sufijos, raíces, prefijos, etc. Sistemas de categorías gramaticales como es, genero, numero. Actualmente en desarrollos computacionales en sistemas de análisis.

3) Sintaxis

Analiza las relaciones de las palabras dentro de la frase. La sintaxis computacional está encargada para desarrollar métodos de análisis y síntesis automática. Los sintetizadores automáticos se encargan de generar texto en base a estructuras, en cambio un analizador sintáctico (*parsers*) [Gelbukh et al. 2010], es un algoritmo difícil de crear, especialmente en idiomas como el español, donde el orden de las palabras no es fijo.

4) Semántica

El objetivo de la semántica es ser capaz de "entender" la frase o palabra. Con apoyo de lexicología y lexicografía permiten definir el sentido de las palabras, por ejemplo la palabra radio, es referirse al aparato electrónico que nos permite escuchar música, el elemento químico o la mitad del diámetro de una circunferencia.

5) Pragmática

Trata de la relación entre la oración y el mundo externo, por ejemplo. La acción en la siguiente frase: "*me pasas la sal"*, es muy diferente a decir "*me podrías dar la sal".*

6) Discurso

Esta encargada de analizar la entidad llamada discurso, que es el conjunto de oraciones que están relacionadas entre sí. Comúnmente nos expresamos con un conjunto de palabras que son dependientes, y no de forma aislada.

Para efectos de este trabajo se analiza el nivel morfológico y sintáctico.

## 2.5 NIVEL MORFOLÓGICO

Es la parte de la gramática que se encarga de estudiar la estructura de la oración y sus mecanismos de formación, también indica que parte de la oración es sustantivo o nombre, verbo, pronombre, etc. Las palabras están conformadas por unidades mínimas que contienen significado, conocidos como morfemas [Mervyn, 1992], estas a su vez están clasificadas en morfemas léxicas y morfemas gramaticales.

Los morfemas léxicos, lexemas o raíces, son los elementos que proporcionan significado a la palabra, por ejemplo: hablar .Los morfemas gramaticales, también conocidos como afijos o por extensión, son aquellos que permiten modificar el significado básico de la raíz, ejemplo: hablases.

Los afijos pueden ser clasificados en prefijos, que son antepuestos al lexema, ejemplo en la palabra innecesario; los sufijos, son pospuestos al lexema (hablador), los infijos son los elementos que pertenecen intercalados en el interior de la estructura de una palabra. (humareda).

Los afijos pueden seguir clasificándose de acuerdo de como alteran la raíz, los afijos flexivos, son aquellos que representan conceptos como género y numero, por ejemplo hablադoras, persona, modo, tiempo y aspecto (hablases). Los afijos derivativos, producen un cambio semántico al lexema base y un cambio en la categoría semántica.

Un módulo morfológico reconoce las palabras y las convierte de cadenas de letras en referencias al diccionario y en marcas de tiempo, genero, numero, entre otras.

El análisis morfológico se liga a la morfología de dos niveles [Koskenniemi, 1983], en la que establece considerar a una palabra entre el nivel léxico, que es el enlace de los morfemas que forman una palabra, y el nivel superficial, indica la forma escrita real de la palabra. Apoyado en lo anterior el análisis de una palabra se lleva a cabo a través de un conjunto de reglas, que hacen corresponder secuencias de morfemas y rasgos morfológicos del nivel léxico.

Por ejemplo, forma superficial de la palabra gatos, se convierte en forma léxica: gat + sustantivo + masculino + singular; la palabra anterior es un sustantivo, masculino y expresado en singular.

## 2.6 NIVEL SINTÁCTICO

La sintaxis es la disciplina lingüística que se ocupa de la agrupación de las distintas palabras dentro de unidades superiores y de sus respectivas funciones en el seno de una oración. En tal sentido, las distintas agrupaciones con las que nos podemos encontrar son las siguientes:

El sintagma, es el conjunto de palabras que desempeñan una misma función dentro de la oración. Podemos hablar de la existencia de los siguientes sintagmas:

- Sintagma Verbal, que tiene como núcleo a un verbo.
- Sintagma Adjetival, que tiene como núcleo a un adjetivo calificativo.
- Sintagma Adverbial, que tiene como núcleo a un adverbio.

La Nueva gramática de la lengua española habla de grupos sintácticos y, dentro de ellos, distingue entre los que llama Grupos Nominales (que se forman en torno a un sustantivo o nombre), Grupos Adjetivales (que expanden un adjetivo), Grupos Verbales (que se construyen en torno a un verbo), Grupos Adverbiales (construidos en torno a un adverbio), Grupo Pronominal (que tiene como núcleo un pronombre) y el calificado como "polémico" Grupo Preposicional (en el que habría una preposición y su término). A estos grupos se añade el llamado Grupo Conjuntivo (formado por una conjunción y su término) y el Grupo Interjectivo (formado por algunas interjecciones con su complemento).

-**Las proposiciones**, se llama así a cada una de las partes que integran una oración compuesta y que no tienen autonomía propia ni significado completo por sí mismas.

-**La oración**, se llama oración a aquel enunciado, formado por un sujeto y un predicado, que tiene sentido completo e independencia sintáctica y que está comprendida entre dos pausas mayores.

-**El enunciado o frase**, un conjunto de palabras que tienen sentido completo en una situación determinada pero que carece de predicado verbal.

-**El párrafo**, la unidad que contiene una idea completa dentro de un texto y se marca con un punto y aparte.

-**El texto**, es la unidad de mayor carácter comunicativo y puede abarcar desde un simple saludo hasta una obra literaria completa. Tiene autonomía e independencia total.

### 2.6.1 ANÁLISIS SINTÁCTICO

El representar el contenido de un texto como una cadena, así como reconocer que esta pertenece al lenguaje generado por las reglas de las gramaticales y proponer una presentación para efectuar el análisis es parte del problema que soluciona un análisis sintáctico.

Se identifican dos algoritmos para realizar las tareas de análisis sintácticas, Reconocedores Sintácticos encargados de reconocer la gramática de una cadena de texto y un Analizador sintáctico también conocido como *parser*, su función consiste en representar de una manera apropiada la gramática de una cadena por lo tanto tiene que ver con el reconocimiento de secuencias de las palabras y tiene como objetivo interpretar los atributos de las secuencias.

En el contexto del lenguaje natural, los analizadores sintácticos cobran mayor fuerza donde las cadenas propuestas contienen ambigüedad, propensas a errores y complejas, se proponen analizadores sintácticos que combaten estas condiciones.

Análisis Sintáctico robusto: esta clase de análisis está dirigido a obtener la mayor cantidad de información posible a partir de una cadena de entrada con errores, lagunas gramaticales y en donde no es siempre posible tener de entrada cadenas correctas y completas, este tipos de análisis pueden ser capaces de corregir dichos errores para obtener un análisis completo.

Análisis Sintáctico parcial: este término para referirnos a las técnicas de análisis capaces no solos de obtener, el análisis completo de una entrada, sino también corregirlos.

Análisis sintáctico superior: Algunas veces no es necesario hacer análisis profundos, basta con identificar las estructura de mayor entidad, como son, estructuras nominales, grupos preposicionales.

**FreeLing**

El paquete *FreeLing* es una herramienta que consta de una biblioteca que proporciona servicios de análisis de lenguaje, la versión 3.1 actual presta funciones como identificación del idioma, tokenizacion, división de frase, análisis morfológico, detección y clasificación, reconocimiento de fechas , número, magnitudes físicas, moneda, proporciones, codificación fonética, etiquetado, análisis sintáctico superficial, análisis de dependencia, desambiguación , solución de correferencia.

*FreeLing* está diseñado para ser utilizado como una biblioteca externa de cualquier aplicación que requiera este tipo de servicios.
En el futuro se espera que las versiones posteriores incorporen nuevas características.

Con apoyo de *FreeLing* una aplicación que desee desarrollar por ejemplo, un sistema de traducción automática y que es necesario algún tipo de procesamiento lingüístico del texto original, puede ser utilizados los módulos de *FreeLing* para hacer los análisis requeridos.

*FreeLing* no es una herramienta de análisis de texto orientado al usuario. Es decir, que no está diseñado para ser fácil de usar o que los resultados de salida contenga una presentación adecuada para interactuar con el usuario, o en un formato determinado.

Los resultados de análisis de datos son presentados en estructuras de datos, con la finalidad de que el usuario final pueda acceder a ellos y trabajarlos de la manera que lo necesite, Sin embargo, el paquete de *FreeLing* ofrece una de aplicación bastante completa (analizador) que permite a un usuario final sin conocimientos de programación obtener el análisis de un texto, además *FreeLing* tiene un conjunto de capacidades que puede cubrir muchas opciones procesamiento adecuadas a la necesidad.

A continuación se presenta la Tabla 4 indicando la compatibilidad de los idiomas relacionado con el servicio de análisis que ofrece *FreeLing*.

Idiomas: Asturiano (as), catalán (ca), Inglés (en), francés (fr), Gallego (gl), Italiano (it), portugués (pt), ruso (ru), esloveno (sl), Español (es), y el galés (cy).

Tabla 2.4. Servicios de análisis disponibles para cada idioma.

| | as | ca | cy | en | es | fr | gl | it | pt | ru | sl |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Tokenizacion | X | X | X | X | X | X | X | X | X | X | |
| División de oración | X | X | X | X | X | X | X | X | X | X | |
| Detección de números | | X | | X | X | | X | X | X | X | |
| Detección de fechas | | X | | X | X | | X | | X | X | |
| Diccionario morfológico | X | X | X | X | X | X | X | X | X | X | |
| Fijación de normas | X | X | X | X | X | X | X | X | X | | |
| Detección multipalabra | X | X | X | X | X | X | X | X | X | | |
| Detección básico nombre de entidad | X | X | X | X | X | X | X | X | X | X | |
| B-I-O detección nombre de entidad | | X | | X | X | | X | | X | | |
| Clasificación de nombre de entidad | | X | | X | X | | | | X | | |
| Detección de cantidad | | X | X | X | X | | X | | X | X | |
| etiquetado Pos | X | X | | X | X | X | X | X | X | X | |
| Codificación fonética | | | | X | X | | | | | | |
| WN sentido de anotación | | X | | X | X | X | X | | | X | X |
| UKB sentido desambiguación | | X | | X | X | X | | | | | X |
| Análisis superficial | X | X | | X | X | | X | | X | | |
| Análisis dependencia/completa | X | X | | X | X | | X | | | | |
| Resolución de correferencia | | | | | X | | | | | | |

*FreeLing* incluye diccionarios basados en *WordNet* para algunos de los idiomas cubiertos, así como el código *FreeLing* está licenciado bajo GNU *General Public License* (GPL).

# CAPÍTULO 3. ESTADO DEL ARTE

## 3.1 LUCENE

Apache Lucene no es un programa si no una biblioteca de código abierto gratuito, caracterizado por ser un motor de búsqueda de alto rendimiento para texto, es una tecnología adecuada para agregar capacidades a cualquier aplicación en donde sea necesario búsqueda de texto completo (inglés y español). Es un software que trabaja sobre Java y bajo la licencia de Apache. Es comercialmente amistosa, es multiplataforma y se encuentra disponible. Actualmente está a disposición Apache Lucene en la versión 4.8.0.

Lucene posee características importantes entre estas se encuentran ofrecer búsquedas por relevancia, varias formas de consulta, consultas por frases, consultas por proximidad, búsquedas por campo (autor, titulo, contenido), etc.

Lucene utiliza métodos de recuperación de información, estos son el Modelo Booleano, el Modelo Vectorial con uso de tf-idf y diferentes formas de similitud. Los documentos que apruebe el modelo booleano serán ordenados por medio del modelo vectorial.


Figura 3.1. Logotipo de Apache Lucen.Cortesia de Lucene

### 3.1.1 REPRESENTACIÓN

La representación de las consultas y los documentos se hace a través de vectores ponderados plasmados un plano vectorial, donde cada uno de ellos están compuestos por términos y valores tf-idf distintos. En el modelo vectorial no es necesario usar tf-idf, sin embargo, los valores ft-idf definitivamente mejoran los resultados, por lo tanto Lucene usa esta técnica. Para calcular la puntuación de similitud se tiene un término t, un documento o

consulta x, tf(t,x) que varía al número de ocurrencia del termino t en x e idf (t) que de manera similar varia con la inversa del número de documentos que contienen t.

Por medio de la operación coseno similitud el modelo vectorial otorga una puntuación del documento con respecto a la consulta, evaluando la similitud entre estas.

Lucene refina los puntajes del modelo vectorial tanto para la calidad de búsqueda y usabilidad, usando el siguiente procedimiento:

- Se normaliza el vector documento en donde se elimina información relativa a la longitud del documento en función del número de términos que contiene, que puede ser correcto en el caso de que el contenido o los términos presentes en los textos sean duplicados, sin embargo, en el caso donde la información no sea redundante sería un problema, para evitarlo se crea un factor de normalización diferente, que normaliza un vector del mismo tamaño que el vector unitario. (doc-len-norm(d)).
- En el proceso de indexación los usuarios pueden especificar que determinados documentos son importantes que otros, mediante la asignación de valor para acentuar esos documentos. Para ello se multiplica este valor de "importancia" por la puntuación de similitud de cada documento. (doc-boost(d)).
- Durante el tiempo de recuperación o búsqueda los usuarios pueden especificar la importancia de algunos términos de la solicitud que se toma en cuenta mediante el query-boost (q). Resultado de multiplicar la "importancia del término" por el valor de la consulta.
- Con este modelo es posible que los documentos recuperados no contengan todos los términos de la consulta. En lucene los usuarios también tiene la capacidad documentos con mayor términos, es posible mediante un factor de coordinación que entre mayor sea este factor significa que los documentos tanto las consultas compartirán mayor cantidad de términos comunes.

Se presenta una ecuación conceptual (Ecuación 6) utilizada en Lucene para la recuperación de información. Es una simplificación de la formula realmente utilizada.

$$score\,(q,d) = coord - factor(q,d) \cdot \quad query - boost(q) \cdot \frac{V(q)\cdot V(d)}{|V(q)|} \cdot \quad doc - len - norm(d) \quad \cdot \quad doc - boost(d) \qquad \ldots. \quad (6)$$

|

### 3.1.2 RECUPERACIÓN

A continuación se describe como Lucene implementa la formula anterior y posteriormente deriva en la Función de Puntaje Práctico de Lucene. Algunos de los componentes de la función son calculados y agregados previamente.

-*query-boost* para la consulta (actualmente para cada término de la consulta) es conocida cuando inicia la búsqueda.

- El vector normalizado de consulta | V (q) |, es calculado cuando la búsqueda se inicia, ya que es independiente del documento, que ayuda a mantener la puntuación del vector unitario y no haya perdidas de información, además de ayudar la comparabilidad hasta cierto punto de dos o más consultas.

- la longitud del documento normalizado (*doc-len-norm(*d)) y el documento m mayor importancia (*doc-boost*(d)), son conocidos en el tiempo de indexación. Se calculan con antelación y su multiplicación es guardada como un único valor. En la siguiente expresión se presenta *norm*(t,d) significando *norm* ( campo(t) en documento d) donde el campo t hace referencia al termino t. La Función de Puntuación Práctica de Lucene con los elementos anteriores se describe en la siguiente figura.

$$score(q,d) = coord(q,d) \cdot queryNorm(q) \cdot \sum_{t\ in\ q}( tf(t\ in\ d) \cdot idf(t)2 \cdot t.getBoost() \cdot norm(t,d) ) \quad \text{….. ……..} (7)$$

Donde:

- tf(t in d) se relaciona a la frecuencia del termino y se define como el número de veces que aparece un término t en un determinado documento d. Los documentos en lo que hay más apariciones de un término reciben puntuación más alta. el cálculo predeterminado en *Lucene* de tf(t in d) (Ecuación 8) es por medio de la clase *DefaultSimilarity*, expresada en la siguiente manera.

$$tf(t\ in\ d) = \sqrt{frecuencia} \quad \text{……} (8)$$

- *idf* (t) representa la frecuencia inversa del documento. Este valor se correlaciona a la inversa de *docFreq* (el número de documentos en los que aparece el término t). Esto significa que más apariciones dan mayor contribución a la puntuación total. idf (t) para t aparece tanto en la consulta y en el documento, por lo que se eleva al cuadrado en la ecuación. El cálculo predeterminado para idf (t) en *DefaultSimilarity* es:

$$idf(t) = 1 + log\left(\frac{num\ Docs}{doc\ Frec+1}\right) \quad \text{..... (9)}$$

- *coord* (q, d) es un factor de puntuación basada en cuántos de los términos de consulta se encuentran en el documento especificado. Comúnmente, un documento que contiene más de los términos de la consulta recibirá una puntuación más alta que otro documento con un menor número de términos de la consulta.
- *queryNorm* (q) es un factor de normalización utilizado para hacer las puntuaciones entre consultas comparables. Se expresa en la siguiente ecuación (Ecuación 10).

$$queryNorm(q) = queryNorm(sumOfSquaredWeights) = \frac{1}{\sqrt{sumOfSquaredWeights}} \text{..... (10)}$$

El parámetro *sumOfSquaredWeights* (de los términos de consulta) se calcula por el peso de consulta. Por ejemplo, se calcula en la Consulta booleana este valor como:

$$sumOfSquaredWeights = q.getBoost()^2 \cdot \sum_{t\ in\ q}(idf(t) \cdot t.getBoost())^2 \text{.....(11)}$$

- *t.getBoost* () mejora el tiempo de búsqueda de término t en la consulta tal como se especifica en el texto de la consulta (ver la sintaxis de consulta), o según lo establecido esta es procesada por la aplicación llama a *setBoost ()*.

## 3.2. WUMPUS SEARCH ENGINE

*Wumpus* es un paquete de software escrito en C++ y sistema de recuperación de información desarrollada en la universidad de Waterloo. El objetivo del método es el estudio de los problemas que surgen del contexto de indexación, colecciones de textos dinámicos o entornos multiusuario. El escenario particular en el que se estudia es en la búsqueda de sistemas de archivos.

*Wumpus* tiene dos posibilidades de uso:

- Como un sistema de recuperación común con soporte multi- usuario.
- Un sistema de indexación de archivos que guarda automáticamente las pistas de todos los cambios en el sistema de archivo.

*Wumpus* es muy escalable y se ha utilizado en colecciones de texto que consisten de muchos cientos de gigabytes de texto y que contiene docenas de millones de documentos. Wumpus está libremente disponible bajo los términos de la Licencia Pública General de GNU.


Figura 3.2. Logotipo de Wumpus. Cortesia de Wumpus

### 3.2.1 SISTEMA DE RECUPERACION DE INFORMACION WUMPUS

En muchos de los casos una persona hace búsquedas más complejas incapaces de ser expresadas por solo un término o una frase ,a diferencia de la mayoría de los otros motores de búsqueda, se ha incorporado el concepto de GLC (listas de concordancia generalizada, en español), lenguaje de consulta que es compatible con una variedad de operadores para expresar limitantes estructurales en la consulta , es decir, puede ser usado en muchos casos

para remplazar la estructura de la consulta por muy compleja que sea por palabras sencillas.

Por ejemplo, estos operadores pueden ser utilizados con el fin de buscar los pasajes, donde *"wumpus"* y "buscar" aparecen dentro de 5 palabras de uno al otro:

*("Wumpus"* ^ *"search"*) < [5]

La estructura más importante en la mayoría de las aplicaciones de recuperación de información son documentos. Con el fin de acceder a los documentos en *Wumpus*, tienen que estar marcados de alguna manera dentro de los archivos de origen Si, por ejemplo, el archivo de origen es un documento XML en el que cada documento se inicia con una etiqueta <doc> y termina con un </ doc> etiqueta, entonces usted puede utilizar GCL para crear el conjunto de todos los documentos:

"<doc>" .. "</ Doc>"

Esto se puede extender fácilmente para que Wumpus devuelva una lista de todos los documentos que contengan una determinada palabra:

("<doc>" .. "</ Doc>")> "*wumpus*"

O todos los documentos de una determinada longitud, dicen 1000:

(("<doc>" .. "</ Doc>")> [1000]) <[1000])

Devuelve todos los documentos que contenga la palabra “hombre” y “mujer”:

("<doc>" .. "</ Doc>")> ("hombre" ^ "mujer")

A continuación se muestran los operadores que utiliza *Wumpus* en la siguiente Tabla 3.1

Tabla 3.1. Operadores en *Wumpus*

| Operador | Expresión |
|---|---|
| (A ^ B) | Devuelve la extensión de la consulta que aparece en A y en B |
| (A + B) | Devuelve la extensión que coincida tanto en cualquiera A o en B o en ambos |
| (A .. B) | Devuelven la extensión que comiencen en un punto en A y terminen con un punto en B |
| (A> B) | Devuelve la extensión de un punto que coincida con A y contiene al menos un punto que coincida con B |
| (A /> B) | Devuelve todas las extensión que pertenecen en A y no tiene ningún punto que coincide con B |
| (A <B) | Devuelve los puntos que coincidan con *A* y que están contenidos en una medida que coincide con B |
| (A / <B) | Devuelve las extensiones que coincidan con *A* y que no están contenidos en una medida que coincide con B. |
| [N] | Devuelve todas las extensiones del índice de longitud n |

Podemos observar que el método utilizado en *Wumpus* por sus características de recuperación pertenece a un modelo Probabilístico.

*Wumpus* soporta varios tipos de consultas de relevancia. La sintaxis general de una consulta relevancia es:

`rank[`*`method`*`][count=`*`N`*`][id=`*`ID`*`][`*`additional_parameters`*`] what by scorer`$_1$ `... scorer`$_m$

Los componentes individuales de la consulta son:

*Method*: Esta puede ser una de OKAPI, QAP, o QAP2. BM25 es un alias para OKAPI.

*Count:* El número de extensiones del índice que debe ser devuelto por el algoritmo de puntuación. Si no se especifica, N = 20 se asume.

Id: Es útil para el procesamiento por lotes. Si se le da un identificador de la consulta, aparecerá en el inicio de cada línea de resultados, por lo que es más fácil de decir que

pertenece línea de resultados a los que consulta cuando varias consultas se presentan en forma consecutiva.

*What*: Se trata de una expresión GCL arbitraria que define la lista de extensiones de índice que van a ser anotada y cuyos resultados son de actualidad para ser informado al usuario. Este es usualmente `<doc>..</doc>`.Si no se especifica la lista de extensiones que se anotó, el comportamiento depende de la función real de puntuación. BM25 marcará archivos enteros, mientras que el PGC se acaba de informar de los pasajes pertinentes sin ninguna información sobre la medida en que contiene un pasaje.

*scorer*$_i$: De nuevo, esto puede ser una expresión GCL arbitraria, pero es por lo general un solo término (o tal vez una frase) para consultas de relevancia.

Existen accesos directos con los significados obvios: @ okapi, @ BM25, @ QAP

Dependiendo de la función de puntuación utilizada, es posible ajustar los parámetros internos de la función de manera que cumplan los requisitos específicos de la aplicación. Para BM25, por ejemplo, los parámetros b y K1 se pueden modificar como se muestra a continuación.

## Ejemplos de consultas elaboradas en *Wumpus.*

**@rank[bm25][id=42][count=5] "<doc>".."</doc>" by "information", "retrieval"**
42 16.733690 331444 331906
42 16.414286 33999512 34000002
42 16.310987 1788586 1789040
42 16.283293 34968932 34969266
42 16.223730 331907 332390
@0-Ok. (16 ms)

**@okapi[k1=1.5][b=0.5][count=8] "<doc>".."</doc>" by "information", "retrieval"**
0 18.165737 331444 331906
0 17.831360 33999512 34000002
0 17.577662 1788586 1789040
0 17.548101 331907 332390
0 17.147125 34968932 34969266
0 17.138239 27589177 27589551
0 16.899017 1445358 1445827
0 15.956622 1346297 1346997
@0-Ok. (18 ms)

**@qap[count=5][id=32] "<doc>".."</doc>" by "information", "retrieval"**
32 19.661619 1788586 1789040 1789004 1789005
32 19.661619 1445358 1445827 1445790 1445791
32 19.661619 27589177 27589551 27589246 27589247
32 19.661619 331444 331906 331869 331870
32 19.661619 331907 332390 332332 332333
@0-Ok. (15 ms)

### 3.2.2 OKAPI BM 25

El sistema de puntaje que maneja *Wumpus* es el método Okapi Bm25, el método se describirá a continuación.

La componente clave debido al éxito de Bm25 es el termino de frecuencia tf, La escala y la forma de este componente TF normalización es controlado por un parámetro K1, que generalmente se establece en un constante-término independiente. La fórmula Bm25 presenta la puntuación de un documento D con respecto a la consulta Q como sigue:

$$\sum_{q\in Q\cap D} \frac{(K_3+1)\cdot c(q,Q)}{k_3+c(q,Q)} \cdot dtf(q,D) \cdot \log \frac{N+1}{df(q)+0.5} \qquad \text{.....(12)}$$

Dónde:

$c(q,Q)$ es la cuenta del término q en la consulta Q, N es el número total de documentos, df (q) es la frecuencia de q en el documento y $k_3$ es un parámetro.

La fórmula de normalización de tf (*frecuency term*).

$$dtf(q,D) = \frac{(K_1+1)\cdot c(q,D)}{k_1 1\left(-b+b\left(\frac{|D|}{avdl}\right)\right)+c(q,D)} = \frac{(K_1+1)\cdot c'(q,D)}{K_1+c'(q,D)} \qquad \text{.....(13)}$$

Dónde:

$$c'(q,D) = \frac{c(q,D)}{1-b+b\left(\frac{|D|}{avdl}\right)} \qquad \text{.....(14)}$$

# CAPITULO 4. METODOLOGÍA

## 4.1 LA METODOLOGÍA PROPUESTA

En el presente capítulo se presenta la metodología desarrollada que opera bajo una petición de entrada, entra al sistema de recuperación y posteriormente arroja los documentos que más cumplen con la petición del usuario.

El modelo de recuperación está conformado por siete etapas: pre procesamiento de texto análisis del texto mediante *FreeLing*, selección de términos de interés (sustantivos), cálculo de la medida tf-idf, procesamiento de la petición del usuario y la etapa de recuperación.
Cada documento es representado por un vector, donde cada componente toma el valor de la medida tf_idf de cada lema que conforma el texto. Los resultados que arroja el método depende de que tanto se parecen los vectores del documento comparados con el vector de la petición, para esto se usa la función similitud coseno es así como el método muestra los documentos con mayor relevancia para la petición de usuario. En el capítulo siguiente se tratara la manera de evaluar el sistema y se obtendrán las conclusiones correspondientes.

## 4.2 DESCRIPCIÓN DEL MODELO PROPUESTO

En el siguiente diagrama a bloques se muestra el modelo propuesto.

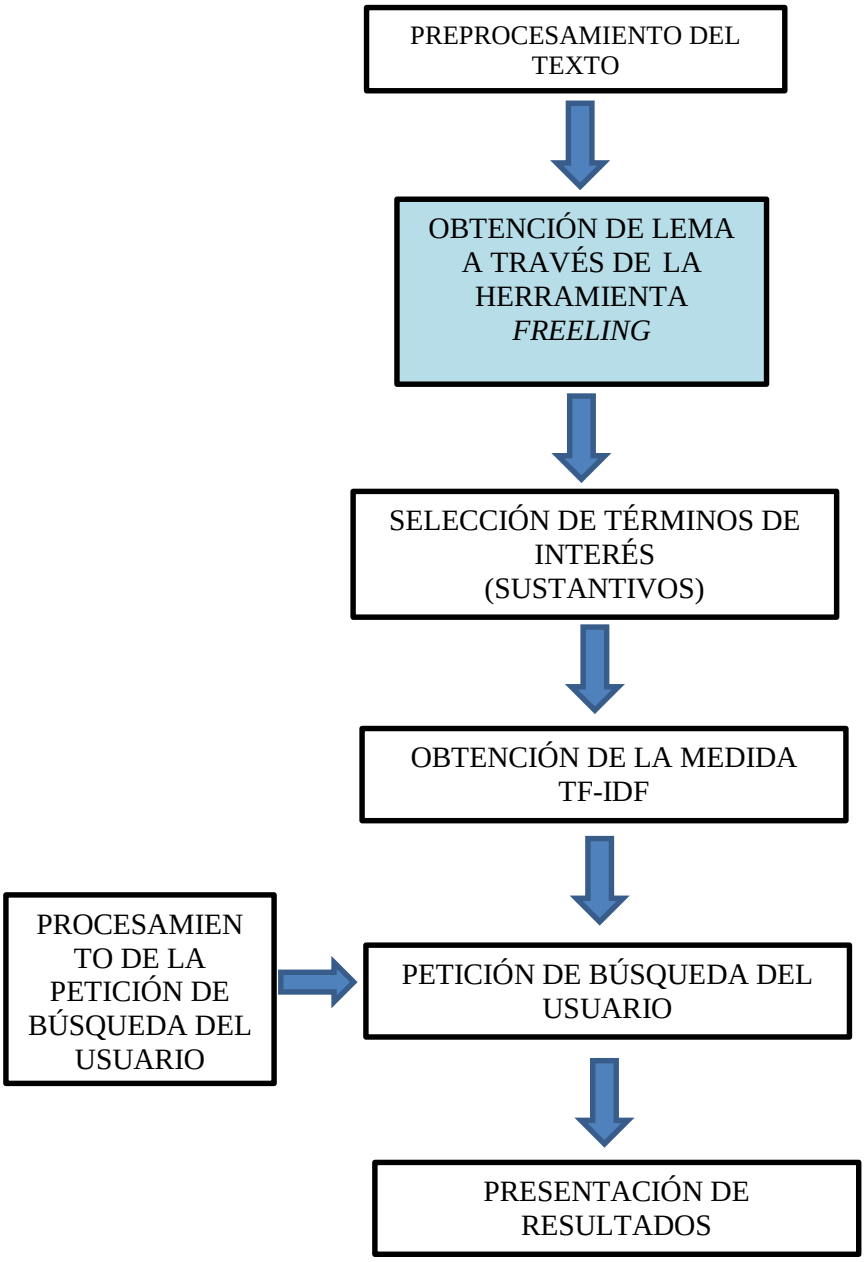



Figura 4.1. Diagrama a bloques del sistema de recuperación de información

El diagrama anterior muestra las etapas que sigue la metodología de recuperación de información, desde el pre procesamiento del texto hasta la presentación de resultados. Cada una de las etapas realiza una función diferente que requiere de la anterior para poder continuar con el proceso. En las siguientes secciones se explican cada una de las etapas del diagrama anterior.

Un punto importante en la en la figura es el bloque sombreado; esto es porque la herramienta *Freeling* no es una herramienta creada en este trabajo si no es un herramienta que es requerida por el proceso de recuperación, posteriormente se explica cómo es que funciona, y porque es utilizada.

## 4.3 ETAPA DE PREPROCESAMIENTO

En esta etapa se obtiene un *corpus* al que llamaremos texto plano, es decir no contiene imágenes, tablas, expresiones matemáticas y citas bibliográficas. A continuación se muestra un ejemplo de un texto plano, para este trabajo los textos fueron obtenidos de página del Periódico "El Universal".

Asteroide del tamaño de tres campos de futbol pasa "cerca" de la Tierra

Washington. Un gran asteroide -conocido como 2000 EM26- pasó cerca de la Tierra esta noche (tiempo de México) sin que exista peligro alguno para nuestro planeta, aseguran científicos.

La roca, del tamaño de tres campos de futbol, se desplaza a una velocidad de unos 12 kilómetros por segundo, pero no es motivo de alarma ya que lo más cerca que estuvo de la superficie terrestre es a una distancia equivalente a 8.8 veces la distancia de la Tierra y la Luna, explican.

Hace poco más de un año, el 15 de febrero de 2013, un meteorito impactó en la región de Cheliábinsk, Rusia, donde provocó daños en varios edificios y centenares de personas heridas, sobre todo a causa de los cristales rotos.

En ese entonces, investigaciones realizadas al objeto cósmico determinaron que el meteorito había sufrido un intensivo proceso de descomposición antes de ser sometido a temperaturas extremadamente altas al entrar en la atmósfera de la Tierra.

También se conoció que pertenecía a un tipo conocido como una condrita LL5 y es bastante común entre aquellos que se han derretido antes de caer en la Tierra.
Para el desarrollo del sistema se requiere de este tipo de texto ya que las herramientas posteriores solo trabajan bajo este tipo de corpus.

## 4.4 ETAPA DE ANÁLISIS SINTÁCTICO

Después de obtener el texto plano se prosigue con un análisis morfológico del texto esto para obtener el lema de cada palabra que conforma dicho texto.
Para la realización de este análisis se requirió de la herramienta *FreeLing* que es una herramienta de procesamiento del lenguaje disponible para el idioma español. El proyecto *FreeLing* fue creado y actualmente está liderado por Lluís Padró como un medio para poner a disposición de la comunidad los resultados de la investigación llevada a cabo en el grupo de investigación del lenguaje natural UPC (Universidad Politécnica de Cataluña).

A continuación se muestra un ejemplo de cómo funciona *Freeling* en su modo de análisis sintáctico.

Tabla 4.1. Funcionamiento de *FreeLing*

| **Palabra original** | El | gato | come | pescado | y | bebe | agua |
|---|---|---|---|---|---|---|---|
| **Lema** | el | gato | comer | pescado | y | beber | agua |
| **Clave** | DA0MS | NCMS0 | VMIP3S0 | NCMS000 | CC | VMIP3S0 | NCCS000 |

Como se puede observar en la tabla esta herramienta nos devuelve el lema de cada palabra más una clave que va a ser utilizada posteriormente en la siguiente etapa para filtrar los términos de interés (sustantivos).
*Freeling* que se ejecuta desde el símbolo del sistema, a continuación se muestra cómo es que se realiza dicha operación.

En las siguientes imágenes se muestra el proceso para utilizar la herramienta *FreeLing.*

1. Procesando el archivo Azteca1

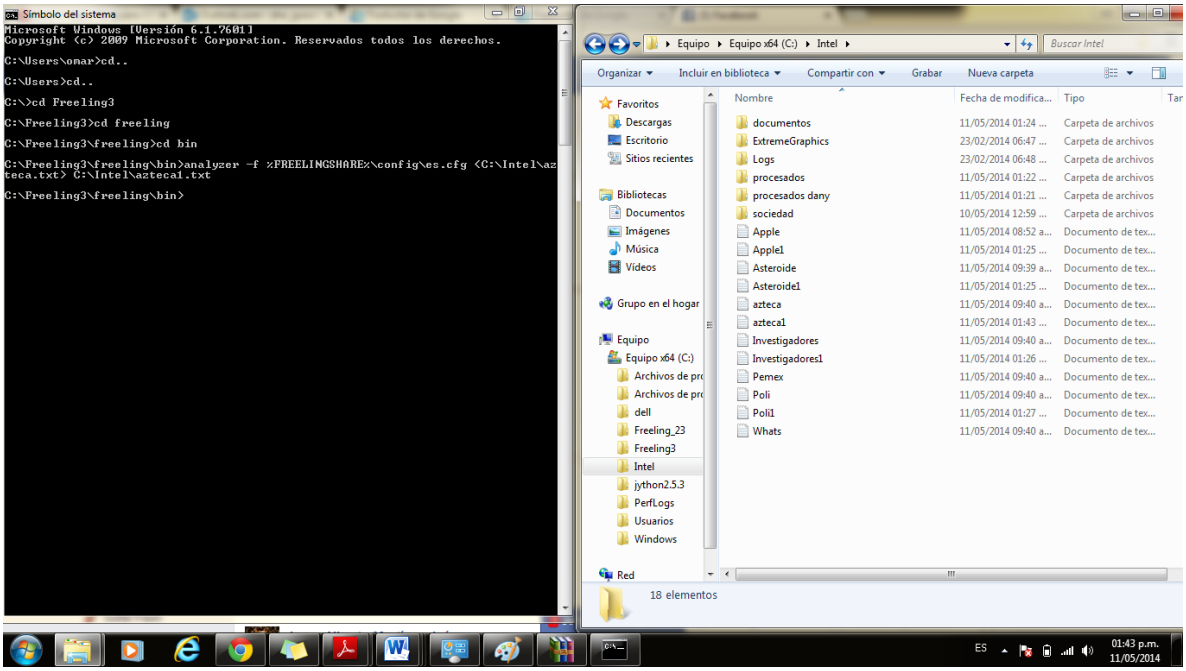



Figura 4.2. Procesando documento en *Freeling.*

En la figura anterior se muestra de lado izquierdo la ventana de MS-DOS donde se ingresan los comandos para poder utilizar la herramienta *FreeLing* y procese los documentos para poder seguir con la siguiente etapa, de lado derecho se muestran los documentos que serán procesados por dicha herramienta, estos documentos están guardado con extensión txt previamente procesados para que solo contengan texto plano.

2. Proceso terminado.

En la siguiente figura se muestra nuevamente una ventana en MSDOS donde muestra que el documento ha sido analizado, del lado derecho muestra los documentos que ya fueron procesados y los que se crearon después del procesamiento con la herramienta *FreeLing.* Nuevamente crea un documento con extensión txt, el cual contiene todas las palabras del texto, el lema de cada palabra y la clave que diferencia entre sustantivos, verbos, preposiciones, artículos, adjetivos, puntuación, numerales, adverbios, conjunciones y determinantes.

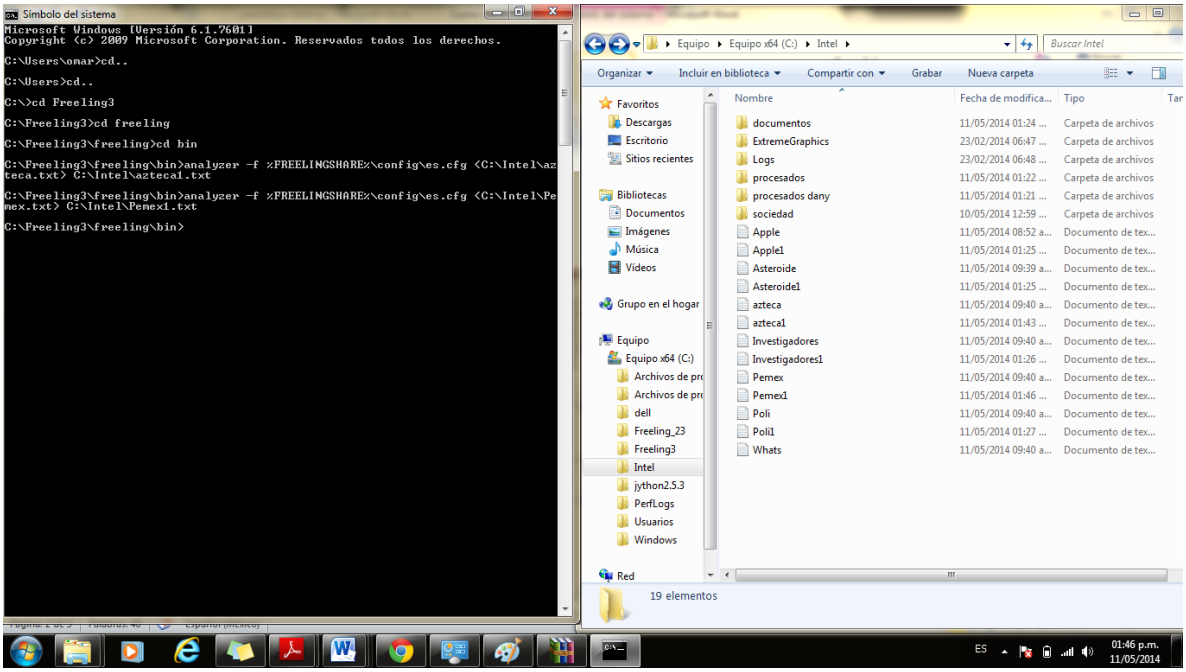



Figura 4.3. Procesamiento terminado en *FreeLing*

A continuación se muestra la salida de *freeling* para el documento Azteca1 (solo es un fragmento del texto).

Tv_Azteca tv_azteca NP00000 1
denunciará denunciar VMIF3S0 1
penalmente penalmente RG 1
a a SPS00 0.996023
presidente presidente NCMS000 1
de de SPS00 0.999984
administración administración NCFS000 1
de de SPS00 0.999984
Dish_México dish_méxico NP00000 1

## 4.5 ETAPA DE SELECCIÓN DE TÉRMINOS

Los términos que pueden determinar el contenido de un archivo son los sustantivos, es por eso que para el modelo propuesto son los términos de interés. Después de procesar el archivo en *FreeLing* se hace un filtrado de todos los sustantivos que contiene el texto, a partir de este punto se realiza la programación en Python, el método usado para esta etapa es el siguiente:

*Freeling* devuelve la siguiente estructura:

Tabla 4.2. Etiquetas de referencia de *FreeLing*

| ETIQUETAS | | | |
|---|---|---|---|
| **Posición** | **Atributo** | **Valor** | **Código** |
| *Columna 1* | *Columna 2* | *Columna 3* | *Columna 4* |

En la columna 1 encontramos un número que hace referencia al orden y posición en que aparecen los atributos. La columna 2 hace referencia a los atributos, el número de los cuales varía dependiendo de la categoría. En la columna 3 encontramos los valores que puede tomar cada atributo y, finalmente, la columna 4 representa los códigos que se han establecido para su representación. Las etiquetas en sí sólo son los códigos (columna 4) y se sabe a qué atributo pertenecen por la posición (columna 1) en la que se encuentran.

*Freeling* pone etiquetas para poder identificar sustantivos, verbos, preposiciones, pronombres, fecha, hora, conjunción, numerales, adverbios, puntuación, determinantes e interjecciones.

Los sustantivos tienen como lema la forma singular, tanto si es de género femenino como masculino o neutro. Para los nombres invariables, es decir, aquellos que tanto para el singular como para el plural presentan la misma forma, el lema y la forma coincidirán.

Los nombres propios tendrán la etiqueta NP00000 si no se usa clasificación semántica, o bien, el valor correspondiente en los dígitos 5-6.

El atributo grado se especificará para aquellos nombres que tengan sufijación apreciativa (aumentativos, despectivos, etc.). Para el resto de nombres este valor será 0.[Padro,1998]

Ahora entonces se prosigue con la discriminación de términos de interés en este caso los sustantivos, esto se hace por medio de programación en Python, en los anexos se ve la codificación que se utilizó en esta etapa.

## 4.6 APLICACIÓN DE LA MEDIDA TF-IDF

Después de haber obtenido los términos de interés, se prosigue a obtener la medida tf-idf, como ya se mencionó en el capítulo dos, dicha medida se obtiene de la siguiente manera:
El termino tf (*term frequency)* que hace referencia a la frecuencia de una palabra dentro de un documento se obtiene de obtener el número de veces que aparece una palabra en un texto dividido entre el número total de elementos del documento, en el anexo se muestra la codificación de esta etapa, cabe mencionar que esta medida se debe obtener para cada sustantivo de cada texto.
El siguiente paso consiste en obtener el idf (*inverse document frequency)* frecuencia inversa de documentos, a cada tf se le asocia un idf debido a que si una palabra aparece en todos los documentos de la colección, no serviría para reconocer un documento, por lo tanto esta palabra no se puede utilizar para la función similitud. El idf se obtiene sacando el logaritmo del cociente del número total de documentos de la colección entre el número de documentos en el que aparece cada palabra del texto y se debe obtener para cada palabra, en este caso para cada sustantivo de cada documento.
La expresión para el cálculo del idf es la siguiente:

$$idf_i = \log \frac{N}{DF_i} \quad \text{.....( 15 )}$$

Dónde:

N = Número total de documentos de la colección.

$DF_i$ = Número de documentos donde la palabra *i* se encuentra por lo menos una vez.

Por último se realiza la multiplicación de tf * idf y de esta manera se obtiene la medida tf-idf para cada palabra de cada texto en la colección. Sin embrago esta medida de manera individual no sirve de mucho, requiere de otras herramientas para poder ser utilizada, en el caso particular de la recuperación de información se aplica en el modelo de espacio vectorial para comparar documentos, en este trabajo se utiliza para comparar el documento de la colección con la petición del usuario.

La siguiente tabla muestra un ejemplo de cómo obtener la medida tf_idf.

Tabla 4.3. Procesamiento en FreeLing

| Parámetros para el calculo de la medida tf_idf | | | | | | | |
|---|---|---|---|---|---|---|---|
| Palabra | Número de veces que aparece en el documento | Total de elementos del documento | Medida tf | Número de documentos en los que aparece | Numero total de documentos | Medida idf | Medida tf-idf |
| Pensión | 11 | 81 | 0.1358 | 1 | 13 | 1.1139 | 0.1512 |
| Café | 5 | 85 | 0.058 | 1 | 13 | 1.1139 | 0.0655 |
| Automóvil | 4 | 89 | 0.0449 | 1 | 13 | 1.1139 | 0.05 |
| Freud | 18 | 331 | 0.054 | 3 | 13 | 0.06368 | 0.03463 |
| Telmex | 7 | 46 | 0.1521 | 1 | 13 | 1.1139 | 0.1695 |

## 4.7 PROCESAMIENTO DE LA PETICIÓN DEL USUARIO

En esta etapa se procesa la petición del usuario de la misma manera que se procesaron los textos; primero se obtienen los lemas de la petición por medio de *FreeLing*, posteriormente se hace un filtrado para quedarse con los términos de interés (sustantivos) y se le asigna una medida tf-idf con valor de uno.

El siguiente ejemplo muestra el funcionamiento de esta etapa.

“Algún documento que hable sobre una tesis”

Después de procesar con *FreeLing* la oración anterior se obtiene lo siguiente:

Tabla 4.4. Procesamiento de petición en *FreeLing*

| Palabra | Lema | Clave | Código |
|---|---|---|---|
| Algún | Algún | RG | 1 |
| documento | documento | NCMS000 | 0.973494 |
| que | que | CS | 0.437483 |
| hable | hablar | VMSP3S0 | 0.618894 |
| sobre | sobre | SPS00 | 0.997091 |
| una | uno | DI0FS0 | 0.951575 |
| tesis | tesis | NCFN000 | 1 |

Ahora que se tiene la petición procesada por *FreeLing* se clasifican los términos de interés (sustantivos), para la petición anterior las palabras que se requieren son:

Tabla 4.5. Términos de interés de la petición

| **Lema** | **Clave** |
|---|---|
| Documento | NCMS000 |
| tesis | NCFN000 |

Las palabras anteriores se utilizan para formar el vector petición.

## 4.8 OBTENCIÓN DE LA FUNCION SIMILITUD COSENO

En este trabajo se ocupa el método de espacio vectorial, por lo que es necesario comparar dos vectores de dos documentos, para el modelo propuesto se compara el vector a que es el vector formado por la medida tf-idf de cada elemento de cada documento (se genera un vector por cada documento de la colección) con el vector b que se forma de la medida tf-idf de cada elemento de cada petición (por cada petición se genera un vector). Cada vector se forma de la siguiente manera:

Tabla 4.6. Vector para un texto

| **LEMA** | Hola | Amigo | Cumpleaños | Feliz |
|---|---|---|---|---|
| **Tf_idf** | 0.3 | 0.7 | 0.2 | 0.6 |

Mediante la expresión de la función similitud se obtiene el coseno del ángulo entre ambos vectores, en el anexo se puede ver la codificación de dicha función.
La función de similitud se obtiene con la siguiente expresión.

$$\cos\theta = \frac{a.b}{|a| * |b|} \dots\dots(16)$$

## 4.9 PRESENTACIÓN DE LOS DOCUMENTOS RECUPERADOS

Como última se presenta la función similitud coseno de cada documento, siendo el documento de mayor valor el que contiene la búsqueda del usuario, en estos resultados se pueden dar los siguientes casos:

1. Solo un documento tiene un valor diferente *Cos* $\theta$ diferente de cero. Significa que en un solo documento se encuentra las palabras de la petición del usuario.

2. Dos o más documentos aparecen con un valor *Cos* $\theta$ diferente de cero. Significa que varios documentos contienen las palabras de la petición del usuario siendo el de mayor valor el documento más relevante para la petición del usuario.

3. Todos los documentos aparecen con valor *Cos* $\theta = 0$. Significa que ningún documento contiene alguna de las palabras de la petición del usuario.

4. Dos o más documentos aparecen con *Cos* $\theta$ diferente de cero pero iguales, en este caso se muestran ambos documentos y es indistinto el orden en que se muestra.

# CAPITULO 5. PRUEBAS Y RESULTADOS.

## 5.1 TEXTO DE PRUEBA.

El corpus de prueba para esta metodología son textos obtenidos de noticias del periódico "El Universal" tratando temas de tecnología, ciencia y sociedad; además se ocuparon tres artículos sobre Sigmund Freud. Se ocuparon 22 diferentes peticiones de usuario.

Para la petición:

"Algún documento que hable sobre una tesis."

El método recupero los siguientes documentos:

Tabla 5.1. Resultados sobre la petición.

| **Resultados de la petición.** | | |
|---|---|---|
| **Texto** | **Similitud coseno** | **Resultado** |
| Azteca1 | 0.06471 | Esperado |
| Freud41 | 0.0216 | Esperado |

En la tabla anterior se puede ver que el documento Azteca 1 es el más relevante debido a que su similitud con la petición es mayor que la del documento Freud 41. En la petición se toman las palabras: documento y tesis ya que ambas son sustantivos, en el documento Azteca1 solamente aparece una sola vez la palabra documento, la palabra tesis no aparece; en el documento Freud41 la palabra tesis aparece dos veces y la palabra documento no aparece; por lógica es de esperarse que el documento con mayor similitud es Freud41 pero no lo es debido a los siguiente factores:

- El documento Freud41 contiene mayor número de palabras que el documento Azteca1.

- La palabra tesis tiene menor porcentaje de importancia en el documento Freud41 que la palabra documento en el texto Azteca1, esto se puede observar en las siguientes tablas y gráficas.

Tabla 5.2. Palabras más representativas del documento Freud 41

| Documento Freud41 | |
|---|---|
| Palabra | Porcentaje (%) |
| Freud | 8.8 |
| Parte | 3.6 |
| Conciencia | 3.6 |
| Recuerdo | 3.2 |
| Sentimiento | 2.8 |
| Tesis | 0.8 |

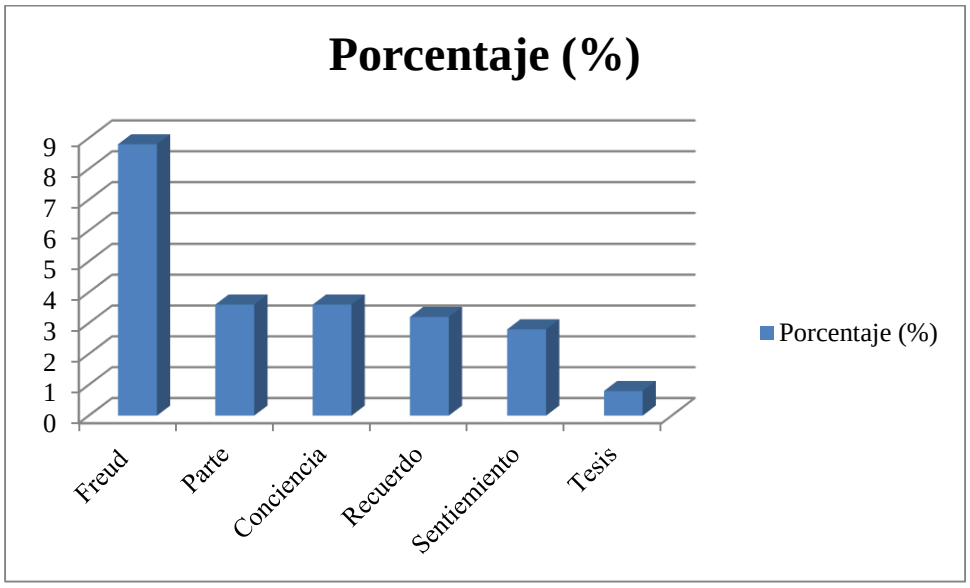



Figura 5.1. Porcentaje de las palabras más representativas del doc. Freud41

En la tabla y la gráfica anterior se puede observar que la palabra tesis solo representa el 0.8% del contenido del documento es decir un porcentaje muy pequeño.

En la siguiente tabla y figura se muestran las palabras más representativas del documento Azteca1.

Tabla 5.3. Palabras más representantes del Documento Azteca1.

| Documento Azteca1 | |
|---|---|
| Palabra | Porcentaje (%) |
| Telmex | 15.21 |
| Dish | 15.21 |
| Empresa | 6.52 |
| Acuerdo | 6.52 |
| Documento | 2.17 |

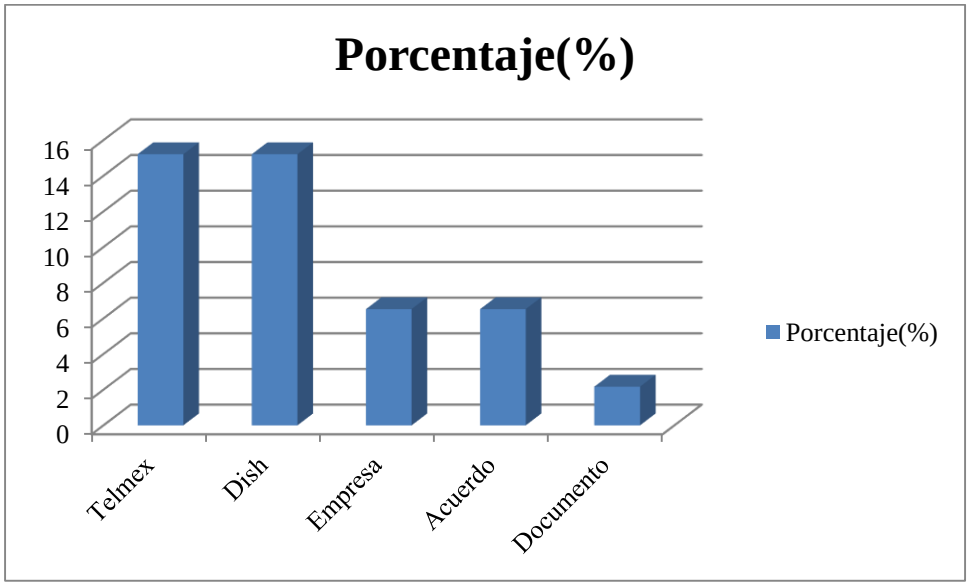



Figura 5.2 Porcentaje de las palabras más representativas del doc. Azteca1

De acuerdo con la tabla 13 y la figura 20 la palabra documento tiene más porcentaje de relevancia que la palabra tesis es por eso que el método recupera el documento Azteca1 con mayor similitud que el documento Freud41, entonces la respuesta del método es la correcta.

## 5.2 BASELINE.

El método *Baseline* se refiere a comparar palabra por palabra de toda la frase de la petición dentro de cada uno de los textos de prueba, si encuentra por lo menos una vez la oración dentro de alguno de los documentos de la colección, este regresara el nombre y la ruta del archivo (ver la codificación en el anexo B). De esta manera aseguramos que la petición no sea parte del texto de prueba y que los resultados que está devolviendo el método propuesto son los que contienen términos de la petición de usuario, es decir los que satisfacen la petición del usuario.

En el ejemplo anterior la petición que se hace es: “Algún documento que hable sobre una tesis”. Usando el *Baseline* lo que se va a comparar es lo siguiente “Algún, documento, que, hable, sobre, una, tesis” es decir se va a comparar palabra por palabra y si se encuentra la misma secuencia significa que la oración es parte del documento.

Al realizar dicha prueba la petición no se encontró en el documento por lo que nuevamente podemos ver que la metodología propuesta funciona.

## 5.3 MEDIDAS DE EVALUCACIÓN

En el capítulo dos se muestran las medidas de evaluación, cuyas expresiones se pueden ver en la tabla 2, para este ejemplo se obtuvo la siguiente evaluación:

**Precisión.**

Número de documentos relevantes = 2

Numero de documentos recuperados = 2

Precisión = 2/2 = 1 o 100%

***Recall***

*Recall*=2/2 = 1

Si evaluamos el *Baseline* bajo las mismas circunstancias obtenemos lo siguiente:

**Precisión:**

Número de documentos relevantes = 2

Numero de documentos recuperados = 0

Precisión = 0/2 = 0 o 0%

***Recall***

*Recall = 0*

De acuerdo a la evaluación anterior el modelo propuesto es mejor que *Baseline*, ya que no realiza la recuperación de información por medio de la igualdad de las palabras, si no por medio de un análisis del texto.

## 5.4 COMPARACIÓN CON OTROS MÉTODOS.

Con la evaluación hecha en la sección anterior podemos hacer una comparación del método propuesto con el método *baseline*. En la siguiente tabla se hace dicha comparación:

Tabla 5.4 Comparación con otro método.

| Método | Precisión | *Recall* | Evaluación |
|---|---|---|---|
| Propuesto | 1 | 0 | bueno |
| *Baseline* | 0 | 1 | malo |

Con la tabla 5.4 se observa que el método propuesto es más eficiente que el método que se ha tomado como referencia, cabe mencionar que se eligió este método debido a que durante la investigación no se encontró con algún otro método que realice recuperación de información, ya que solo los hay en otros idiomas como lo es el inglés, alemán, etc.

El método propuesto puede tener la desventaja que si la petición del usuario es una sola palabra *Baseline* es más eficiente, pero generalmente las peticiones son oraciones, por lo que el método propuesto termina siendo más factible y sobre todo confiable.

## 5.5 ERRORES

El método puede llegar a equivocarse y eso se debe a los siguientes factores:

- El método depende de *FreeLing,* por lo que si esta herramienta comete un error, el método a su vez lo cometerá. Esta herramienta es muy utilizada por lo que se espera que en un futuro tenga mejores y su índice de error disminuya.

- La petición este mal escrita o contiene faltas de ortografía, por ejemplo si proponemos una petición de búsqueda que diga: “la caída de los precios del cafe”.

Al no poner el acento en la letra “e” *FreeLing* lo tomara como un adjetivo y no como un sustantivo, por lo tanto el método lo descartara y no será un término de interés.

- La petición este mal construida, es decir que la idea no este expresada con los términos que definen la petición, eso confundirá al modelo y hará que cometa errores.

# CAPÍTULO 6. CONCLUSIONES

## 6.1 CONCLUSIONES

Después de realizar las pruebas con las peticiones de usuario, pudimos encontrar que el método propuesto realiza ciertas mejoras debido al análisis que realiza ya que le permite recuperar información bajo cualquier petición sin importar cuál sea.

Una de las desventajas que tiene el método es no poder buscar la frase de petición en todo el documento, debido a que el método solo trabaja con los sustantivos, sin embargo se espera que con posibles mejoras a futuro esta desventaja vaya disminuyendo.

Por otra parte se esperan mejoras de la herramienta *Freeling*, debido a que el método propuesto depende de esta herramienta y si dicha herramienta llega a cometer un error lo mismo lo hará el método.

El uso de la medida tf-idf y la función de similitud permitieron que el método lograra recuperación de información relevante, tuvo mejoras en comparación del método *Baseline.*

El objetivo del método es recuperar información sin embrago las aplicaciones que pueden derivar son muchas entre ellas: bibliotecas virtuales, búsqueda de expedientes en centros de readaptación social, criminología, entre otros.

Sabemos que como seres humanos no podemos cambiar en cuanto a nuestra manera de hablar o expresarnos y es lo que en ocasiones obstaculiza la comunicación entre hombre y máquina y hace que el trabajo de búsqueda de información sea de alguna manera complejo, entonces una opción es enseñarle a la maquina como hablamos, como nos expresamos y ella podrá procesar información igual que nosotros optimizando el trabajo. Al término de este trabajo se hacen aportaciones para mejorar la recuperación de información.

## 6.2 TRABAJO A FUTURO

El método propuesto trabaja con los sustantivos ya que son los que pueden definir el contenido de un texto, en recuperación de información se busca que dicha recuperación sea exacta, precisa y confiable, por lo que podemos esperar que el método trabaje sinónimos, ya que en el español existen distintas palabras que significan lo mismo. De esta manera el método puede ser más eficaz, preciso y sobre todo confiable, así también se podría probar con documentos más extensos.

Una de las desventajas que tiene el modelo bajo el método de *Baseline* es que si la frase de búsqueda se encuentra en el texto, el método propuesto puede tener una precisión menor, sin embrago con futuras mejoras puede llegar a eliminar estas desventajas.

El presente trabajo aborda la tarea de recuperar información, proponiendo un método que utiliza diversas herramientas, una de ellas y la más importante es *FreeLing*, si en un futuro esta herramienta logra tener mejoras aumentara la confiabilidad del modelo, teniendo márgenes de error cada vez menores.

### 6.3 CONTRIBUCIONES

Los beneficios derivados del desarrollo de este metodología propuesta en Recuperación de Información con el empleo de técnicas en Procesamiento de Lenguaje Natural, puede ser ampliamente usado para aplicaciones y sistemas que integren la administración y recuperación de información, mejorando el funcionamiento de manera significativa usando herramientas de procesamiento de texto, en áreas de investigación los beneficios son dirigidos a motivar al desarrollo de más herramientas basados en Procesamiento de Lenguaje Natural y ciencias computacionales que enfrenten los problemas actuales que genera el análisis del lenguaje y la recuperación de información.

Otros beneficios que otorga el método propuesto son los siguientes:

- Desarrollar una herramienta en recuperación de información que implique el uso de procesamiento de lenguaje natural, que refleje las ventajas y características de este método.
- Utilizar la metodología propuesta como una base, para desarrollar más métodos en recuperación de información en textos en español, nuevas aplicaciones, tales como, optimización de buscadores web, recuperación de información en bases textuales, clasificadores de texto, sistemas de resúmenes automáticos, búsqueda de información en bibliotecas virtuales.
- La introducción de Procesamiento de lenguaje Natural mejora los resultados de recuperación, el tratamiento un documento en base a su contenido de información de un texto es prescindible.
- El método propuesto ofrece una perspectiva novedosa para el análisis de texto, extendiéndose al manejo de contenido informativo de un documento.

**ANEXO A**

Programa del método propuesto en el lenguaje de programación Python. La entrada es la petición del usuario, es procesada mediante el siguiente programa y a la salida es la ruta del texto que satisface la petición del usuario.

El programa tiene cuatro funciones: genera_diccionario, genera_dic_peticiones, calcula_tf-idf y la func_similitud. La primera función genera los diccionarios donde se va a guardar los términos de interés (sustantivos) y para cada uno de ellos guardara su medida tf, idf y tf_idf , en esta función también se calcula el tf de cada término de interés.

La función genera_dic_peticiones: crea los diccionarios que contiene el lema de los términos de interés (sustantivos) y les asigna un peso tf_idf de 1.

La función calcula_tf_idf: realiza las operaciones correspondientes para el cálculo de la medida tf-idf de cada palabra que conforma todos los textos de la colección, dicha medida es la que se ocupa para representar al texto en un vector que posteriormente será utilizada.

La función_similitud: calcula la función similitud coseno entre el vector de la petición con los vectores de cada uno de los textos mostrando la ruta y nombre del texto que satisface la consulta del usuario.

```
#!/usr/local/bin/python
#coding: latin-1
'''
Created on 11/05/2014

@author: Daniel Valencia, Omar Zuñiga
'''
```

```
from __future__ import division
from sets import Set

class content(object):
    def __init__(self):
        self.word = ""
        self.lema = ""
        self.kind = ""
        self.tf = 0
        self.idf = 0
        self.tf_idf = 0
        self.conjunto = 0
        self.tf2 = 0
        self.lema2 = ""


def genera_diccionario(documents, total):
    for doc in documents:
        try:
            g = open (doc + "_analysis.txt", 'r')
            aux = doc + "_freq.dat"
            dic = shelve.open(aux, 'n')
        except IOError:
            print "Error"
            #print "I/O error({0}): {1}".format(e.errno, e.strerror)
            exit(1)

        for linea in g.readlines():
            linea = linea[:len(linea)-1]
            if len(linea) > 1:
                tupla = linea.split(" ")
```

```
        clave = tupla [2]
        try:
            if clave.index('N') == 0:
                lema = tupla[1]

                if not total.has_key(lema):
                    total[lema] = 0

                if dic.has_key(lema):
                    aux = dic[lema]
                    aux.tf += 1
                    dic[lema] = aux
                else:
                    aux = content()
                    aux.word = tupla[0]
                    aux.lema = lema
                    aux.kind = clave
                    aux.tf = 1
                    dic[lema] = aux
        except ValueError:
                a=0
g.close()

tam = len(dic.keys())
for item in dic.keys():
    aux = dic[item]
    aux.tf = aux.tf/tam
    dic[item]=aux

dic.close()
```

```
def genera_dic_peticiones(peticiones):
    for doc in peticiones:
        try:
            g = open (doc + "_analysis.txt", 'r')
            aux = doc + "_freq.dat"
            dic = shelve.open(aux, 'n')
        except IOError:
            print "Error"
            #print "I/O error({0}): {1}".format(e.errno, e.strerror)
            exit(1)

        for linea in g.readlines():
            linea = linea[:len(linea)-1]
            if len(linea) > 1:
                tupla = linea.split(" ")
                clave = tupla [2]
                try:
                    if clave.index('N') == 0:
                        lema = tupla[1]
                        if dic.has_key(lema):
                            aux = dic[lema]
                            aux.tf2 += 1
                            dic[lema] = aux
                        else:
                            aux = content()

                            aux.lema2 = lema

                            aux.tf2 = 1
                            dic[lema] = aux
                except ValueError:
```

```
                a=0
        g.close()
        dic.close()

def calcula_tfidf(documents, total):

    for idx,doc in enumerate(documents):
        print "Analizando el documento "+str(idx)
        doc_dic = shelve.open(doc+"_freq.dat",'r')
        for key in doc_dic.keys():
            total[key] += 1
        doc_dic.close()
    ##################################################
    for doc in documents:
        doc_dic = shelve.open(doc+"_freq.dat",'w')
        for key in doc_dic.keys():
            aux = doc_dic[key]
            idf = total[key]
            idf= math.log10 (13/total[key])
            aux.idf=idf
            aux.tf_idf = aux.tf * idf
            doc_dic[key]=aux
        doc_dic.close()
    print "Ya termine"

def func_similitud(documents,peticiones):
    acumulador = 0
    resultado = {}
            aux = obj[key]

            if doc.has_key(aux.lema2):
```

```
            #print "Lema del documento: "+doc[aux.lema2].lema
            #print "Lema de la peticion: "+aux.lema2
            aux2 =doc[aux.lema2]
            acumulador += aux2.tf_idf

        norma = 0
        for key in doc.keys():
            aux3 = doc[key]
            norma += aux3.tf_idf*aux3.tf_idf
        norma *= math.sqrt(len(obj.keys()))

        norma = math.sqrt(norma)
        similitud = acumulador/norma

        resultado[doc2]=similitud
        print doc2+"---->"+str(similitud)
        doc.close()


            #else:
             #   acumulador = 0

if __name__ == '__main__':


    total = {}

    documents = ["C:\\TesisDaniel\\documentos\\Poli1",
                 "C:\\TesisDaniel\\documentos\\apple",
                 "C:\\TesisDaniel\\documentos\\Asteroide1",
                 "C:\\TesisDaniel\\documentos\\azteca1",
```

```
            "C:\\TesisDaniel\\documentos\\bolsa1",
            "C:\\TesisDaniel\\documentos\\freud21",
            "C:\\TesisDaniel\\documentos\\freud31",
            "C:\\TesisDaniel\\documentos\\freud41",
            "C:\\TesisDaniel\\documentos\\Investigadores1",
            "C:\\TesisDaniel\\documentos\\lindavista1",
            "C:\\TesisDaniel\\documentos\\Pemex1",
            "C:\\TesisDaniel\\documentos\\Tlalpan1",
            "C:\\TesisDaniel\\documentos\\Whats1"
            ]

peticiones = [
             "C:\\TesisDaniel\\peticiones\\pet1",
             "C:\\TesisDaniel\\peticiones\\pet3",
             "C:\\TesisDaniel\\peticiones\\pet4",
             "C:\\TesisDaniel\\peticiones\\pet5",
             "C:\\TesisDaniel\\peticiones\\pet7",
             "C:\\TesisDaniel\\peticiones\\pet8",
             "C:\\TesisDaniel\\peticiones\\pet9",
             "C:\\TesisDaniel\\peticiones\\pet10",
             "C:\\TesisDaniel\\peticiones\\pet11",
             "C:\\TesisDaniel\\peticiones\\Pet12",
             "C:\\TesisDaniel\\peticiones\\pet13",
             "C:\\TesisDaniel\\peticiones\\pet14",
             "C:\\TesisDaniel\\peticiones\\pet15",
             "C:\\TesisDaniel\\peticiones\\pet16",
             "C:\\TesisDaniel\\peticiones\\pet17",
             "C:\\TesisDaniel\\peticiones\\pet18",
             "C:\\TesisDaniel\\peticiones\\pet19",
             "C:\\TesisDaniel\\peticiones\\pet20",
             "C:\\TesisDaniel\\peticiones\\pet21",
```

```
            "C:\\TesisDaniel\\peticiones\\pet22",
            "C:\\TesisDaniel\\peticiones\\pet23",
            "C:\\TesisDaniel\\peticiones\\pet24"]


    #genera_diccionario(documents, total)

    #calcula_tfidf(documents, total)
    #genera_dic_peticiones(peticiones)

    func_similitud(documents,peticiones)
    '''

    for item in dic.keys():
        print dic[item].lema + "---->" + str(dic[item].tf) +"---->"+ str(dic[item].idf)+"---->"+
str(dic[item].tf_idf)
    dic.close()
    '''
```

**ANEXO B**

Programa del método *Baseline* en el lenguaje de programación Python. La entrada para este programa es la petición del usuario; teniendo como salida la ruta del texto que contenga la petición del usuario.

Este programa segmente en palabras la petición del usuario, posteriormente comienza a comparar palabra por palabra de la petición en cada uno de los textos de la colección, si encuentra las palabras de la petición devuelve el nombre y la ruta del texto que contiene dichas palabras, en caso contrario devuelve la expresión: ERROR frase no encontrada.

```
#!/usr/local/bin/python
#coding: latin-1
'''
Created on 11/05/2014

@author: Daniel Valencia, Omar Zuñiga
'''
from __future__ import division
import shelve,copy
import math
from sets import Set
from string import punctuation
def wordsNgrams( min_size, max_size, oracion):
    Words = []
    words = []
    aux = []
    for i in xrange (min_size, max_size+1):
```

```
        Words.append({})


    items = oracion.split(" ")
    for item in items:
        if (item != "") and (item not in punctuation):#Eliminate the empty elements caussed by the multiple spaces between words
            while item[0] in punctuation:#Eliminate the punctuation marks at the begin and at the end
                item = item[1:]
                if len(item) == 0:
                    break
            while (len(item) > 0) and (item[len(item)-1] in punctuation):
                item = item[:len(item)-1]
            words.append(item.lower())
    aux = copy.deepcopy(words) #Save a copy of the original list
    ############################################################


    if min_size <= len(words):
        if min_size > 0:


            if max_size > len(words):
                #print "Message: The maximum size in wordsNgrams function  is bigger than the size of the paragraph. The maximum size will be set as the length of the paragraph"
                print  "\tMessage: The maximum size in wordsNgrams function  is bigger than the size of the paragraph. The maximum size will be set as the length of the paragraph\n"


                max_size = len(words)


            for tam in xrange(min_size, max_size+1):
                '''
```

```
            del words[:]
            words = copy.deepcopy(aux)
            if tam > 1:
                for m in xrange(0,tam-1):
                    words.insert(0, "#")
                    words.append("#")
            '''
            for i in xrange(0,len(words)-tam+1):
                ngram = "["

                j = 0
                while j <= tam-1:
                    ngram += words[i+j]+","
                    j += 1

                ngram=ngram.rstrip(",")
                ngram +="]"
                #Se actualiza el diccionario de los Ngramas


                if Words[tam - min_size].has_key(ngram):
                    Words[tam - min_size][ngram] += 1
                else:
                    Words[tam - min_size][ngram] = 1
        else:
            #print "ERROR: The minimum size in wordsNgrams function must be greater than 0"
            print "\tERROR: The minimum size in wordsNgrams function must be greater than 0\n"
```

```
        else:
            #print "ERROR: The minimum size in wordsNgrams function is bigger than the size of the paragraph"
            print "\tERROR: The minimum size in wordsNgrams function is bigger than the size of the paragraph\n"
        return(Words)


    ############################
def len_wordsNgrams( ngram):
    size = ngram.count(",")+1
    return (size)




if __name__ == '__main__':


    total = {}

    documents = ["C:\\Corpus\\poli1",
                 "C:\\Corpus\\apple1",
                 "C:\\Corpus\\Asteroide1",
                 "C:\\Corpus\\TvAzteca1",
                 "C:\\Corpus\\bolsa",
                 "C:\\Corpus\\freud2",
                 "C:\\Corpus\\freud3",
                 "C:\\Corpus\\freud4",
```

```
        #"C:\\Corpus\\Investigadores1",
        "C:\\Corpus\\lindavista1",
        "C:\\Corpus\\Pemex1",
        #"C:\\Corpus\\Tlalpan1",
        #"C:\\Corpus\\Whatts1"
        ]

peticiones = [
         "C:\\peticiones\\pet1.txt",
         "C:\\peticiones\\pet3.txt",
         "C:\\peticiones\\pet4.txt",
         "C:\\peticiones\\pet5.txt",
         "C:\\peticiones\\pet7.txt",
         "C:\\peticiones\\pet8.txt",
         "C:\\peticiones\\pet9.txt",
         "C:\\peticiones\\pet10.txt",
         "C:\\peticiones\\pet11.txt",
         "C:\\peticiones\\Pet12.txt",
         "C:\\peticiones\\pet13.txt",
         "C:\\peticiones\\pet14.txt",
         "C:\\peticiones\\pet15.txt",
         "C:\\peticiones\\pet16.txt",
         "C:\\peticiones\\pet17.txt",
         "C:\\peticiones\\pet18.txt",
         "C:\\peticiones\\pet19.txt",
         "C:\\peticiones\\pet20.txt",
         "C:\\peticiones\\pet21.txt",
         "C:\\peticiones\\pet22.txt",
         "C:\\peticiones\\pet23.txt",
         "C:\\peticiones\\pet24.txt"]
eval = {}
```

```
obj = open ("C:\\peticiones\\pet24.txt",'r')
for ln in obj.readlines():
    if ln != "":
        min_size = len (ln.split(" "))
        max_size = min_size



        ngram = wordsNgrams(min_size, max_size, ln)
obj.close()
peticion = ngram[0].keys()[0]



            min_size = len (ln.split(" "))
            max_size = min_size
            corpus = wordsNgrams(min_size, max_size, ln)
            if corpus[0].has_key(peticion):
                if eval.has_key(doc):
                    eval[doc] += 1
                else:
                    eval[doc] = 1
print eval
```

## REFERENCIAS


Atserias J.TXALA un analizador libre de dependencias para el castellano *Procesamiento del Lenguaje Natural*, n. 35, pg. 455--456. September, 2005.

Baeza-Yates, R. (1999). *Modern information retrieval* (Vol. 463). New York: ACM press.

Bekkerman, R. (2005) Using Bigrams in Text Categorization, CIIR Technical Report.

Bennett, G. (2008, January). A comparative study of probabilistic and language models for information retrieval. In*Proceedings of the nineteenth conference on Australasian database-Volume 75*(pp. 65-74). Australian Computer Society, Inc..

Bolshakov, I. (2004). *Computational Linguistics Models, Resources, Applications*. CIENCIA DE LA COMPUTACIÓN.

Burkowski, F. (1992, June). Retrieval activities in a database consisting of heterogeneous collections of structured text. In *Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 112-125). ACM.

Carreras, X. A simple named entity extractor using adaboost. *Proceedings of CoNLL-2003 Shared Task* Edmonton, Canada, June 2003.

Codina, L. (1995). Teoría de recuperación de información: modelos fundamentales y aplicaciones a la gestión documental. *Information World en español*, *38*, 18-22.

Croft, W. (1987). Approaches to intelligent information retrieval. *Information Processing & Management, 23*(4), 249-254

Dan,R. A relational feature extraction language (fex). *Technical report* University of Illinois at Urbana Champaign, 2004. http://l2r.cs.uiuc.edu/~ cogcomp/software.php

Ferreira, A. (2009). Disminución de la sobrecarga de información en la World Wide Web a partir de interacciones dialógicas hombre-computador. *Revista signos*, *42*(69), 9-27.

Frakes, W. (1992). Information retrieval: data structures and algorithms.

Gelbukh, A. (2006). *Procesamiento automático del español con enfoque en recursos léxicos grandes*. Centro de Investigación en Computación, Instituto Politécnico Nacional.

Gelbukh, A. (2007). *Investigaciones en análisis sintáctico para el español*. Instituto Politécnico Nacional, Dirección de Publicaciones.

Greengrass, E. (2000). Information retrieval: A survey.

Harman, D. (1992). Ranking Algorithms.

Hiemstra, D. (2009). Information retrieval models. *Information Retrieval: searching in the 21st Century*, 1-19.

Kowalski, G. (2011). *Information retrieval architecture and algorithms* (pp. 95-139). New York: Springer.

Kraft, D. (1979). A mathematical model of a weighted Boolean retrieval system. *Information Processing & Management*, *15*(5), 235-245.

Losee, R. (1997). Comparing Boolean and probabilistic information retrieval systems across queries and disciplines. *JASIS*, *48*(2), 143-156.

Manning, C.*Introduction to information retrieval*. Cambridge: Cambridge university press, 2008.

Martínez , J. A. (2006). Los modelos clásicos de Recuperación de información y su vigencia.

Martínez, F. (2004). *Recuperación de información: modelos, sistemas y evaluación*. Murcia: Kiosko, 2004..

Oliván, S. (2006). Una aproximación al concepto de recuperación de información en el marco de la ciencia de la documentación. *Investigación bibliotecológica*, *20*(41), 13-43.

Piek Vossen (ed) EuroWordNet: A Multilingual Database with Lexical Semantic Networks Kluwer Academic Publishers, Dordrecht, 1998.

Sagayam, R,.A Survey of Text Mining:Retrieval, Extraction and Indexing Techniques, IJCER, Vol. 2 Issue. 5, sep 2012, PP: 1443-1444.

Salton G. A vector space model for information retrieval. *Communications of the ACM*, 613–620, November 1975

Salton, G., & McGill, M. J. (1983). Introduction to modern information retrieval.

Sharma. A Survey on Information Retrieval Models, Techniques And Applications.

Sidorov G. (2013) Construcción no lineal de n-gramas en la Lingüística Computacional; n-gramas sintácticos, filtrados y generalizados, México: Sociedad Mexicana de Inteligencia Artificial, 2013.

Soon,W. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 27(4):521-544, 2001.

Thorsten,B. TnT: A statistical part-of-speech tagger. *Proceedings of the 6th Conference on Applied Natural Language Processing.* ANLP, ACL. 2000

Zhai, C. (2011, October). Adaptive term frequency normalization for bm25. In *Proceedings of the 20th ACM international conference on Information and knowledge management* (pp. 1985-1988). ACM.

Padró,L.(1998). Un entorno híbrido para el Etiquetado de sintaxis-semántica. PhD tesis Dept. I Sistemes Informàtics Llenguatges. Universitat Politècnica de Catalunya,

http://lucene.apache.org/

http://www.wumpus-search.org/

http://nlp.lsi.upc.edu/freeling/doc/tagsets/tagset-es.html}