BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

Detección Automática de Temas Importantes

Tesis que para obtener el grado de

Licenciado en Ciencias de la Computación

Presenta

Roberto Carlos Ortiz Méndez

Asesor: Dr. David Eduardo Pinto Avendaño
Coasesor: M.C. Mireya Tovar Vidal

Septiembre 2010

DEDICATORIA

Dedico mi tesis a mis padres: Juan Sergio Ortiz Martínez y Ma. Guadalupe Méndez Damián por su cariño, comprensión, sacrificio y amor incondicional que siempre me han brindado, por alentarme a cumplir todas mis metas y objetivos, pero sobre todo por la confianza que han depositado en mí, por lo que, la realización de esta tesis, es una muestra de la gratitud y amor que tengo así ellos.

A mis hermanos: Víctor Hugo y Ma. Del Carmen, ya que siempre estuvieron ahí para apoyarme, en los buenos y malos momentos, y formar parte de la inspiración que me hace seguir adelante.

A mis abuelos: por su amor y cariño, sus consejos y orientación, que en base a sus experiencias, supieron darme la mejor opinión para hacer siempre lo correcto.

Por último una dedicatoria especial a Marisol Villa Corona, por estar siempre presente en los momentos especiales, por brindarme su comprensión, cariño y el apoyo que siempre me ha manifestado.

AGRADECIMIENTOS

Quisiera agradecer a mis asesores: Dr. David Pinto Avendaño y M.C. Mireya Tovar Vidal, por la ayuda, paciencia, atención constante y orientación, así como el conocimiento transmitido durante la realización de mi trabajo de tesis.

También a todos mis amigos y compañeros, ya que siempre me brindaron su apoyo, amistad y confianza,

De la misma forma, agradezco el financiamiento parcial para la realización de este trabajo de tesis por parte de:

CONACYT (Proyecto # 106625)

PROMEP (Grant# 103.5/09/4213).

INDÍCE GENERAL

INTRODUCCIÓN

La internet se ha vuelto día a día una forma natural para disponer de información en forma textual, debido a su rapidez y eficiencia, un ejemplo claro es a través de las numerosas publicaciones en medios electrónicos, la cual ha crecido de forma exponencial, en ocasiones es tanta la información obtenida, que es imposible leer toda, es por esta razón, que surge la importancia de desarrollar un método que permitan localizar de manera clara términos clave o importantes, los cuales nos den una aproximación acerca de los temas tratados en dichos textos.

Los términos importantes de un texto son aquellos que de alguna manera describen o capturan el contenido semántico del mismo. Estos términos pueden ser considerados incluso una especie de resumen y por tanto son de gran interés para diversas tareas del Procesamiento del Lenguaje Natural (PLN).

La extracción automática de términos importantes ha sido un tema estudiado durante varios años, como es el caso de la semántica de metadatos, útil para tareas como resúmenes, síntesis [Barzilay and Elhadad, 1997; Lawrie et al., 2001; Davanzo and Magnini,2005], pero no es, sino años posteriores, cuando se reconoce el verdadero impacto de obtener buenos términos clave los cuales podrían mejorar la calidad de varias aplicaciones dentro del PLN [Frank et al., 1999; Witten et al., 1999; Turney, 1999; Barker and Corrnacchia, 2000].

Por lo tanto, la selección de términos importantes que se encuentran dentro del contenido de un texto puede ser usado para mejorar el desempeño de sistemas o tareas del PLN, tales como recuperación de información, clustering, question-answering, resúmenes automáticos, etc., son algunos ejemplos de las posibilidades de aplicación de un sistema para la detección automática de términos importantes.

En general, un término clave podría ser considerado como una secuencia de una o más palabras que captan el tópico principal de un documento, por lo que se espera que estos términos represente lo que el autor trata de expresar o dar a conocer acerca del documento.

En numerosas ocasiones se hallan términos sin ningún significado, para identificar estas palabras, es necesario la realización de un pre-procesamiento de la información, dentro de este pre-procesamiento se encuentran métodos tales como la eliminación de palabras cerradas (Stopwords) como son artículos, preposiciones, etc., truncamiento de palabras, entre otros.

Se tomaron dos diferentes técnicas para el análisis de textos para la realización de un algoritmo capaz de realizar dicha tarea: Secuencias Frecuentes Maximales para extraer una

secuencia de una o más palabras de un documento determinado y Pageranking, esperando extraer de estas palabras, secuencias que representen las ideas principales que el autor de un determinado documento trata de dar a conocer.

Los términos clave obtenidos fueron evaluados en base a su precisión, recall y F-score con respecto a dos gold-standards: 1) términos clave dados por el lector y 2) un conjunto resultante de la combinación de los términos clave dados por el lector y los términos dados por el autor.

Los resultados derivados de la evaluación con respecto al gold-standard fueron comparados con tres distintas baselines: uno no supervisado (TF-IDF based) y dos supervisados (Naïve Bayes y Maximum Entropy), con lo que se pudo observar que el algoritmo propuesto en esta tesis, obtuvo mejores resultados que los baselines.

CAPÍTULO I

MARCO DE REFERENCIA

En este capítulo se presenta la descripción del problema a tratar, así como los objetivos, justificación, hipótesis y trabajos relacionados que respaldan la realización del presente trabajo.

1.1 Descripción del problema

En este trabajo de tesis se experimentará con un corpus de documentos de índole científico. El objetivo principal consiste en aplicar métodos automáticos para la detección de términos clave y así extraer los temas centrales aludidos a dicho texto.

Se hará uso de técnicas relacionadas con la recuperación de información, el procesamiento del lenguaje natural, así como la clasificación de documentos textuales.

En particular, se aplicarán métodos estadísticos basados en la frecuencia para determinar la influencia que podría tener uno o más términos en el discurso sostenido en los documentos de análisis. Estos métodos incluyen aquellos basados en Secuencias Frecuentes maximales y Pageranking.

Dado que se cuenta con un corpus basado en artículos científicos, los experimentos a llevar a cabo se harán en base a Test-data extraída de la tarea #5 de SemEval-2010, que tiene como nombre: "*Automatic Keyphrase Extraction from Scientific Articles*" [SEMEVAL-2].

En resumen, se trata de aplicar técnicas del procesamiento del lenguaje natural para la detección automática de términos clave, que se encuentran dentro de artículos científicos, los cuales deben ser buenos, ya que la calidad de estos términos tiene un impacto directo sobre la calidad del método sobre las aplicación en el PLN.

1.2 Justificación

Los términos importantes de un texto son aquellos que de alguna manera describen o capturan el contenido semántico del mismo. Estos términos pueden ser considerados incluso una especie de resumen y por tanto son de gran interés para diversas tareas del Procesamiento del Lenguaje Natural (PLN). Tareas como recuperación de información, clustering, question-answering, resúmenes automáticos, etc., son algunos ejemplos de las posibilidades de aplicación de un sistema para la detección automática de términos importantes.

Existe motivación suficiente para llevar a cabo un trabajo de investigación y análisis en esta área debido a su aplicación en tareas del PLN. Si bien, sistemas como los motores de búsqueda podrían resultar beneficiados, en nuestro caso, se trata principalmente de aplicar los algoritmos de detección automática de tópicos sobre un conjunto de documentos pertenecientes a artículos científicos.

Hasta el momento, y hasta donde sabemos, no existe un sistema totalmente funcional que encuentre de manera automática aquellos términos científicos importantes, dado un conjunto de artículos.

1.3 Hipótesis

Consideramos que el desarrollo y la aplicación de métodos y técnicas del procesamiento textual (en especial, detección automática de términos clave) aplicadas a dominios relacionados con artículos científicos, tendrá un rendimiento aceptable, comparado con aquellos generados de manera manual.

Por tanto, se formulan las siguientes hipótesis:

1. Es posible inferir los temas importantes para un determinado artículo científico dado los términos clave extraídos del mismo.

2. La técnica de detección automática de tópicos que se implantará en el sistema final tendrá un rendimiento superior a otras presentadas en la literatura.

3. La presentación adecuada de resultados en el sistema basado en la extracción de términos clave será utilizada para aquellos usuarios interesados en los distintos temas contenidos en los documentos.

1.4 Objetivos generales y particulares

Objetivo general:

Implementar técnicas automáticas dentro del procesamiento del lenguaje natural para la detección de términos clave y así extraer los temas centrales aludidos a dicho texto.

Objetivos particulares:

- Analizar la estructura de los documentos, así como su contenido, los cuales serán usados para el análisis del mismo.

- Estudiar diversas técnicas estadísticas para la detección automática de términos importantes.

- Implementar una técnica o modelo adecuado para el sistema a desarrollar

- Confrontar los resultados obtenidos con otras técnicas presentadas en la extracción de palabras clave.

Los objetivos anteriormente mencionados, nos llevaran a la creación de un sistema capaz de extraer términos clave, que nos indique los tópicos centrales de dicho documento.

1.5 Trabajos relacionados

A continuación se presentaran diversos trabajos relacionados con la extracción automática de términos:

Generalmente los servidores de búsqueda solicitan a los usuarios una pregunta que indique sus necesidades de información. Sin embargo, la pregunta típica suministrada al servidor de búsqueda solo contiene a lo más tres palabras, lo cual hace muy difícil para el servidor de búsqueda identificar los documentos relevantes. Además hay muchos documentos relevantes disponibles, los cuales no son encontrados pues no contienen las palabras exactas usadas en la pregunta.

[Krulwich and Burkey1996] Desarrollaron un agente inteligente llamado InfoFinder, este sistema le pide al usuario que le proporcione algunos documentos de ejemplo, entonces realiza la búsqueda por documentos relevantes. Los documentos de ejemplo suministrados al agente dan información sobre la frecuencia de palabras en los artículos buscados así como algunos patrones o frases comunes que pueden ser importantes para una mejor clasificación de los documentos finales.

El agente InfoFinder utiliza la heurística de extracción de frases significativas de los documentos de ejemplo para el aprendizaje. Las heurísticas son basadas en la observación de elementos que agregan los autores dentro de sus documentos, ya que tienden a utilizar métodos sintácticos para obtener frases claves o ideas en documentos, tales como el uso de palabras resaltadas, la presencia de frases en las cabeceras del texto y el uso de acrónimos.

Su motivación es producir frases, como características cuando se clasifican automáticamente los documentos. Esto le permite aprender los criterios de búsqueda muy general basada en un pequeño número de documentos de una muestra.

Posteriormente, el agente aprende de árboles estándar de decisión para cada categoría de usuario. Estos árboles de decisión se transforman en cadenas de consulta de búsqueda, para los sistemas de búsqueda estándar en lugar de usar los motores de búsqueda especializados.

En general el agente se basa en un proceso que costa de tres pasos:

1. Extracción semántica de las frases significativas de cada documento.

2. Aprendizaje de los árboles de decisión para cada categoría en base a las frases extraídas.

3. Transformar cada árbol de decisión en una búsqueda booleana de cadena de consulta.

En el trabajo [krulwich and Burkey1996] se hace uso de extracción de términos clave para realizar búsquedas, sin embargo su algoritmo tiende a producir una lista relativamente larga de frases o términos clave, por lo tal tiene una baja precisión, en cuestión de extracción de términos claves.

El sistema que presentan [Mateo et. al., 2003] realiza también la extracción de frases para la generación de resúmenes. Su sistema consta de cinco módulos, como:

1. Análisis morfosintáctico
2. Ponderación de frases
3. Detección de anáforas
4. Selección de frases
5. Post-procesado del extracto.

El proceso de síntesis comienza con el análisis morfosintáctico del documento de entrada. Basándose en esta información y en la presencia de diversas características superficiales, el módulo de ponderación asigna puntuaciones a las frases del texto según su importancia. Además, entrega a su salida una lista de las frases candidatas (un porcentaje dado de las frases más importantes).

El módulo de selección de frases escoge las oraciones candidatas que han obtenido mayores puntuaciones, teniendo en cuenta la longitud deseada del resumen y la presencia de referencias anafóricas. Permite realizar tanto extracción de párrafos completos como de frases sueltas.

Una vez seleccionadas las frases del extracto, el módulo de post-procesado comprueba la presencia de ciertas expresiones al comienzo de las mismas, con el objetivo de editarlas si fuera necesario. A su salida entrega el resumen del documento.

[Steier and Belew1993] Usan la información mutua para identificar frases claves de dos palabras. Esta aproximación tiene las mismas limitaciones que [Muñoz1996], cuando se considera como un algoritmo de frase clave, este produce una lista de baja precisión de frases de dos palabras. Comparan la información mutua de pares de palabras dentro de un

área de una temática específica (por ejemplo: documentos referentes a relaciones laborales) y a través de mas colecciones (por ejemplo, documentos legales). En el trabajo de Steier y Belew se hace la observación, de que ciertas frases que parecerían tener un alto grado característico de alguna área (por ejemplo:"union member" podría ser característico de documentos que concierne a relaciones laborales) realmente tienen un alto valor estadístico de información mutua, a través de colecciones más generales (esto quiere decir que, "union member" tiene información mutua alta a través de la colección general legal en lugar del área temática de relaciones laborales).

[Soderland and Lehnert1994] Desarrollaron Wrap-Up, un sistema para la extracción automática de información acerca del domino relacionado con el análisis de un discurso. Wrap-Up usa aprendizaje supervisado para formar un conjunto de clasificadores desde un corpus de entrenamiento de textos representativos, donde cada texto es acompañado por texto, resultado de la salida de un etiquetamiento semántico realizado por un experto.

Wrap-Up esta implementado junto con el algoritmo de arboles de decisión ID3 [Quinlan1986], el cual actúa como un componente de aprendizaje en la extracción de información. Es posible ver las guías predefinidas para un tema o dominio dado, definiendo una plantilla para ser llenada por el sistema de recuperación de información, cada celda es una plantilla y el sistema es manejado por un grupo de arboles de decisiones, que han sido entrenados especialmente para ese caso. Los nodos en el árbol de decisión están basados en características sintácticas del texto, tales como la presencia de ciertas palabras.

. Wrap-Up es un sistema completamente entrenable y es único, ya que no sólo decide que clasificadores son necesarios para el dominio o tema, sino que automáticamente se deriva el conjunto de características para cada uno de los clasificadores.

Durante su ejecución Wrap-Up recibe como entrada todos los objetos extraídos del texto. Cada uno de estos objetos es representado como un "*case frame*" junto con una lista de referencias en el texto, la ubicación de cada referencia, y los patrones lingüísticos usados para extraer estas referencias.

Wrap-Up tiene 6 fases de procesamiento, cada uno de estas con su propio conjunto de arboles de decisión diseñados para transformar los objetos cuando estos pasan por cada una de las fases.

Fases del algoritmo Wrap-Up:

1) Filtración

Cada área del objeto tiene un árbol de decisión propio que juzga si el área contiene información fiable. Descarta el valor del área del objeto si el árbol devuelve negativo.

2) Unión

Crea una instancia para cada par de objetos del mismo tipo. Combina los dos objetos si el árbol de decisión para ese tipo de objeto regresa verdadero. Esta fase puede combinar un objeto con atributos extraídos por separado para ese objeto.

3) Creación de Conexiones

Considera todos los posibles pares de objetos que podrían posiblemente estar vinculadas o ligadas. Añade un apuntador entre objetos si el árbol de decisión devuelve positivo.

4) División de Objetos

Supongamos que el objeto A esta relacionado con el objeto B y el objeto C, si su árbol de decisión devuelve verdadero parte A en dos copias con uno señalando a B y la otra a C.

5) Inferencia de objetos perdidos

Cuando un objeto no tiene otro objeto apuntando a él, una instancia es creada por un árbol de decisión que devuelve el objeto primario más probable. Crea una matriz y la vincula a este objeto "huérfano".

6) Inferencia de Valores Perdidos

Cuando una celda del objeto con una clase cerrada de posibles valores esta vacia, crea una instancia para un árbol de decisión el cual devuelve un valor por defecto sensible del contexto para una celda.

[Turney1999] Introduce el algoritmo GenEx (Genitor plus Extractor) el cual es un hibrido del algoritmo generador de estado continuo genético [whitley1989] y del algoritmo de extracción parametrizada de palabras clave. El extractor funciona asignando un score numérico a las frases en el documento de entrada. El resultado final de esta extracción es esencialmente una lista de frases con el score más alto. El comportamiento de la función de score es determinado por una docena de parámetros numéricos. El generador ajusta estos parámetros, para maximizar el rendimiento del extractor sobre un conjunto de entrenamiento como ejemplo.

La extracción se realiza en 9 pasos los cuales son:

1. Búsqueda Stem simple: Hace una lista de las palabras del texto de entrada, eliminando stop words y convierte las palabras restante a minúsculas. Realiza el stemming por truncamiento usando Lovins o Porter.

2. Score de Stem simple: Por cada stem, cuenta la frecuencia que aparece en el texto, su score es la frecuencia con que aparece en el texto multiplicado por un factor variable.

3. Selección de Stem relevante: Clasifica los stem en orden decreciente del score y hace una lista de estos.

4. Búsqueda de Frases Compuestas de Stem: Hace una lista de todas las frases en el texto de entrada, una frase es definida como una secuencia de una, dos o tres palabras que aparecen consecutivamente en el texto.

5. Score de Frases Compuestas de Stem: Para cada frase cuenta la frecuencia en que aparecen dentro del texto, asigna un score para cada frase, como se realizó en el paso 2.

6. Expansión de Stem simple: Para cada stem de la lista obtenida en el paso 3, busca las frases con el más alto score que contenga el stem buscado, el resultado es una lista de las frases ordenada decrecientemente de acuerdo al score del stem buscado.

7. Eliminación de Duplicados: La lista de frases ordenadas en el paso 6, puede contener frases duplicadas, se eliminan las frases duplicadas, dejando la frase con el más alto score.

8. Añadir Sufijos: Para cada una de las frases restantes, se busca la frecuencia correspondiente a las frases completas (no stemmming), del texto de entrada. Cuando contamos la frecuencia de las frases completas, si la frase tiene al final indicios de ser un posible adjetivo, la frecuencia para esta frase completa es cero. Si los adjetivos se encuentran en medio de las frases estos son aceptados, sólo frases con adjetivos al final de esto son eliminados. Si una frase contiene un verbo, la frecuencia para esta frase es cero.

9. Añadir Principales: Para cada uno de las frases completas (frases con sufijos), busca la principal.

10. Por último se ordena la lista con el score final de las frases, dado por los pasos anteriores.

[Lucio2002] Implementa el sistema ATA, el cual es un sistema de adquisición Automática de términos que procesa cualquier texto dado y crea una lista de sustantivos y frases de sustantivos que probablemente pueden estar unidas terminológicamente en el texto.

El desarrollo en la extracción de frase de sustantivos es una tarea delicada muy limitada por la robustez y precisión. Robustez, significa que puede usarse en una amplia gama de textos sin restricción reunidos en un gran corpora, esto significa que el sistema tiene que ser independiente del dominio; dentro de la precisión el problema radica en que las frases de sustantivos extraídos por el sistema son sólo términos candidatos que se propondrán al usuario para que construya una terminología del dominio

El proceso general del sistema en su conjunto, es decir, si se considera el sistema más grande que contiene ATA, la entrada consiste en un texto, un diccionario, un conjunto de normas que definan el agrupamiento de palabras, y una gramática exterior, de lo contrario la entrada de datos ATA consiste de texto plano previamente analizado, tanto morfológicamente como sintácticamente.

Primero ATA lematiza el texto usando un diccionario externo, el texto resultante es pasado a un proceso post-morfológico, que detectó y forma grupos especiales según la recomposición y reglas de correspondencia. El sistema agrupa las palabras en frases, antes de que el principal proceso comience, el texto se envía a un analizador sintáctico que, usa los grupos que constituyen las frases de la gramática exterior. La principal razón para usar una gramática exterior es la eficiencia y robustez del proceso, de esta manera distinguir las palabras que son sustantivos y las que no lo son.

Posteriormente se realiza la extracción de palabras las cuales serán los términos candidatos. Después de esto, un proceso basado en estadística evalúa la lista de candidatos, en esta etapa, un archivo contiene la frecuencia de cada palabra, calculada sobre el contenido de todo el texto. La salida es producida finalmente y se envía al sistema de usuario. La salida de ATA es dividida en dos grupos, los cuales pueden estar vacios. El primer grupo contiene términos simples identificados en el texto. El segundo contiene los términos compuestos detectados en el texto.

Un especialista debe confirmar la corrección de ambos conjuntos. Dadas las características de los términos simples, su tratamiento se inicia mediante la recopilación de todas las palabras de entrada que aparecen con frecuencias más altas. De esta colección, las palabras clasificadas como sustantivos pueden ser considerados como términos simples candidatos. Aquellos clasificados como desconocidos, pero considerados como sustantivos como consecuencia del proceso sintáctico, también son considerados como candidatos. La lista de candidatos será procesada antes de la salida final. Un sustantivo es considerado como termino simple si se produce con una frecuencia más alta que el umbral basado en el corpus.

Los términos compuestos obedecen a algunas restricciones sintácticas y gramaticales que hacen que el proceso de detección sea más fácil. La estructura de los grupos de

palabras que ha de ser considerados como términos compuestos candidatos son descritos en un archivo externo. Un grupo de palabras es un término compuesto si se ajusta a una estructura definida previamente. Al igual que los términos simples, los términos compuestos también tienen limitaciones de frecuencia dentro del corpus.

Además, el sistema distingue entre palabras de alta frecuencia que ocurren dentro de los términos compuestos, las apariciones aisladas de las mismas palabras. El formato de información también debe ser considerado por el sistema, el formato del texto depende del editor. Es necesario crear una descripción del formato externo, la idea es dar más o menos importancia a una palabra de acuerdo a su formato.

En resumen, el sistema ATA utiliza una combinación de métodos estadísticos y métodos con conocimientos lingüísticos para determinar si una combinación de palabras es un término clave. El uso de esta herramienta es la construcción semiautomática de índices terminológicos, y sistemas de extracción de términos.

[Nakagawa1997] Extrae automáticamente los términos simples y compuestos a partir de técnicas manuales, para crear índices *back-of-book*. Cada término compuesto se obtiene mediante una fórmula que se basa en la frecuencia de términos en un documento. En sus experimentos, Nakagawa evalúa su algoritmo mediante la comparación de los índices generados por un experto con índices generados por su algoritmo.

La principal característica que distingue a los índices *back-of-book* de una lista de palabras clave es la longitud, como señala *Nakagawa*, en un documento usualmente se asignan palabras clave, pero los índices *back-of-book* por lo general contienen términos índice. Además, los términos clave usualmente se destinan a cubrir el documento en su totalidad, pero los términos índice son destinados a cubrir sólo una pequeña parte de un documento.

Un algoritmo de extracción de términos clave podría usarse para generar índices *back-of-book* dividiendo un documento grande en secciones de unas tres páginas cada uno. Un algoritmo para generar índices *back-of-book* no es simplemente por orden alfabético de términos. A menudo existe una estructura jerárquica, donde el término índice mayor es seguido por una lista de términos relacionados con índices menores.

[Frank et al.1999] Han puesto en práctica un sistema, Kea, que se basa en [Turney1999]. Donde trata la extracción de términos clave como un problema de aprendizaje supervisado, que utiliza un enfoque bayesiano.

El algoritmo de Kea se basa en dos pasos:

1. *Training*: crea un modelo para la identificación de términos clave usando documentos de entrenamiento, donde muestran los términos clave del autor.

2. *Extraction*: elige términos clave de un documento utilizando el modelo del training.

Ambas etapas eligen un conjunto de términos candidatos de los documentos de entrada, y luego calculan los valores de ciertos atributos (llamados características) para cada candidato.

Para la selección de los términos clave Kea considera todas las subsecuencias en cada línea y determina cuales de estas son las más adecuadas como términos clave, para esto se basan en las siguientes reglas:

- Los términos clave candidatos se limitan a una determinada longitud máxima (usualmente de tres palabras).

- Los términos clave candidatos no pueden ser nombres propios.

- Los términos clave candidatos no pueden iniciar o finalizar con Stopwords.

Para cada unos de los términos candidatos, se calculan dos características, los cuales son TF.IDF (Term Frecuency, Inverse Document Frecuency)., la cual es una medida de la frecuencia de términos en un documento respecto a su rareza en el uso general y first occurrence, que es la distancia dentro de documento de la aparición de la primera frase.

El modelo que determina la probabilidad de que cada candidato sea un término clave se basa en el modelo Naïve Bayes, en combinación de TFxIDF y *d* (distancia) de cada uno de estos términos.

Se ha evaluado Kea como componente de un nuevo tipo de motor de búsqueda, Keyphind, diseñados especialmente para apoyar la navegación web. Sus experimentos sugieren que ciertos tipos de tareas son mucho más fáciles con Keyphind que con los motores de búsqueda convencionales.

Otros trabajos relacionados son: [Muñoz1996] Usa un algoritmo de aprendizaje no supervisado para descubrir frases clave de dos palabras. El algoritmo está basado en la teoría de redes neuronales y la teoría de resonancia adaptiva (ART). Este algoritmo tiende a producir una larga lista de frases, por tal motivo, tiene baja precisión, también, el algoritmo no es aplicable a frases clave de una palabra o con más de dos palabras. [Damerau1993]

Utiliza una relación sobre la frecuencia relativa entre dos corpus para extraer palabras clave específicas de un dominio. Uno de los problemas de usar la frecuencia relativa es que tiende a asignar un puntaje muy alto para las palabras cuya frecuencia en el corpus es pequeña.

Actualmente existen algunas propuestas relacionadas con el uso de Pagerank dentro del PLN, una de ellas es el presentado en [Mihalcea and Tarau, 2004], donde dan a conocer el algoritmo llamado *Textranking*, el cual maneja un versión ponderada del Pagerank y presentan algunas aplicaciones dentro del PLN usando unigramas, al igual que el Pagerank, construyen términos multi-palabra, haciendo uso de los conexiones existentes entre las palabras y su ranking.

1.6 Estructura de la tesis

El documento se encuentra organizado de la siguiente manera:

- En el Capítulo I se da a conocer un marco de referencia en el cual se menciona la descripción del problema a tratar, la justificación del por qué se realizó este trabajo, hipótesis, objetivos generales y particulares, así como la mención de algunos trabajos relacionados con la extracción de términos clave de documentos.

- En el capítulo II se revisa una serie de conceptos teóricos enfocados al trabajo de tesis, los cuales nos dan una perspectiva del cómo se aplican dentro del trabajo de tesis, estos conceptos abarcan: Secuencias Frecuentes Maximales y Pagerank, también se habla acerca del pre-procesamiento que se realiza a los textos, así como las métricas de evaluación que se usan, así como el modelo de evaluación que está basado en SemEval-2, por último se da a conocer otros modelos utilizados para la extracción de términos clave.

- Dentro del Capítulo III se presenta la parte central y fundamental de este trabajo de tesis, que consiste en describir los procedimientos y algoritmos para la generación de términos clave dentro de los textos, así como la descripción de los diferentes módulos resultantes de la aplicación de dichos algoritmos.

- En el capítulo IV se describen los resultados y evaluaciones obtenidas de la aplicación del modelo propuesto, así como el corpus utilizado y la comparación con otros métodos.

- Por último se presentaran las conclusiones del trabajo de tesis y la bibliografía consultada para la realización del mismo.

CAPÍTULO II

MARCO TEÓRICO

En este capítulo se presenta la teoría en la cual se basa el presente trabajo, con el objetivo de dar a conocer los distintos algoritmos y técnicas que se aplicaran para la detección automática de temas importante.

2.1 Pagerank

El algoritmo de Pagerank (PR) fue publicado por Larry Page y Sergey Brin en la web de la Universidad de Stanford en 1998 [Brin and Page1998], como una fórmula que calculaba el ranking de una página web para el buscador Google. PR se define como una escala, la cual es un valor numérico que representa que tan importante es una página web.

El modelo predecesor de este algoritmo es el *Science Citation Index* (SCI) elaborado por Eugene Garfield para el Instituto de información científica (ISI) durante la década de 1950 [Garfield1983]. El Science Citation Index pretende resolver la asignación objetiva de méritos científicos suponiendo que los investigadores cuyo factor de impacto (número de publicaciones y/o referencias bibliográficas en otros trabajos científicos) es más alto, colaboran en mayor medida en su campo.

A diferencia de SCI, el algoritmo de Pagerank no le interesa ni el contenido o el tamaño de la página web, sólo considera los link de entrada y salida de cada página, con el fin de construir un grafo, en el cual cada vértice (nodo) es una página web y cada enlace representa los links de entrada o salida para cada página (Figura 2.1).

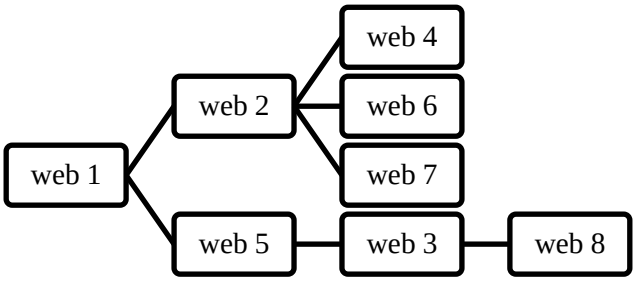


Figura 2.1 Representación Pagerank.

Google expone que cuando una página tiene un enlace a otra, está realizando un voto sobre ésta última. A más votos, más importante es una página.

El algoritmo interpreta un vínculo desde la página A hacia la página B como un voto de la página A por la página B. Pero Google revisa otras cosas aparte del número de votos o de vínculos que una página recibe, puesto que también analiza la página que emite el voto. Los votos emitidos por páginas que son en sí mismas "importantes" pesan más y ayudan a convertir a otras páginas también en "importantes"[Langville et al.2006].

El algoritmo propuesto por Brin y Page descrito de manera formal, se puede definir de la siguiente manera en la ecuación 1:

$$PR(V_i) = (1-d) + d\left(\sum_{j \in In(V_i)} \frac{1}{|Out(V_j)|} * PR(V_j)\right) \qquad (1)$$

Denotamos por:

- $In(V_i)$: es el conjunto de links entrantes de la pagina web V_i
- $Out(V_j)$: Es el conjunto de links salientes de V_j.
- $PR(V_j)$: Es el Pagerank de V_j.
- d es el factor de amortiguamiento que puede ser cualquier valor entre 0 y 1, algunos expertos aseguran que el valor de la variable suele ser 0.85, ya que representa la probabilidad de que un usuario continúe pulsando links al navegar por internet en vez de escribir una url directamente en la barra de direcciones. Por lo tanto, la probabilidad de que el usuario deje de pulsar links y navegar directamente a otra web aleatoria es de $(1-d)$ [Page1998].

La introducción del factor de amortiguamiento en la ecuación 1, resta algo de peso a todas las páginas, esto hace que las paginas que no tienen enlaces a ningún otra página, no salgan beneficiadas, lo que hará será navegar a cualquier otra página aleatoriamente, lo que equivaldría a suponer que una página sin ningún enlace saliente tiene enlaces a todas las páginas.

Las características del Pagerank así como su cálculo puede explicarse con el siguiente ejemplo (vea Figura 2.2):

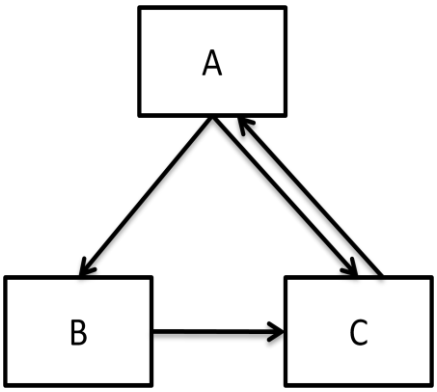


Figura 2.2 Ejemplo de Pagerank

Tenemos un sitio con tres páginas, A, B y C, donde A enlaza B y C, B enlaza C, y C enlaza A. De acuerdo con el algoritmo, el factor aleatorio d es de 0.85. Este factor d sabemos que afecta al Pagerank, pero no afecta a los principios del Pagerank.

Así que hagamos algunos cálculos:

PR(A) = 0.15 + 0.85 PR(C)*

*PR (B) = 0.15 + 0.85 *(PR(A) / 2)*

*PR(C) = 0.15 + 0.85 *(PR(A) / 2 + PR (B))*

Resolviendo las ecuaciones nos queda el siguiente Pagerank para cada página:

PR(A) = 1.07692305

PR(B) = 0.76923076

PR(C) = 1.15384615

Debido al tamaño actual de la Web, el buscador de Google usa un valor iterativo aproximado de Pagerank. Esto significa que a cada página se le asigna un valor inicial de Pagerank, y después el Pagerank de todas las páginas se calcula mediante distintas iteraciones en base al algoritmo. Este cálculo iterativo puede representarse en la Tabla 2.1 para nuestro ejemplo de la Figura 2.2 con un valor inicial de Pagerank para cada página de 1.

Iteración	P(A)	P(B)	P(C)
0	1	1	1
1	1	0.75	1.125
2	1.0625	0.765625	1.1484375
3	1.07421875	0.76855469	1.15283203
4	1.07641602	0.76910400	1.15365601
5	1.07682800	0.76920700	1.15381050
6	1.07690525	0.76922631	1.15383947
7	1.07691973	0.76922993	1.15384490
8	1.07692245	0.76923061	1.15384592
9	1.07692296	0.76923074	1.15384611
10	1.07692305	0.76923076	1.15384615

Tabla 2.1 Iteraciones del ejemplo de Pagerank.

Como se puede observar el Pagerank de una página se define de forma iterativa por el Pagerank de las páginas que enlazan a dicha pagina, el número mínimo de iteraciones necesario para que le Pagerank sea valido es de 100, pero algunos expertos sugieren que entre unas 40 y 50 iteraciones son suficientes para alcanzar un punto en que una iteración más no produciría suficiente cambio apreciable.

2.2 Secuencias Frecuentes Maximales

Una secuencia frecuente maximal (MFS por sus siglas en ingles) es una secuencia de palabras que debe aparecer en un número dentro de un documento y además, no debe estar contenida en otra secuencia de palabras.

A continuación, describiremos los conceptos básicos de las Secuencias Frecuentes Maximales, para esto denotaremos como S a un conjunto de documentos y cada documento consiste de una secuencia de palabras [Ahonen H.2002].

Definición 1. Si una secuencia *p* es una subsecuencia de *q* y el número de elementos de *p* es igual a *n*, entonces *p* es llamado *n-grama* en *q*.

Definición 2. Una secuencia $p = a_1 \ldots a_k$ es una subsecuencia de una secuencia *q* si todos los elementos a_i, $1 \leq i \leq k$, ocurren en *q* y además, ocurren en el mismo orden que en *p*. Si una secuencia *p* es una subsecuencia de una secuencia *q*, entonces se dice que *p* ocurre en *q*.

Definición 3. Una secuencia *p* es frecuente en *S* si *p* es una subsecuencia de al menos β documentos de *S*, donde β es un umbral de frecuencia predefinido. Sólo se cuenta una

ocurrencia de la secuencia en el documento. Varias ocurrencias dentro de un documento no hacen que una secuencia sea más frecuente.

Definición 4. Una secuencia *p* es una secuencia frecuente maximal en *S* si no existe alguna otra secuencia *q* en *S* de tal manera que *p* es una subsecuencia de *q* y *p* es frecuente en *S*.

El objetivo principal de este método es encontrar todas las subsecuencias en un conjunto de secuencias dentro de un texto, la descripción del esquema de este método se presenta en el Algoritmo 1.

En la fase inicial (Algoritmo 1) se extraen todos los pares ordenados de palabras o bigramas de palabras (A, B), tal que A y B acurren dentro del documento en este orden, y este bigrama es frecuente dentro del mismo, una vez obtenido estos bigramas se buscan secuencias más largas que se calculan a partir de ocurrencias de pares, todas estas ocurrencias obtenidas son almacenadas, es decir, el cálculo de ABC puede ayudar a calcular posteriormente la frecuencia ABCD. Lo que se intenta hacer es expandir un *n*-grama sólo sino no es una subsecuencia de alguna frecuencia máxima descubierta anteriormente, lo que garantiza que la secuencia máxima no se repita varias ocasiones.

Cuando todos los *n*-gramas se han descubierto, cada uno de estos es llamada secuencia de máximos, si algún *n*-grama dentro del proceso no se pudo ampliar, este es denominado secuencia máxima. Dentro del modulo *expand* del algoritmo 1.2.1, se analiza todas las posibles expansiones que puede tener un *n*-grama, es decir, que en cualquier momento se puede añadir un nuevo elemento a las secuencia de máximos. Por otro lado, si un *n*-grama existe dentro del conjunto de secuencia de máximo, este automáticamente se elimina el *n*-grama.

Posteriormente, una vez realizado esta selección se unen todas las secuencias máximas, en un nuevo conjunto y vuelve a iniciar con *n+1*-gramas, comparándolos con el nuevo conjunto de secuencias máximas.

Input: *S:* a set of documents, *σ:* a frequency threshold
Output: *Max:* the set of maximal frequent sequences

for all the documents d ∈ S
collect all the ordered pairs within *d*
end;
G_2 = all the ordered pairs that are frequent in *S*
k: = 2
Max: = σ;
while G_k is not empty **do**
for all grams $g \in G_k$ **do**
if *g* is not a subsequence of some $m \in Max$
if g is frequent **then**
max: = **expand** (g)
max: = Max ∪ max
if *max = g* **then**
remove g from G_k
else
remove *g* from G_k
Prune (G_k)
G_{k+1}:= Join (G_k)
k: = k + 1;
end;
end;
return *Max;*

Algoritmo 1: Extracción de Secuencias Frecuentes Maximales

2.3 Pre-procesamiento

Previo a aplicar los métodos para la extracción de términos clave descritos, es necesario realizar un pre-procesamiento a cada texto del corpus por analizar. La omisión de ciertos elementos del texto, es un paso relevante ya que puede confundir al sistema al considéralos importantes.

2.3.1 Eliminación de palabras cerradas

En el lenguaje natural hay palabras que tienen muchos significados (como los sustantivos y los adjetivos) y otras que no tanto (como los artículos o las conjunciones). Estas últimas se conocen como palabras cerradas o vacías.

Las palabras cerradas son palabras sin significado, como los artículos, los pronombres, las preposiciones, las conjunciones o los adverbios, que se pueden eliminar antes o después del procesamiento del texto.

Las palabras cerradas son muy frecuentes pero aportan poco valor de contenido semántico al texto y poca ayuda a la recuperación de información, se conocen como lista de stopword [Luhn P.1958].

Una lista de palabras vacías (stopwords) es un listado de términos (preposiciones, determinantes, pronombres, etc.) considerados de escaso valor semántico, que cuando se identifican en un documento se eliminan, sin considerarse términos índices para la colección de textos a analizar. La supresión de todos estos términos evita los problemas de ruido documental y supone un considerable ahorro de recursos, ya que aunque se trata de un número relativamente reducido de elementos tienen una elevada tasa de frecuencia en los documentos.

Hay sistemas en los que la lista de palabras vacías:

a) Viene Predeterminada: el sistema dispone, ya desde el principio, de la lista de palabras vacías de su idioma o idiomas. De hecho, su elaboración es fácil, ya que sólo hay que añadir las categorías vacías de una base de datos de terminología al idioma que se quiera. Los artículos siempre son los mismos, las conjunciones también, incluso los verbos se pueden llegar a contabilizar y flexionar en todos los tiempos verbales.

b) Se evita expresamente para permitir al sistema la búsqueda por frases. Los sistemas que los evitan disponen de otras herramientas para reducir significativamente el número de palabras indizadas, como técnicas de Stemming o lematización que veremos más adelante.

2.3.2 Lematización

La lematización de términos es una parte del procesamiento lingüístico que trata de determinar el lema de cada palabra que aparece en un texto. Su objetivo es reducir una palabra a su raíz, de modo que las palabras clave de una consulta o documento se representen por sus raíces en lugar de sus palabras originales. El lema de una palabra comprende su forma básica más sus formas declinadas. Por ejemplo, "informa" podría ser el lema de "información", "informaciones", e "informar".

El proceso de lematización se lleva a cabo utilizando algoritmos de truncamiento (o Stemming), que permiten representar de un mismo modo las distintas variantes de un término, a la vez que reducen el tamaño del vocabulario y mejoran, en consecuencia, la capacidad de almacenamiento de los sistemas y el tiempo de procesamiento de los documentos. No obstante, estos algoritmos presentan el inconveniente de no agrupar en ocasiones palabras que deberían estarlo, y viceversa, mostrar como iguales palabras que realmente son distintas [Strzalkowski T.1999].

Hoy en día existe software para realizar la lematización de textos, los cuales realizan un etiquetamiento de estos, también se conoce el lema de cada una de las palabras contenidas en los textos, es decir la raíz o la forma del diccionario de todas las palabras de un texto.

Las reglas para etiquetar un texto son relativas al etiquetador utilizado, pero generalmente, la estimación de que una palabra tenga una etiqueta gramatical se hace en relación con el contexto. A partir de una serie de tres etiquetas conocidas que constituyen el conjunto de aprendizaje, los etiquetadores utilizan arboles binarios construidos para saber cuál es la clase más probable para una palabra como se ve en la Figura 2.3

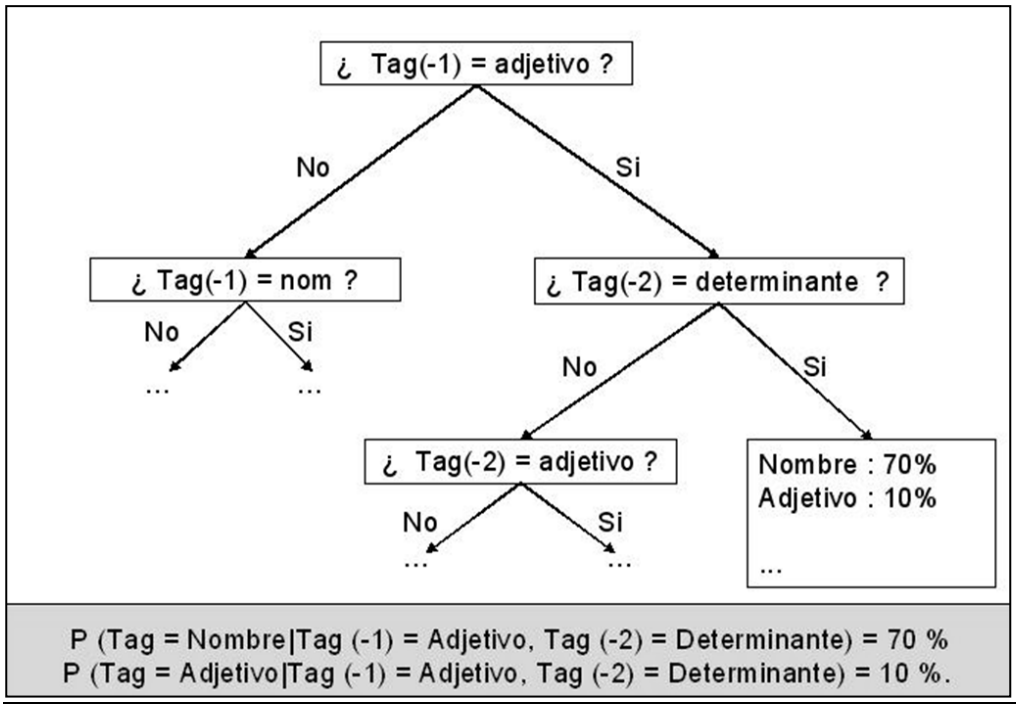


Figura 2.3 Etiquetamiento dado por un lematizador

Las reglas para etiquetar las palabras pueden ser establecidas según el entrenamiento del etiquetador sobre un corpus de etiquetado a la mano. Es el caso de Winbrill que está entrenado a partir del Wall Street Journal. Pero, en este caso el léxico puede no estar adaptado para textos especializados.

Cada etiquetador tiene archivos de parámetros que utilizará para etiquetar mejor un texto. Aquellos archivos de parámetros están generalmente compuestos de archivos léxicos, reglas léxicas, reglas del contexto o excepciones en el idioma.

Una vez que el texto esta etiquetado, se hace la lematización. Entonces, se necesita encontrar una buena clase para cada palabra, sino cada palabra podría tener varios lemas. Entre los lematizadores automáticos se pueden mencionar los siguientes:

- Tree Tagger
- Winbril
- Tnt Tagger
- Mbt Tagger

2.3.3 Truncamiento

El truncamiento o Stemming es un método que tiene por objetivo reducir una palabra a su raíz o stem. Este proceso se emplea dentro de muchas técnicas del Procesamiento de Lenguaje Natural debido a que nos ayuda a reducir varias palabras a una sola raíz.

Existen distintos algoritmos de truncamiento de los cuales se pueden mencionar el de Porter [Paice C. and Jones P.1993] y Lovins [Lovins J.B.1968], los cuales son los dos algoritmos más populares para el truncamiento de palabras en el idioma ingles. Ambos algoritmos usan reglas heurísticas para remover o transformar sufijos ingleses.

Otra aproximación al truncamiento es usar un diccionario que explícitamente liste las raíces por cada palabra que puede ser encontrada en un texto dado. Las heurísticas son usualmente preferidas para un diccionario. El truncador de Lovins es más agresivo que el truncador de Porter, es decir, en el truncador de Lovins es más probable encontrar dos palabras con la misma raíz, pero este es más probable que cometa errores [Krovetz R.1993].

Por ejemplo, el truncador de Lovis mapea correctamente las palabras *"psychology"* y "psychologist" a la misma raíz *"psycholog"*, pero el truncador de Porter mapea incorrectamente hacia diferentes raíces como *"psychologi"* y *"psychologist"*. Por el otro lado, el truncador de Porter mapea correctamente *"pólice"* and *"policy"* a diferentes raíces, *"polic" "polici"*, pero el de Lovins mapea incorrectamente para la misma raíz *"polic"*.

En algunos trabajos, se ha observado que el truncado agresivo es mejor para la extracción de frases claves que el truncado conservativo. En estos experimentos se ha usado un algoritmo de truncado agresivo al que han llamado truncador iterado de Lovis. El algoritmo repetitivamente emplea el truncador de Lovis, hasta que la palabra termina su cambio.

Por ejemplo, si la palabra dada es "incredible" como entrada, el truncador de Lovis generara "incred" como salida. Ahora la palabra "incred" se maneja como entrada, este generara "incr" como salida, luego la palabra "incr" es entrada, la salida también será

"incr". Por esto, para el algoritmo iterado de Lovis, dando "incredible" como entrada genera "incr" como salida. Iterando de esta manera se necesitará incrementar (o dejar sin cambios) la agresividad de cualquier raíz.

2.4 Métricas de evaluación

Es posible ver a la extracción de frase claves como un problema de clasificación. Si se imagina un documento como un conjunto de palabras y frases, entonces la tarea consiste en clasificar cada palabra o frase en una de sus categorías: sea o no sea frase clave. Se puede evaluar un algoritmo de extracción de palabras claves por el grado en que sus clasificaciones corresponden a las hechas por un experto.

La tarea de aplicar un algoritmo de extracción de palabras frases a un corpus se detalla en una matriz de clasificación como se muestra en la tabla 2.2.

	Clasificación de términos clave dados por un experto.	Clasificación de términos no clave dados por un experto.
Clasificación de términos clave dados por una aplicación (software).	*a*	*b*
Clasificación de términos no clave dados por una aplicacion (software).	*c*	*d*

Tabla 2.2 matriz de clasificación

- La variable ***a*** representa el número de veces que un término clave generada por un experto empata con un término clave generada por una aplicación (software) de extracción de términos clave.

- La variable ***d*** representa el número de veces que el experto y la aplicación determinan que un término clave no es un término clave.

- Las variables ***b*** y ***c*** representan el número de veces que el experto y el software estuvieron en desacuerdo en la clasificación de las frases.

Investigaciones en recuperación de información usan precisión (*precision*) y evocación (*recall*) para evaluar el desempeño de los métodos de búsqueda y extracción de términos clave que a continuación se definen.

2.4.1 Precisión

La precisión se define como el porcentaje de respuestas correctas sobre todos los resultados dados por un sistema. Esta medida favorece a los sistemas que obtienen una alta fiabilidad en la asignación de sentidos (ver ecuación 2).

$$Precision = \frac{a}{a+b} \qquad (2)$$

Donde $a + b = total\ de\ resultados\ dados\ por\ un\ sistema$

Por ejemplo:

Se tiene un total de 75 elementos dados por un sistema a evaluar, pero de estos elementos solo 40 son correctos, por lo tanto:

$$presicion = \frac{40}{75} = 0.533$$

2.4.2 Recall

El recall es la métrica básica para decidir la calidad de un sistema porque muestrea el número de casos correctos sobre todos los casos dados por un experto (ver ecuación 3).

$$Recall = \frac{a}{a+c} \qquad (3)$$

Donde $a + c = total\ de\ casos\ del\ experto$

Por ejemplo:

Se tiene un total de 100 elementos dados por un experto, de los elementos dados por el sistema sólo 40 son correctos ya que se encuentran dentro del conjunto del experto por lo tanto:

$$Recall = \frac{40}{100} = 0.4$$

2.4.3 F-Measure

Las métricas de evaluación son realmente informativas si se interpretan conjuntamente. Un sistema puede tener una precisión muy buena, pero un recall muy bajo, lo que significa que ha identificado la respuesta correctamente para muy pocos casos, pero con alta fiabilidad. En general hay un equilibrio entre las dos medidas y su combinación se puede expresar mediante la métrica F *(F-measure*) según la ecuación 4:

$$F - mesure = 2 * \frac{presicion * recall}{presicion + recall} \quad (4)$$

Tomando los ejemplos de recall y precisión vistos anteriormente tendríamos:

$$F - mesure = 2 * \frac{0.533 * 0.4}{0.533 + .04} = 2 * \frac{0.2132}{0.933} = 0.45702036$$

Cuando la precisión y el recall son muy cercanas, el F-measure está cerca del promedio entre la precisión y el recall. Cuando la precisión y el recall no están balanceadas, entonces el F-measure es menor que el promedio.

2.4.4 Modelo de evaluación usado en SemEval-2

SemEval-2 es una competencia libre con el propósito de evaluar sistemas enfocados al análisis semántico dentro del PLN, mediante un conjunto de tareas, con el propósito de explorar los aspectos científicos y técnicos, en el caso del trabajo de esta tesis nos enfocaremos a la tarea #5 de Semeval-2 2010: "*Automatic Keyphrase Extraction from Scientific Articles*" [SEMEVAL-2].

A grandes rasgos, esta tarea consiste en extraer los términos clave de un corpus compuesto de artículos científicos y compara los resultados con un gold-standard formado de los términos dados por un lector y un conjunto combinado de términos del autor y el lector.

En esta sección se describe un script hecho en Perl llamado *performance.pl*, este script tiene como finalidad evaluar los términos dados por el método propuesto en esta tesis para ver la efectividad que tienen estos términos comparados con los del gold-standard.

Este script tiene como entrada cuatro archivos, uno dado por el sistema y los tres gold-standard, el archivo dado por el sistema del competidor debe contener 15 probables términos clave por cada uno de los artículos científicos.

Este script arroja como salida la precisión, recall y F-score de los primero 5, 10 y 15 probables términos dados por el competidor.

La forma en cómo trabaja este script para realizar estas operaciones se describe a continuación:

- Lee el archivo que contiene los términos dados por el método propuesto, para extraer el nombre del artículo y sus términos clave y los guarda en una tabla hash.

```
my ($papername, $keystr) = split(/ : /, $line);
```

- Posteriormente lee los archivos del gold-standard mediante el método *MeasurePerformance*.

```
MeasurePerformance("reader.stem.final");
MeasurePerformance("combined.stem.final");
```

  En el método *MeasurePerformance* lee línea por línea los gold-standard (cada línea contiene el nombre del artículo y sus respectivos términos clave), de cada una extrae el nombre del articulo y sus términos clave y guarda estos datos, así como el numero de términos que contiene dicho artículo, también cada vez que lee los términos de un artículo, lleva un contador sobre los términos totales de contenidos en cada gold-standard.

- Una vez leídos cada unos de los términos clave correspondientes a cada artículo científico del gold-standard, compara cada unos de los términos con los obtenidos en el método propuesto dividiéndolos en rango de los primeros 5, 10 y 15, de los cuales por cada uno de estos calcula la precisión, recall y F-measure y así dar como resultado estos valores y así observar el porcentaje de efectividad, la forma en cómo realiza estos cálculos se pueden observar a continuación.

```
if($ind == 5)
{
    $Correct[0] += $correct;
    $localP = $correct/5;
    $localR = $correct/$localKeynum;
    if(($localP + $localR) > 0){
        $localF = 2* ($localP * $localR) / ($localP + $localR);}
   $Precision[0] += $localP;
    $Recall[0] += $localR;
```

```
    $Fscore[0] += $localF;
}
elsif($ind == 10)
{
      $Correct[1] += $correct;
      $localP = $correct/10;
      $localR = $correct/$localKeynum;
      if(($localP + $localR) > 0){
          $localF = 2* ($localP * $localR) / ($localP + $localR);}
      $Precision[1] += $localP;
      $Recall[1] += $localR;
      $Fscore[1] += $localF;
 }
elsif($ind == 15)
{
      $Correct[2] += $correct;
      $localP = $correct/15;
      $localR = $correct/$localKeynum;
      if(($localP + $localR) > 0){
          $localF = 2* ($localP * $localR) / ($localP + $localR);}
      $Precision[2] += $localP;
      $Recall[2] += $localR;
      $Fscore[2] += $localF;
}
```

- Por último normaliza las métricas totales entre en número total de artículos, para obtener un promedio de estos valores.

```
my $Prec5 = sprintf ("%.2f\%", ($Precision [0]/$paperNumber) *100);
my $Prec10 = sprintf ("%.2f\%", ($Precision [1]/$paperNumber) *100);
my $Prec15 = sprintf ("%.2f\%", ($Precision [2]/$paperNumber) *100);
my $Recall5 = sprintf ("%.2f\%", ($Recall [0]/$paperNumber)*100);
my $Recall10 = sprintf ("%.2f\%", ($Recall [1]/$paperNumber)*100);
my $Recall15 = sprintf ("%.2f\%", ($Recall [2]/$paperNumber)*100);
my $Fscore5 = sprintf("%.2f\%", ($Fscore[0]/$paperNumber)*100);
my $Fscore10 = sprintf("%.2f\%", ($Fscore[1]/$paperNumber)*100);
my $Fscore15 = sprintf("%.2f\%", ($Fscore[2]/$paperNumber)*100);
```

Así es como el script evalúa a los competidores dentro de la tarea #5, con estos resultados se puede observar que tan eficiente y confiable es el sistema propuesto.

2.5 Otros modelos usados en la extracción de palabras clave

Dentro de la extracción de palabras clave existen métodos de clasificación basados en cálculos estadísticos y probabilísticos, entre los que destacan Naïve Bayes y Máxima Entropía, los cuales toman varias propiedades o atributos de las palabras para realizar dicha clasificación.

2.5.1 Naïve Bayes

El método de clasificación de Naïve Bayes se utiliza cuando queremos clasificar una instancia descrita por un conjunto de atributos $(a_i's)$ en un conjunto finito de clases (V) [Langley P. and Thompson K.1992].

Para clasificar un nuevo ejemplo de acuerdo con el valor más probable dados los valores de sus atributos usamos la ecuación 5.

$$v_{MAP} = argmax_{vj \in V}(P(v_j|a_1, \dots, a_n)) \quad (5)$$

Para clasificarlo usando Bayes (ecuación 6):

$$v = argmax_{vj \in V}\left(\frac{P(a_1, \dots, a_n|P(v_j))}{p(a_1, \dots, a_n)}\right) \quad (6)$$

$$= argmax_{vj \in V}(P(a_1, \dots, a_n|v_j)P(v_j))$$

$P(v_j)$ Se puede estimar con la frecuencia de las clases, pero para $P(a_1, \dots, a_n|P(v_j)$ tenemos muy pocos elementos. El clasificador Naïve Bayes, asume que los valores de los atributos son condicionalmente independientes dado el valor de la clase.

Es decir:

$$P\left(a_1, \dots, a_n|P(v_j)\right) = \prod_i P(a_i|v_j) \quad (7)$$

Por lo que:

$$v_{NB} = argmax_{vj \in V}\left(P(v_j)\prod_i P(a_i|v_j)\right) \quad (8)$$

Los valores $P(a_i|v_j)$ se estiman con la frecuencia de los datos observados.

Por último cabe mencionar que no se hace búsqueda de hipótesis, simplemente se cuentan frecuencias de ocurrencias.

2.5.2 Máxima Entropía

En 1948 Claude Shannon [Shannon1948] desarrolló un esquema matemático que permite cuantificar la información contenida en una distribución de probabilidades $\{p_j; j = 1,..., M\}$ definida sobre un conjunto de *M* sucesos complementarios. Definimos la entropía de Shannon como (vea ecuación 9):

Sea *X* una variable aleatoria discreta, con valores $x_1,..., x_n$, y con distribución de probabilidad p_i: $P(X = x_i)$. Sea $p = (p_1,\ldots, p_n)$.

$$H(p) \coloneqq -\sum_{i=1}^{n} p_i \ln p_i \qquad (9)$$

Entonces *H(p)* mide la cantidad de información que esperamos obtener al conocer el valor de *X*. Equivalentemente, también mide la reducción en la incertidumbre que esperamos obtener en la siguiente realización de *X*.

Esta función es, por lo tanto, una medida de lo que el observador "ignora" antes de conocer el resultado de un experimento. En 1957 Jaynes [Jaynes1979] adoptó esta definición de Shannon para operar matemáticamente con el concepto de "ignorancia" y propuso un método de inferencia, conocido como Principio de Máxima Entropía, por medio del cual dio una prescripción para seleccionar una distribución de probabilidades en base al menor número de hipótesis.

El principio o método de Máxima Entropía (MaxEnt) es un procedimiento para generar distribuciones de probabilidad de forma sistemática y objetiva.

Se puede describir de la siguiente manera: de entre todas aquellas distribuciones de probabilidad compatibles con cierta clase de información, escoger la que conlleve una mayor incertidumbre.

MaxEnt es no-especulativo, en el sentido de que únicamente utiliza la información contenida en los datos y el espacio muestral. Por esta razón, el método de máxima entropía es apropiado cuando tenemos muy poca información más allá de los datos mismos.

Jaynes sugiere que en las fases iniciales de un problema, cuando tenemos poca información previa y en particular no tenemos un modelo, como dijimos antes, es conveniente usar un método como MaxEnt. Sin embargo, una vez que se tenga más y más

información disponible y se puedan definir claramente los modelos apropiados, entonces el problema será mejor analizado usando las técnicas bayesianas.

Veamos ahora como se realizan los cálculos que nos permiten encontrar la distribución de máxima entropía.

Sea *X* una variable discreta con rango finito $x_1,\ldots,x_n$. Para estimar la función de probabilidad de $p_i = P(X = x_i)$ que maximice la entropía, usamos los multiplicadores de Lagrange.

Tenemos que maximizar la medida de Shannon (ecuación 10):

$$H = -\sum_{i=1}^{n} \ln p_i \qquad (10)$$

Sujeta a

$$\sum_{i=1}^{n} \ln p_i$$

Y

$$\sum_{i=1}^{n} p_i g_r(x_i) = a_r \qquad r = 1, \ldots, m;\ p_i \geq 0\ \forall_i$$

Donde la función g_r calcula la esperanza de $x_1,\ldots,x_n$

El lagrangiano está dado por

$$L = -\sum_{i=1}^{n} p_i \ln p_i - (\lambda_0 - 1)\left(\sum_{i=1}^{n} p_i - 1\right) - \sum_{r=1}^{m} \lambda_r \left(\sum_{i=1}^{n} p_i\, g_{ri} - a_r\right)$$

Donde $\lambda_0,\ldots,\lambda_m$ son los m+1 multiplicadores de Lagrange correspondientes a las m+1 restricciones.

Se tiene que

$$\frac{\partial L}{\partial p_i} = 0 \qquad \Longrightarrow \qquad -\ln p_i - \lambda_0 - \sum_{r=1}^{m} \lambda_r g_{ri} = 0$$

ó

$$p_i \;=\; \exp\left(-\lambda_0 - \lambda_1 g_{1i} - \cdots - \lambda_m g_{mi}\right)$$

$$i = 1,2,\ldots,n$$

Los multiplicadores quedan determinados al sustituir los P_i de tal forma que

$$\sum_{i=1}^{n} \exp\left(-\lambda_0 - \sum_{j=1}^{m} \lambda_i g_{ji}\right) = 1$$

y

$$\sum_{i=1}^{n} g_{ri} \exp\left(-\lambda_0 - \sum_{j=1}^{m} \lambda_i g_{ji}\right) = a_r$$

Notemos que tanto λ_0 como las restricciones a_r se pueden poner en términos de los λ_i:

$$\exp(\lambda_0) = \sum_{i=1}^{n} \exp\left(-\sum \lambda_j g_{ji}\right)$$

y

$$a_r = \frac{\sum_{i=1}^{n} g_{ri} \exp\left(-\sum_{j=1}^{m} \lambda_j g_{ji}\right)}{\sum_{i=1}^{n} \exp\left(-\sum_{j=1}^{m} \lambda_j g_{ji}\right)}$$

Para $r=1,2,\ldots, m$.

Se pueden probar varias propiedades de los multiplicadores

- λ_0 es una función convexa de $\lambda_1,\ldots,\lambda_m$.
- $H_{max} = \lambda_0 + a_1\lambda_1 + \ldots + a_m\lambda_m$
- H_{max} es una función cóncava de $a_1,\ldots,a_m$.

CAPÍTULO III

MÉTODO PROPUESTO BASADO EN SECUENCIAS FRECUENTES MAXIMALES Y PAGERANK

3.1 Algoritmo propuesto para la extracción de términos clave

El enfoque principal presentado en este trabajo recae en la combinación de dos diferentes técnicas para la selección y obtención de los términos clave de un determinado texto: MFS y Pagerank. En la Figura 3.1 se puede observar un bosquejo de cómo se hará uso estos dos métodos para la extracción de términos clave.

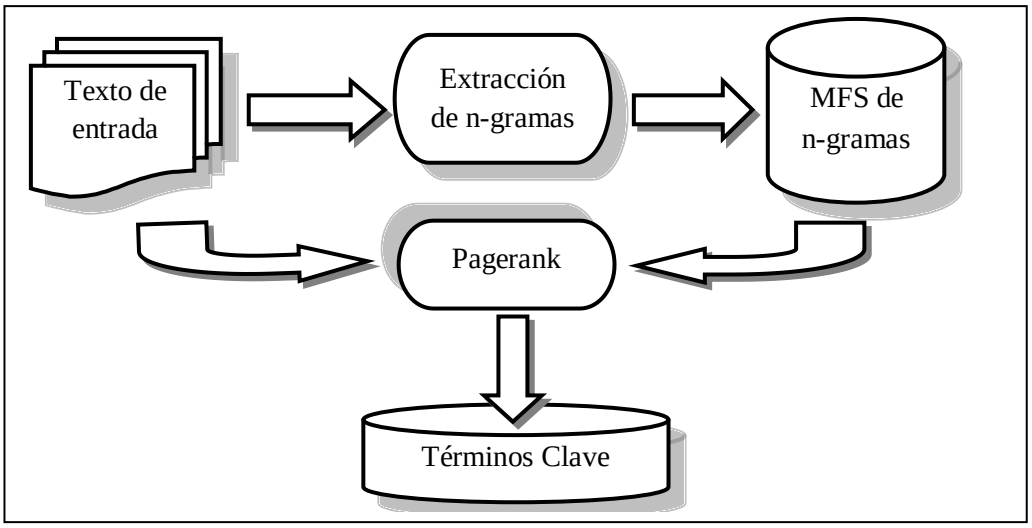


Figura 3.1 Enfoque principal del Proyecto

Como se observa en la Figura 3.1, se realiza la extracción del conjunto de *n*-gramas, en este enfoque se hace uso de ellos debido a que es una parte fundamental para el MFS para producir los posibles términos clave y, a partir de estos, se obtiene un ranking, con el fin de determinar cuáles son los términos más importante (de acuerdo con el algoritmo Pagerank).

Una vez hecho este análisis acerca de la forma en que se va a procesar los textos para la obtención de términos clave, se genera un algoritmo el cual asocia cada uno de los puntos representados en la Figura 3.1. La descripción completa del algoritmo que representa el enfoque del proyecto, se da en el Algoritmo 2.

Input: A set *D* of documents:
$D = \{d_1, d_2, \ldots\}$
Output: *A* set $K = \{K_1, K_2, \ldots\}$ of keyphrases for each document d_i:
$K_i = \{k_{i,1}, k_{i,2}, \ldots\}$
foreach $d_i \in D$ **do**
AS = ***Extract_Acronyms***(d_i);
d_i^1 = ***Pre_Processing***(d_i);
MFS = ***Maximal_Freq_Sequences***(d_i^1);
CK = ***PageRanking***(d_i^1, MFS);
CU = Top_Ten_Unigrams(*CK*);
CB = Top_Ten_Bigrams(*CK*);
CT = Top_Three_Trigrams(*CK*);
$K_i = \theta$;
AcroCounter = 0;
NB =0;
foreach *unigram* ∈ *CU* **do**
if *AcroCounter* < 3 **then**
if *unigram* ∈ *AcronymSet* **then**
$K_i = K_i \cup \{unigram\}$;
EA = *Expand(AcronymSet);*
$K_i = K_i \in \{EA\}$;
AcroCounter++;
end
end
end
NU = AcroCounter-3;
foreach *unigram* ∈ *CU* **do**
if *NU* >= 3 **then**
if *unigram* ∉ *AcronymSet* **then**
$K_i = K_i \cup \{unigram\}$;
NU++;
end
end
end
foreach *trigram* ∈ *CT* **do**
$K_i = K_i \cup \{trigram\}$;
end
while *NB < (15 − (2 * AcroCounter + 3 + NU))* **do**
$K_i = K_i \cup \{bigram\}$;
NB++;
end
end
return $K = \{K_1, K_2, \ldots\}$

Algoritmo 2: Algoritmo del método de extracción de términos clave propuesto.

Dentro del Algoritmo 3.1.1 existen los siguientes módulos:

- *Extract_Acronyms.*
- *Pre_Processing*
- *Maximal_Freq_Sequences*
- *PageRanking*

Estos módulos se describen en forma detallada en secciones posteriores.

3.2 Modulo Extract_Acronyms

Un acrónimo es una palabra que resulta de la unión de las letras iniciales de una o más palabras. Dentro de la obtención de términos claves se observó que el uso de acrónimos, es una parte fundamental ya que en la mayoría de los casos el autor usualmente introduce acrónimos dentro de sus textos.

Dentro de la obtención de términos claves se observó que el uso de acrónimos, es una parte fundamental ya que en la mayoría de los casos el autor usualmente introduce acrónimos dentro de sus textos, por lo cual dentro del Algoritmo 3.1.1 se incluye un módulo llamado *Extract_Acronyms,* el cual se basa en un algoritmo que, mediante el uso de un procesamiento de búsqueda, obtiene tanto el acrónimo, como su versión extendida (frase completa). La descripción completa del algoritmo para la identificación y obtención de acrónimos se describe a en el Algoritmo 2.

Input: Un texto *D* y un conjunto de palabras *p* donde
D= {p1, p2,....};
Output: Un conjunto *A= {A1, A2,...}* donde
Aj= {acrónimo, versión extendida};

foreach $p_i \in D$ **do**
if inicia '(' **and** termina')' $\in p_i$ **then**
delete [(,)] **in** p_i;
l = ***length*** p_i;
n_acro=l;
n_frase= i-1;
for *aux = 1* **to** *l* **do**
if $p_{(n\_frase),1} \in p_{i,n\_acro}$ **then**
version_extendida = version_extendida $\cup$ p_{n_frase};
n_acro--;
n_frase--;
end
else then
n_acro--;
end
end
A_j = {p_i , *version_extendida};*
end
end

Algoritmo 3: Algoritmo para el procesamiento de acrónimos.

3.3 Modulo Pre_Processing

Antes de aplicar los métodos propuestos para la extracción de términos claves, es necesario realizar un pre-procesamiento a cada uno de los textos, de esta forma eliminar la información que podría generar que el sistema de cómo salida términos irrelevantes.

La realización del pre-procesamiento de textos se realizó de la siguiente forma:

- Conversión a minúsculas de todas las palabras contenidas en el texto.
- Lematización y Stemming de los textos usando Treetagger y Porter Stemming respectivamente.
- Eliminación de stopwords usando un diccionario de stopwords.
- Eliminación de signos de puntuación, así como caracteres especiales.

La Figura 3.2, muestra de forma gráfica cada uno de los pasos del pre-procesamiento.

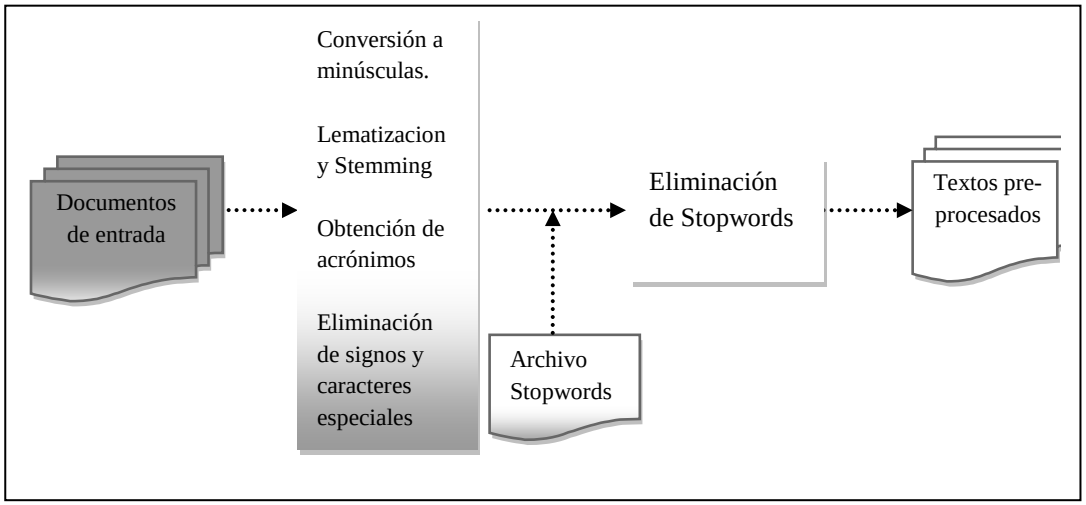


Figura 3.2 Pre-procesamiento de texto.

A continuación se dará una breve explicación de cómo se realizo cada uno de los pasos dentro del pre-procesamiento.

3.3.1 Lematización

La Lematización se define como el proceso mediante el cual las palabras de un texto que pertenecen a un mismo modelo derivativo son llevadas a una forma normal que representa a toda la clase.

Una de sus variaciones consiste en encontrar el lexema de las palabras analizadas, es decir, cambiar las palabras analizadas en un término conceptual (lexema) y morfológicos (morfemas), de esta manera se obtiene el lema que corresponde a dichas palabras.

Dentro de las formas morfológicas que se pueden encontrar para la lematización podemos mencionar:

- ***Morfema de género:*** para indicar si la palabra está en masculino o femenino. Ejemplo: león, leon-a;

- ***Morfema de número:*** indica si la palabra está en plural. Ejemplo: león, leon-es

- ***Desinencias:*** son morfemas que se añaden al lexema de los verbos para indicarnos la persona, el número, el tiempo y el modo. Ejemplo: Ten-emos

Para la realización del proceso de Lematización para este trabajo se uso el paquete Treetagger, el cual es un programa de etiquetado morfológico, además de las características morfológicas, proporciona el lema de la palabra, como se muestra en el siguiente ejemplo de la Tabla 3.1, donde:

VBZ = Verbo, presente simple, 3ª persona del singular.
VB = Verbo infinitivo
JJ = Adjetivo
NP = sustantivo propio
DT = Articulo definido

Word	pos	lemma
The	DT	the
Treetagger	NP	Treetagger
is	VBZ	be
easy	JJ	easy
use	VB	use

Tabla 3.1: Salida usando Treetagger

Esta característica fue lo que impulso el uso de este programa para efectuar esta función, ya que así sólo tomamos el lema de la palabra el cual es sustituido dentro del texto original.

El proceso de instalación del Treetagger se describe a continuación:

- Descargar los archivos tree-tagger-linux-3.2.tar.gz, english-par-linux-3.1.bin.gz, e install-tagger.sh en el mismo directorio.

Para instalarlo, ejecutamos el siguiente comando:

- sh install-tagger.sh

Una vez instalado, para poder lematizar un texto, ejecutamos el siguiente comando.

- cmd/tree-tagger-english *In_file* > *Out_file*.

3.3.2 Stemming

Stemming es un método para reducir una palabra a su raíz o stem. En lo particular, este método se aplicó dentro del pre-procesamiento, con el fin de reducir las palabras que derivan de una misma raíz, y así obtener una generalización de esta, además de evitar que varias palabras descendientes de la misma raíz se encuentren dentro de los términos clave finales.

Por ejemplo se tiene las siguientes palabras: *representative*, *representation*, *represented*, *represented;* su stem que representa estas cuatro palabras es *represent*, lo que nos demuestra que al aplicar este método a los textos, nos genera una mayor precisión acerca de los temas e ideas principales del texto, ya que al truncar las palabras, nos reduce el número de términos usados y aumenta la frecuencia de esta.

Existen varios algoritmos de Stemming que ayudan en sistemas de recuperación de información.

El algoritmo más común para Stemming es el algoritmo de Porter, el cual se aplicó en este trabajo, este algoritmo se basa en extraer los sufijos y prefijos comunes de palabras literalmente diferentes pero con una raíz común que pueden ser consideradas como un sólo término.

3.3.3 Stopwords

Un texto está conformado por una secuencia lo más coherente posible de palabras que caracteriza y definen su estructura. Cuando se analiza el contenido se puede observar que existen diversos tipos de palabras como lo son adjetivos, artículos, preposiciones, adverbios, etc. Las cuales desempeñan un papel muy importante para darle coherencia al texto, pero en nuestro caso, estas palabras carecen de importancia y es por esta razón que se eliminan.Debido a que estas palabras aparecen con mucha frecuencia (pronombres, adjetivos, artículos), cabe la posibilidad de que sean tomadas como términos claves, lo cual no nos daría el resultado esperado.

La técnica para la eliminación de estas palabras se basa principalmente en contar con una lista, la cual contiene todas las posibles palabras (Stopwords), una vez obtenidas estas palabras, se comparan con cada término del texto, y sin son iguales, se eliminan del documento.

3.4 Módulos Secuencia Frecuente Maximal y Pagerank

Una vez que el texto esta pre-procesado, es decir, que sólo contiene términos que son realmente importantes, se extraen un conjunto de *n*-gramas. Un *n*-grama lo podemos definir como una subsecuencia de *n* elementos (palabras), este modelo es muy empleado en varias áreas del PLN.

En este caso se trabajaron con unigramas, bigramas y trigramas los cuales están descritos de la siguiente manera:

Sea *p* una palabra dentro del documento *S;*

Conjunto de 1-grama= {p_1, p_2, p_3, p_4*.....}*
Conjunto de 2-gramas= {$p_1 \cup p_2$, $p_2 \cup p_3$, $p_3 \cup p_4$*.....}*
Conjunto de 3-gramas= {$p_1 \cup p_2 \cup p_3$, $p_2 \cup p_3 \cup p_4$, $p_3 \cup p_4 \cup p_5$*.....}*

Los conjuntos obtenidos son procesados por el modulo ***Maximal_Freq_Sequences***, el cual está basado en el Algoritmo 1 para la extracción de *n*-gramas usando MFS, lo cual nos da como salida un conjunto de gramas, con los cuales alimentaremos al modulo PageRank para obtener el ranking final de cada uno de estos *n*-gramas.

El algoritmo Pagerank se muestra en el Algoritmo 4, el cual tiene como entrada un conjunto de *n*-gramas (*n*=1…3 en particular para este proceso) y da como salida un conjunto ordenado con respecto a su ranking (PageRank) el cual se obtiene aplicando la ecuación 1.

Posteriormente se crea el grafo, en el cual cada vértice (nodo) es un *n*-grama$_i$ y cada enlace representa la unión de un *n*-grama$_{i+1}$ o *n*-grama$_{i-1}$ con respecto al orden dentro del texto.

Como se explicó en el Capítulo I, este algoritmo necesita ser iterativo para obtener el ranking más preciso, así que se usan 20 iteraciones, donde cada vez que culmina una iteración con sus respectivos cálculos, se actualizan los valores de PageRank de cada *n*-grama. Para posteriormente trabajar con los valores de PageRank actualizados y obtener datos más precisos.

Input: Un conjunto *G* de *n-gramas* dados por *MFS* con *n=1…3.*
Output: Un conjunto $A_n = \{A_{n,1}, A_{n,2}, \ldots\}$ donde
$A_{n,j} = \{n\text{-grama}_j, pagerank_j\};$

foreach $n\text{-grama}_i \in G$ **do**
$PR[n\text{-grama}_i] = 1;$
$Link[n\text{-grama}_i, n\text{-grama}_{i+1}];$
end;
forech $y_i \in Link$ **do**
$Out[y_{i,1}]++;$
en
for *aux = 1* **to** *20* **do**
forech $x_i \in PR$ **do**

$$PRT(x_i) = (1 - 0.85) + 0.85 * \left(\sum_{j \in In(x_i)} \frac{1}{|Out(x_j)|} * PR(x_j)\right)$$

end
forech $x_i \in PRT$ **do**
$PR[x_i] = PRT[x_i];$
end
end
forech $x_i \in PR$ **do**
$A_n = A_n\ U[x_i, PR[x_i]];$
end

return *An*

Algoritmo 4: algoritmo Pagerank para conjuntos n-gramas

Una vez obtenidos los *n*-gramas con el PageRank sólo se toman en cuenta los de mayor ranking. Se considera que estos son los términos clave que mejor describen el texto analizado.

CAPÍTULO IV

RESULTADOS Y EVALUACIÓN

4.1 Corpus

El método propuesto para la extracción de términos clave, fue evaluado usando el corpus test-data proporcionado en la tarea #5 de Semeval-2 2010, que tiene como nombre: "Automatic Keyphrase Extraction from Scientific Articles" [SEMEVAL-2], el cual consta de 100 artículos de índole científicos. Los textos contenidos en el test-data se presentan en la Tabla 4.1:

C-14.txt.final	H-10.txt.final	I-1.txt.final	J-1.txt.final
C-1.txt.final	H-11.txt.final	I-10.txt.final	J-10.txt.final
C-17.txt.final	H-12.txt.final	I-11.txt.final	J-11.txt.final
C-18.txt.final	H-13.txt.final	I-12.txt.final	J-13.txt.final
C-19.txt.final	H-14.txt.final	I-14.txt.final	J-14.txt.final
C-20.txt.final	H-16.txt.final	I-15.txt.final	J-15.txt.final
C-22.txt.final	H-17.txt.final	I-16.txt.final	J-17.txt.final
C-23.txt.final	H-18.txt.final	I-18.txt.final	J-18.txt.final
C-27.txt.final	H-19.txt.final	I-19.txt.final	J-2.txt.final
C-28.txt.final	H-2.txt.final	I-20.txt.final	J-20.txt.final
C-29.txt.final	H-20.txt.final	I-21.txt.final	J-21.txt.final
C-3.txt.final	H-21.txt.final	I-22.txt.final	J-22.txt.final
C-30.txt.final	H-24.txt.final	I-26.txt.final	J-23.txt.final
C-31.txt.final	H-25.txt.final	I-29.txt.final	J-25.txt.final
C-32.txt.final	H-26.txt.final	I-30.txt.final	J-26.txt.final
C-33.txt.final	H-29.txt.final	I-31.txt.final	J-27.txt.final
C-34.txt.final	H-3.txt.final	I-32.txt.final	J-28.txt.final
C-36.txt.final	H-30.txt.final	I-33.txt.final	J-3.txt.final
C-38.txt.final	H-31.txt.final	I-34.txt.final	J-30.txt.final
C-4.txt.final	H-32.txt.final	I-35.txt.final	J-31.txt.final
C-40.txt.final	H-4.txt.final	I-4.txt.final	J-32.txt.final
C-6.txt.final	H-5.txt.final	I-5.txt.final	J-4.txt.final
C-8.txt.final	H-7.txt.final	I-6.txt.final	J-7.txt.final
C-86.txt.final	H-8.txt.final	I-7.txt.final	J-8.txt.final
C-9.txt.final	H-9.txt.final	I-9.txt.final	J-9.txt.final

Tabla 4.1 Textos usados en el proceso de evaluación

4.2 Métricas de evaluación

La evaluación consta de listar 15 términos clave candidatos de cada documento del test-data, estos términos clave están compuestos por 1, 2, 3 – gramas, ya que el modelo usado para esta evaluación se basa únicamente en los primeros 15 términos, los cuales fueron calculados y obtenidos por Algoritmo 2. Una vez evaluado el texto por MFS y PageRank, la selección de estos candidatos se hace de la siguiente manera:

***n*-acrónimos** donde *n*=0,1,2,3
***m*-significado del acrónimo** donde *m = n*
***j*-unigramas** donde *j*=*n*-3
k-trigramas
***w*-bigramas** donde w= (n+m+j+k)-15)

- ***n*-acrónimos**

Se realizar una búsqueda y extracción de *n*-acrónimos dentro de los 10 unigramas con el PageRank más alto, donde *n* = 0, 1, 2, 3.

- ***m*-significado del acrónimo**

Una vez obtenidos los *n*-acrónimos, se realiza la búsqueda del significado o definición del acrónimo, por lo que *m*=*n*.

- ***j*-unigramas**

Se selección j-unigramas de los 10 unigramas con el PageRank más alto, sin tomar en cuenta los n-acrónimos seleccionados anteriormente, es decir *j*=*n-3*.

- **k-trigramas**

Se realiza una búsqueda y extracción de *k*-trigramas de los 10 trigramas con el PageRank mas alto, donde *k=3*.

- ***w*-bigramas**

Se seleccionan los 10 unigramas con el PageRank más alto y los 10 bigramas con el PageRank más alto.

Si existe un unigrama que se encuentre dentro de algún bigrama, el valor del PageRank del bigrama es modificado, ya que se le suma el PageRank del unigrama, es decir, *pagerankbigrama = pagerankbigrama + pagerankunigrama*.

Una vez realizada la modificación anterior, se vuelve a ordenar los bigramas con respecto a su PageRank modificado, se seleccionan *w*-bigramas de la lista de bigramas modificada, donde **w= (n+m+j+k)-15),** de estar forma tener los 15 términos necesarios para la evaluación.

Se evaluaron los resultados obtenidos manejando precision, recall y Fscore con respecto a dos diferentes gold-standards que son:

- Conjunto de términos clave dados por el lector.
- Conjunto de término clave dados tanto por el lector como por el autor.

4.3 Baselines

Los resultados finales obtenidos de la evaluación con respecto a los gold-standards mostrados en la Tabla 4.2, fueron comparados con tres distintos baselines: TF-IDF, Naïve Bayes y Maximum Entropy [SEMEVAL-2].

En las siguientes tablas, P, R, F representan precision, recall y F-score respectivamente. En la Tabla 4.2, TF-IDF es un método no supervisado, NB y ME son métodos supervisados los cuales se basan en Naïve Bayes y Maximum Entropy respectivamente. En la segunda columna, R representa los resultados obtenidos dada la evaluación con respecto a los términos usados por el lector del conjunto gold-standards y C representa los resultados obtenidos dada la evaluación con respecto a los términos usados por el autor y lector del conjunto gold-standards, estos divididos en tres sección los cuales describen la forma de selección, es decir, evalúa conforme a la selección de los 5 primeros términos, los 10 primeros términos y por último los 15 términos.

Method	by	Top 5 candidates			Top 10 candidates			Top 15 candidates		
		P	R	F	P	R	F	P	R	F
TF-IDF	R	17.80 %	7.39 %	10.44 %	13.90 %	11.54 %	12.61 %	11.60 %	14.45 %	12.87 %
	C	22.00 %	7.50 %	11.19 %	17.70 %	12.07 %	14.35 %	14.93 %	15.28 %	15.10 %
NB	R	16.80 %	6.98 %	9.86 %	13.30 %	11.05 %	12.07 %	11.40 %	14.20 %	12.65 %
	C	21.40 %	7.30 %	10.89 %	17.30 %	11.80 %	14.03 %	14.53 %	14.87%	14.70 %
ME	R	16.80 %	6.98 %	9.86 %	13.30 %	11.05 %	12.07 %	11.40 %	14.20 %	12.65 %
	C	21.40 %	7.30 %	10.89%	17.30 %	11.80 %	14.03 %	14.53 %	14.87 %	14.70 %

Tabla 4.2: Baselines

En la Tabla 4.2 se puede observar que el mejor método dentro del baseline es el TF-IDF por una diferencia aproximada de 0.20 % con respecto a R, y 40% con relación a C, sobre los otros dos métodos (NB y ME).

4.4 Resultados Experimentales

El método propuesto en esta tesis tuvo como salida de evaluación los resultados mostrados en la tabla 4.3:

Method	by	Top 5 candidates			Top 10 candidates			Top 15 candidates		
		P	R	F	P	R	F	P	R	F
MFS - PAGERANK	R	10.40 %	4.32 %	6.10 %	13.90 %	11.54 %	12.61 %	14.93 %	18.60 %	16.56 %
	C	13.60 %	4.64 %	6.92 %	17.60 %	12.01 %	14.28 %	19.00 %	19.44 %	19.22 %

Tabla 4.3: Método MFS – PageRank

4.5 Comparación de resultados

En esta sección se mostrará gráficamente los resultados obtenidos junto con los del baseline para una mejor visualización y comparación.

2.5.1 Lector

En las Figuras 4.1 – 4.3, se muestra una comparación de los resultados obtenidos en el método propuesto en esta tesis (MFS-PageRank) con los resultados del Baseline (TF-IDF, NB, ME), de acuerdo a los resultados del top 5, 10 y 15 términos candidatos, en relación a los términos usados por el lector.

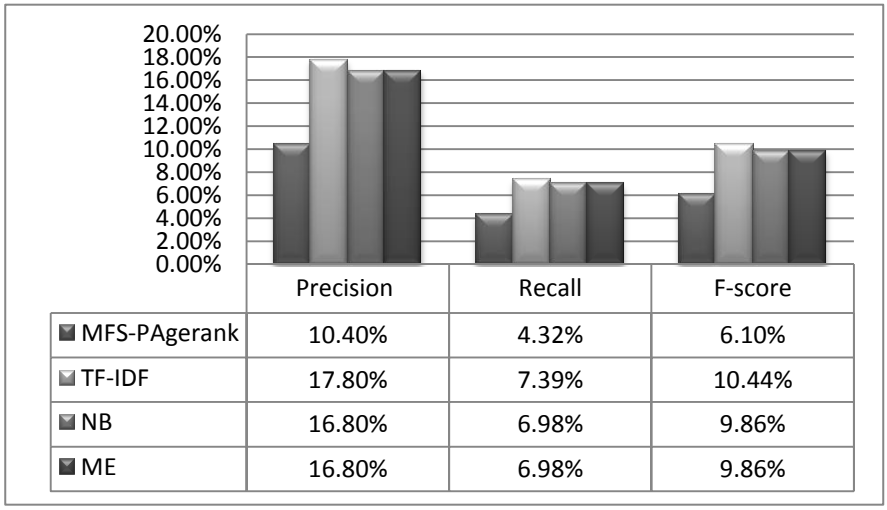
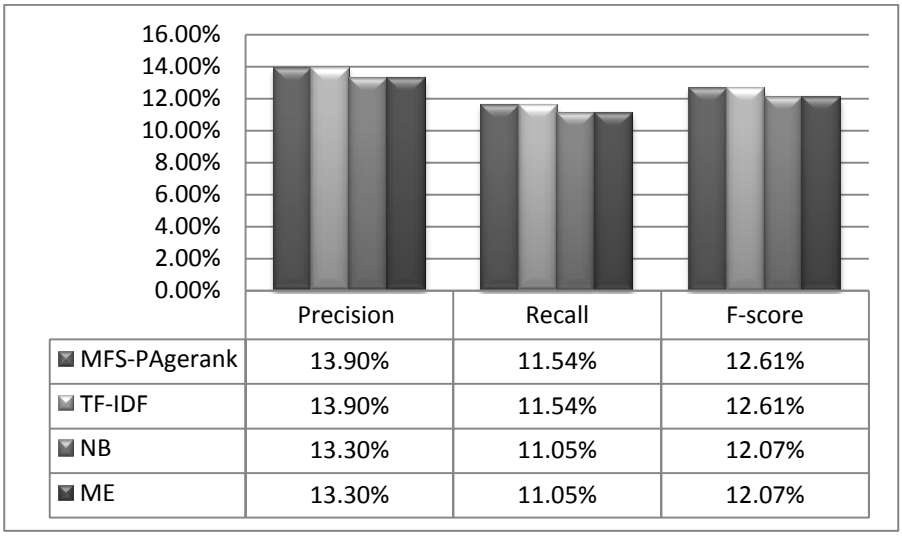


Figura 4.1: Comparación del top 5 candidates

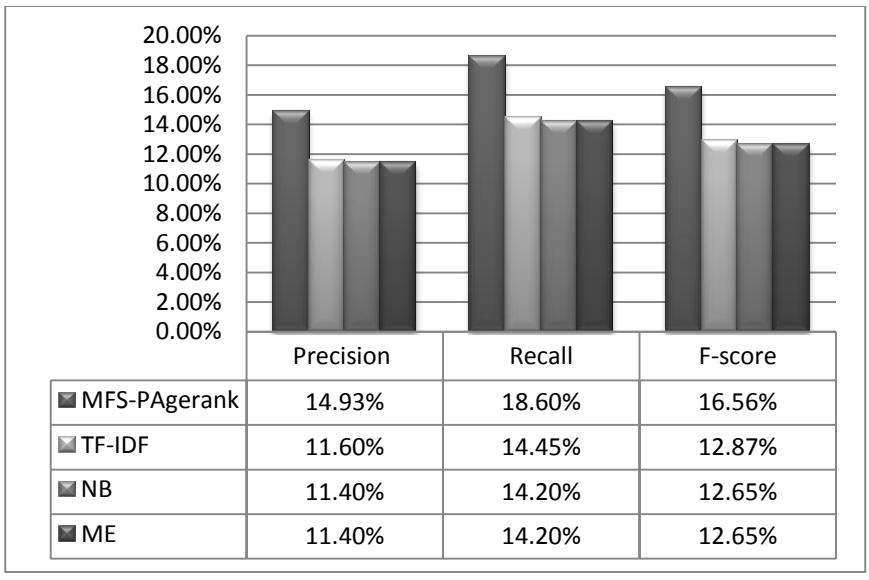
El la Figura 4.1 se observa que al tomar sólo los primeros 5 términos clave candidatos los resultados del baseline son superiores al del método propuesto, ya que los términos candidatos dados por MFS-Pagerank, no tienen muchas coincidencias con respecto a los términos dados por el lector.



	Precision	Recall	F-score
MFS-PAgerank	13.90%	11.54%	12.61%
TF-IDF	13.90%	11.54%	12.61%
NB	13.30%	11.05%	12.07%
ME	13.30%	11.05%	12.07%

Figura 4.2: Comparación del top 10 candidates

El la Figura 4.2 se observa una mejoría del método MFS-PAgerank al tomar los 10 primeros términos clave candidatos, debido a que los resultados son similares con respecto a los del baseline, en particular al método TF-IDF, además estos son superiores en relación a NB y ME.



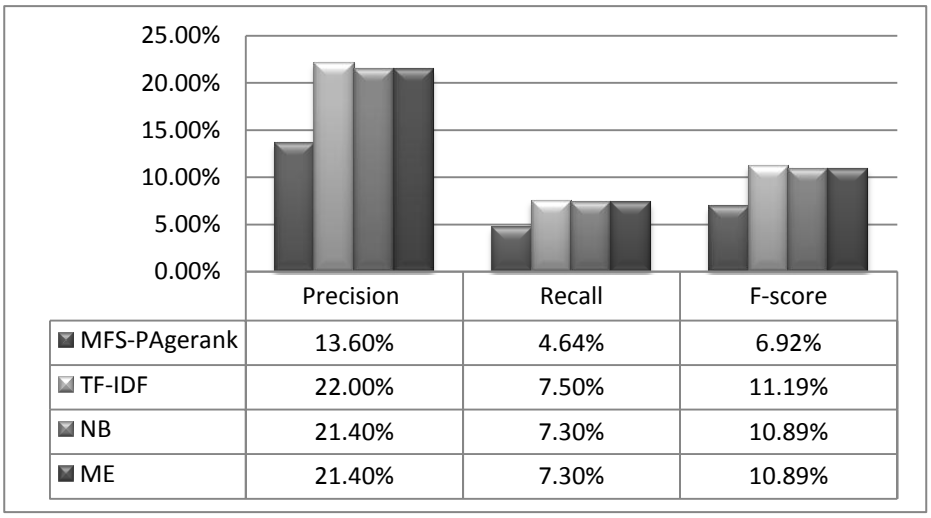
	Precision	Recall	F-score
MFS-PAgerank	14.93%	18.60%	16.56%
TF-IDF	11.60%	14.45%	12.87%
NB	11.40%	14.20%	12.65%
ME	11.40%	14.20%	12.65%

Figura 4.3: Comparación del top 15 candidates

El la Figura 4.3 se muestra como al seleccionar los 15 términos clave candidatos por parte de MFS-Pagerank tiene una mejor precisión y recall lo cual hace que este método supere los resultados dados por el baseline, lo cual nos dice que tuve mejores coincidencias en relación a los términos dados por el lector.

2.5.2 Autor y lector

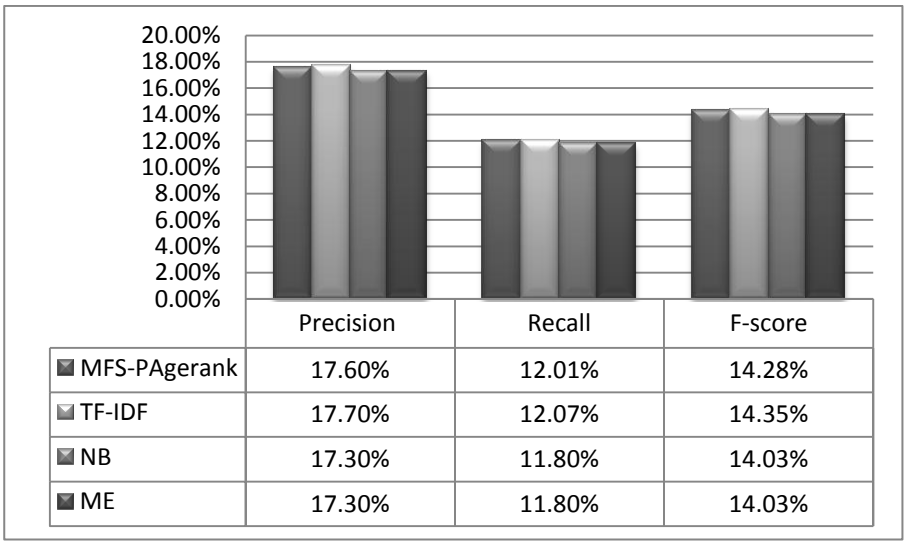
En las Figuras 4.1 – 4.3, se muestra una comparación de los resultados obtenidos en el método propuesto en esta tesis (MFS-PageRank) con los resultado del Baseline (TF-IDF, NB, ME), de acuerdo a los resultados del top 5, 10 y 15 términos candidatos, en relación a los términos usados por el autor y el lector.



	Precision	Recall	F-score
MFS-PAgerank	13.60%	4.64%	6.92%
TF-IDF	22.00%	7.50%	11.19%
NB	21.40%	7.30%	10.89%
ME	21.40%	7.30%	10.89%

Figura 4.4: Comparación del top 5 candidates

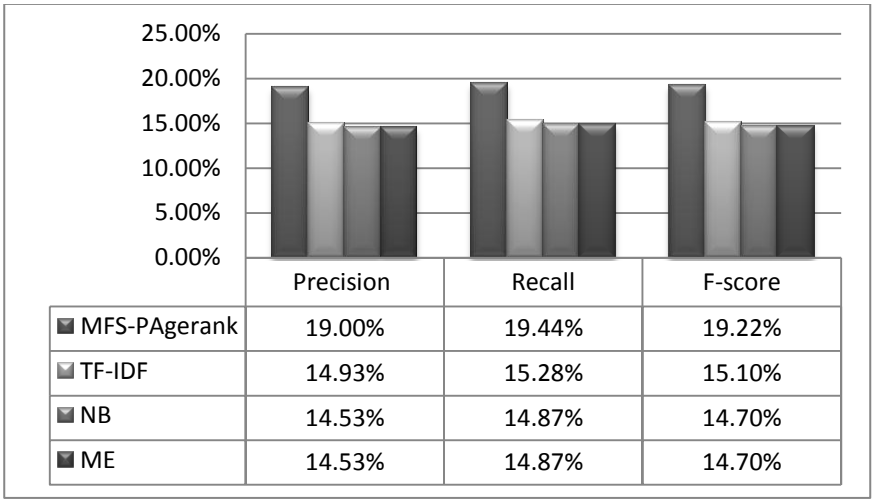
El la Figura 4.4 se observa que al tomar sólo los primeros 5 términos clave candidatos los resultados del baseline son superiores al del método propuesto, ya que los términos candidatos dados por MFS-Pagerank, no tienen muchas coincidencias con respecto a la combinación de los términos dados por el lector y el autor.



	Precision	Recall	F-score
MFS-PAgerank	17.60%	12.01%	14.28%
TF-IDF	17.70%	12.07%	14.35%
NB	17.30%	11.80%	14.03%
ME	17.30%	11.80%	14.03%

Figura 4.5: Comparación del top 10 candidates

El la Figura 4.5 se observa una mejoría del método al tomar los 10 primeros términos clave candidatos, debido a que los resultados son similares con respecto al del baseline, en particular al método TF-IDF ya que la diferencia de estos dos es de 0.10%, además son superiores en relación a NB y ME por 0.30%.



	Precision	Recall	F-score
MFS-PAgerank	19.00%	19.44%	19.22%
TF-IDF	14.93%	15.28%	15.10%
NB	14.53%	14.87%	14.70%
ME	14.53%	14.87%	14.70%

Figura 4.6: Comparación del top 15 candidates

El la Figura 4.3 se muestra como al seleccionar los 15 términos clave candidatos por parte de MFS-Pagerank tiene una mejor precisión y recall lo cual hace que este método supere los resultados dados por el baseline, lo cual nos dice que tuve mejores coincidencias en relación a los términos combinados del lector y autor.

CONCLUSIONES

En este trabajo de tesis se ha presentado un enfoque basado en Secuencias Frecuentes Maximales (MFS) los cuales fueron evaluados, usando el algoritmo de PageRank para generar la extracción de términos clave de un texto y así localizar temas que referencian a dicho texto.

Para este trabajo en primera instancia se llevo a cabo una investigación acerca de los métodos propuestos (MFS y Pagerank), este estudio se realizó con la finalidad de entender su funcionamiento y así relacionarlos con el objetivo principal de esta tesis.

Se evaluó el desempeño de la extracción de términos clave con respecto a la tarea #5 de Semeval-2 2010 y se comparó con tres distintas baselines: TF-IDF, Naïve Bayes y Maximum Entropy, los cuales arrojaron como resultado un mejor comportamiento y evaluación con respecto a los tres baselines dentro del top 15 con una diferencia de 4% a 6% de mayor efectividad, sin embargo dentro del top 10 sólo se supero en la evaluación con respecto al lector con una diferencia de 1% mayor de efectividad .

Con estos resultados se observa que con respecto a los términos dados por el escritor en el gold-standard, no existió una buena precisión del método propuesto. Sin embargo con respecto a los términos dados por el lector se tuvo una mejor evaluación. Por otro lado, cabe mencionar que tanto lector como escritor asignan distintos términos clave, ya que estos cuentan con diferentes criterios de selección de estos términos.

En conclusión nuestro sistema obtiene buenos resultados, es decir, nos da como resultado los términos clave más aproximados a lo que se desea llegar, con lo que podemos decir que los objetivos planteados para la realización de esta tesis se cumplieron satisfactoriamente, sin embargo, pudimos observar que nuestros resultados necesitan ser analizados más a fondo, sobre todo dentro del procedimiento de selección y clasificación de los términos, y así obtener una mayor precisión y mejorar los resultados dentro del top 5.

Si bien es cierto que la investigación realizada en este trabajo de tesis ha obtenido resultados aceptables, los métodos propuestos se pueden mejorar modificando y añadiendo nuevas técnicas de extracción y selección de términos clave tales como el *punto de transición* que se refiere básicamente a un valor de frecuencia en el vocabulario del texto que divide al mismo vocabulario en términos de alta y baja frecuencia **[Reyes et al.2003]** y la *técnica de información mutua* el cual se basa principalmente en calcular el grado de coherencia entre un par dado de palabras de un texto **[Manning2003]**, así cambiar la forma de selección de los *n*-gramas de modo que se ajusten a las necesidades de extracción optimas de cada texto.

Cabe mencionar que este trabajo fue expuesto con el nombre de "BUAP: An Unsupervised Approach to Automatic Keyphrase Extraction from Scientific Articles" **[Ortiz et al.2010]** dentro Work-Shop SemEval-2010 con sede en Uppsala, Sweden, el día 15 de julio de 2010 [SEMEVAL-2]. Además este trabajo será publicado en las memorias del evento que son editadas por la asociación más reconocida a nivel mundial en lingüística computacional: La Association for Computational Linguistics (ACL) .

BIBLIOGRAFÍA

[Ahonen H.2002] Ahonen H. 2002. *Discovery of Frequent Word Sequences in Text*. Proceedings of the ESF Exploratory Workshop on Pattern Detection and Discovery. London, UK, pp. 180-189.

[Barker and Corrnacchia2000] K. Barker and N. Corrnacchia. 2000. Using noun phrase heads to extract document keyphrases. In *13th Biennial Conference of the Canadian Society on Computational Studies of Intelligence: Advances in Artificial Intelligence.*

[Barzilay and Elhadad1997] R. Barzilay and M. Elhadad. 1997. Using lexical chains for text summarization. In *ACL/EACL 1997 Workshop on Intelligent Scalable Text Summarization*, pages 10–17.

[Brin and Page1998] Sergey Brin and Lawrence Page. 1998. The anatomy of a large-scale hypertextual web search engine. In *COMPUTER NETWORKS AND ISDN SYSTEMS*, pages 107–117. Elsevier Science Publishers B. V.

[Damerau1993] Fred J. Damerau. *Generating and evaluating domain-oriented multi-word terms from texts*. Information Processing and Management, 29(4):433–447.

[DAvanzo and Magnini2005] E. DAvanzo and B. Magnini. 2005. A keyphrase-based approach to summarization: the lake system. In *Document Understanding Conferences* (DUC-2005).

[Frank et al.1999] E. Frank, G.W. Paynter, I. Witten, C. Gutwin, and C.G. Nevill-Manning. 1999. Domain specific keyphrase extraction. In *16th International Joint Conference on AI*, pages 668–673.

[Garfield1983] E. Garfield. 1983. In *How to use the Science Citation Inclex (SCI)*, 9: 5-14.

[Jaynes1979] Edwin T. Jaynes. *The Maximum Entropy Formalism, chapter "Where do we Stand on MaximumEntropy?"*. MIT Press.

[Krovetz R.1993] Krovetz, R., *Viewing morphology as an inference process*. Proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR'93, 191-203.

[krulwich and Burkey1996] Krulwich, B., and Burkey, C. (,*Learning user information interests through the extraction of semantically significant phrases*. In M. Hearst and H. Hirsh, editors, AAAI 1996 Spring Symposium on Machine Learning in Information Access. California: AAAI Press.

[Langley P. and Thompson K.1992] Langley, and thompson, K., *An analysis of bayesian classifiers*. En: aaai-92.

[Langville et al.2006] Langville, Amy N.; Meyer, Carl D. 2006. *Google's PageRank and Beyond: The Science of Search Engine Rankings*. Princeton University Press.

[Lawrie et al.2001] D. Lawrie, W. B. Croft, and A. Rosenberg. 2001. *Finding topic words for hierarchical summarization.* In SIGIR 2001.

[Lovins J.B.1968] Lovins, J.B., *Development of a stemming algorithm*. Mechanical Translation and Computational Linguistics, 11, 22-31.

[Lucio2002] Lucio Paolo, J. Correia, M. Mamede and Hagége C., *Using Morphological, Syntactical and Statistical Information for Automatic Term Acquisition*. In Proceedings of Third International Conference, Protal 2002, Lisboa, Portugal.

[Luhn P.1958] Luhn Hans Peter .1958. *The Automatic Creation of Literature abstract.* National Convention, IBM Journal.

[Manning2003] Manning, C. D., & Shutze, H.,"*Foundations of Statistical Natural Language Processing"*, The MIT Press, 2003.

[Mateo et. al., 2003] Pedro Luis Mateo, José Carlos González, Julio Villena y José Luis Martínez, *Un sistema para resumen automático de textos en castellano.* DAEDALUS, S.A.

[MC Callum et al.1998] Mc callum, a. Y nigan, k. *A comparison of event models for naive bayes text classification.* ICML workshop on learning for text classification.

[Mihalcea and Tarau2004] R. Mihalcea and P. Tarau. 2004. Textrank: Bringing order into texts. In EMNLP 2004, ACL, pages 404–411.

[Muñoz1996] *Compound key word generation from document databases using a hierarchicalclustering ART model*. Intelligent Data Analysis Amsterdam: Elsevier.

[Nakagawa1997] Nakagawa, H. 1997. *"Extraction of index words from manuals"*. In *Conference Proceedings of Computer-Assisted Information Searching on Internet*, 598-611.

[O. Medelyan2008] I. H. Witten O. Medelyan. 2008. Domain independent automatic keyphrase indexing with small training sets. *Journal of American gSociety for Information Science and Technology*, 59(7):1026–1040.

[Ortiz et al.2010] Roberto Ortíz, David Pinto, Mireya Tovar, Héctor Jiménez-Salazar: *BUAP:An Unsupervised Approach to Automatic Keyphrase Extraction from Scientific Articles*. 5th. Workshop on Semantic Evaluations - SemEval-2 2010. Association for Computational Linguistics, ACL Anthology.

[Page1998] Lawrence Page, Sergey Brin, Rajeev Motwani, Terry Winograd 1998. The PageRank Citation Ranking: Bringing Order to the Web.

[Paice C. and Jones P.1993] Paice, C.D., and Jones, P.A. 1993, *The identification of important concepts in highly structured technical papers.* SIGIR-93: Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 69-78, New York: ACM.

[Quinlan1986] Quinlan, J.R. *Induction of Decision Trees*. Machine Learning, 1, 81-106. 1996.

[Reyes et al.2003] Reyes B., Moyotl E & Jimenez H., "*Reduccion de terminos indice usando el punto de transicion*", 2003.

[SEMEVAL-2] SemEval-2010, *Evaluation Exercises on Semantic Evaluati*on, ACL (Association for Computational Linguistics) SigLex (Special Interest Group on the Lexicon), http://semeval2.fbk.eu/semeval2.php

[Shannon1948] Claude E. Shannon. A *mathematical theory of communication*. Bell System Technical Journal ,27:379–423.

[Soderland and Lehnert1994] Soderland, S., and Lehnert, W., *Wrap-Up: A trainable discourse module for information extraction*. Journal of Artificial Intelligence Research, 2, 131-158, 1994.

[Steier and Belew1993] Steier, A. M., and Belew, R. K.. *Exporting phrases: A statistical analysis of topical language*. In R. Casey and B. Croft, editors, Second Symposium on Document Analysis and Information Retrieval, pp. 179-190.

[Strzalkowski T.1999] Strzalkowski, T. 1999. *Natural Language Information Retrieval. Netherlands: Kluwer Academic Publishers*.

[Turney1999] P.D. Turney."Learning to extract keyphrases from text", Technical Report ERB- 1057, National Research Council, Institute for InformationTechnology, 1999.

[whitley1989] Whitley, D. *The GENITOR algorithm and selective pressure*. Proceedings of the Third International Conference on Genetic Algorithms *(ICGA-89)*, pp. 116-121.1989.

[Witten et al.1999] I. Witten, G. Paynter, E. Frank, C. Gutwin, and G. Nevill-Manning. 1999. Kea: practical automatic key phrase extraction. In *fourth ACM conference on Digital libraries*, pages 254–256.