

"SWMiner"
**Determinación Automática de Perfiles de Usuarios en
Internet utilizando Técnicas de Data Mining**

Tesis de Licenciatura



Universidad de Buenos Aires
Facultad de Ciencias Exactas y Naturales
Departamento de Computación

Alumnos

Mariela Da Graca Guerra	LU 1331/88
María Eugenia Modarelli	LU 465/92

Director

Alejandro Vaisman

Abril 2003

Resumen	6
CAPÍTULO 1: Introducción	7
1.1 Introducción.....	7
1.2 Motivación	8
1.3 Contribuciones	9
1.4 Organización de la Tesis	9
CAPÍTULO 2: Estado del Arte	10
2.1 Introducción.....	10
2.2 Categorización de Páginas Web	10
2.2.1 Clasificación con Mega documentos.....	11
2.2.2 Clustering de Hipertexto en Aplicaciones de Búsqueda en la Web.....	11
2.2.3 TopCat	12
2.2.4 Método de Selección de Palabras Claves	13
2.2.5 Reputación de una Página sobre un Tópico.....	13
2.2.6 Centros y Autoridades	14
2.3 Resumen	17
CAPÍTULO 3: Marco Teórico	19
3.1 Introducción.....	19
3.2 Log Files.....	19
3.2.1 Tipos de Log File.....	20
3.2.2 Formatos de Log File	21
3.2.3 Data Cleaning de Log Files	24
3.3 Identificación de Transacciones	24
3.3.1 Modelos de Identificación de Transacciones.....	25
3.4 Preprocesamiento de Documentos Web	27
3.4.1 Information Extraction e Information Retrieval	28
3.4.2 Tipos de Documentos	28
3.4.3 ¿Podemos analizar cualquier página web?.....	29
3.4.4 ¿Qué es mejor: traducir los documentos o inferir su idioma?.....	29
3.4.5 Categorización de Texto usando Estadísticas de Frecuencias de NGrams.....	29
3.4.6 ¿Debemos extraer todas las palabras contenidas en los documentos?	31
3.4.7 ¿Todas las palabras extraídas sirven para la categorización semántica?	32
3.5 Categorización Semántica	36
3.5.1 Reglas de Asociación (Association Rules)	36
3.5.2 Definición Formal del Problema.....	37
3.5.3 Itemsets Frecuentes.....	37
3.5.4 Algoritmos de Reglas de Asociación.....	38

3.6	WordNet.....	38
3.6.1	Tipos de Relaciones.....	39
3.6.2	Generalización de Tópicos.....	40
3.6.3	Ambigüedad Sintáctica y Semántica.....	41
3.7	Clustering.....	43
3.7.1	Medidas de Similaridad.....	44
3.7.2	Clasificación de Algoritmos de Clustering.....	45
3.7.3	Peso de un Cluster.....	46
3.8	Bondad de un Tópico.....	46
3.9	Métricas de Evaluación.....	47
3.9.1	Problemas de Precision y Recall.....	48
3.9.2	Definiciones de Precision y Recall para SWMiner.....	48
3.10	Resumen.....	49
CAPÍTULO 4: Arquitectura de SWMiner		50
4.1	Introducción.....	50
4.2	Proceso General de SWMiner.....	50
4.2.1	Depurador.....	51
4.2.2	Preprocesador.....	51
4.2.3	Categorizador.....	58
4.3	Resumen.....	59
CAPÍTULO 5: Implementación		60
5.1	Introducción.....	60
5.2	Requerimientos del Sistema.....	60
5.3	Estructura de Directorios.....	60
5.4	Implementación del Depurador.....	62
5.4.1	Data Cleaning.....	62
5.4.2	Pseudo código del Proceso de Data Cleaning.....	63
5.5	Implementación del Preprocesador.....	66
5.5.1	Distinción de Tipo de Página.....	66
5.5.2	Identificación de Idiomas.....	67
5.5.3	Obtención de Páginas Vecinas.....	70
5.5.4	Representación de Páginas.....	73
5.6	Implementación del Categorizador.....	78
5.6.1	Generación de Itemsets Frecuentes.....	79
5.6.2	Clustering de Itemsets Frecuentes.....	80
5.6.3	Obtención de Tópicos.....	81
5.6.4	Selección del tópico de mayor Bondad.....	90
5.6.5	Mapeo Tópicos-Usuarios.....	92

5.7	Resumen	92
CAPITULO 6: Experimentación		93
6.1	Introducción.....	93
6.2	Experimentación.....	93
6.2.1	Identificación de Idiomas.....	93
6.2.2	Part Of Speech	93
6.2.3	Lemmatiser.....	93
6.2.4	Clustering e Identificación de Tópicos.....	94
6.2.5	SWMiner vs. Google	94
6.2.6	SWMiner vs. TOPIC.....	94
6.2.7	SWMiner vs. Categorías asignadas manualmente.....	95
6.2.8	Experimentos	95
6.3	Experimentos con Log de Mamíferos Marinos.....	95
6.3.1	SWMiner con Normalización Morfológica y con Clustering.....	95
6.3.2	SWMiner con Normalización Morfológica y sin Clustering.....	111
6.3.3	SWMiner sin Normalización Morfológica y con Clustering.....	113
6.3.4	SWMiner sin Normalización Morfológica y sin Clustering.....	116
6.3.5	SWMiner vs. TOPIC.....	117
6.3.6	SWMiner vs. Categorías asignadas manualmente.....	118
6.4	Experimentos con Log de Animales Salvajes.....	119
6.4.1	SWMiner con Normalización Morfológica y con Clustering.....	119
6.4.2	SWMiner con Normalización Morfológica y sin Clustering.....	123
6.4.3	SWMiner sin Normalización Morfológica y con Clustering.....	125
6.4.4	SWMiner sin Normalización Morfológica y sin Clustering.....	125
6.4.5	SWMiner vs. TOPIC.....	126
6.4.6	SWMiner vs. Categorías asignadas manualmente.....	127
6.5	Experimentos con Log General	128
6.5.1	SWMiner con Normalización Morfológica y con Clustering.....	128
6.5.2	SWMiner con Normalización Morfológica y sin Clustering.....	129
6.5.3	SWMiner sin Normalización Morfológica y con Clustering.....	130
6.5.3.1	SWMiner vs. Google	130
6.5.4	SWMiner sin Normalización Morfológica y sin Clustering.....	131
6.5.4.1	SWMiner vs. Google	131
6.5.5	SWMiner vs. TOPIC.....	131
6.5.5.1	SWMiner vs. Categorías asignadas manualmente.....	132
6.6	Evaluación de Resultados.....	133
6.7	Resumen	134
CAPÍTULO 7: Conclusiones		135

7.1	Introducción.....	135
7.2	Conclusiones.....	135
7.3	Aportes	136
7.4	Trabajos Futuros.....	136
	Referencias.....	138
	Referencias Web	139
	Anexo A.....	141
	WordNet	141
	Anexo B.....	145
	Códigos de Estado del Protocolo HTTP.....	145
	Anexo C.....	146
	Ejemplo de Archivo de Perfiles de Idioma	146
	Anexo D.....	147
	Ejemplo de Tags Html.....	147
	Anexo E.....	148
	Stop Words	148
	Anexo F.....	149
	Experimentación	149

Resumen

En los últimos años Internet ha logrado ser parte de nuestras vidas y en muchos casos dependemos de ella para trabajar, estudiar, investigar, intercambiar ideas, etc.

Existen muchas razones matemáticas, sociológicas y comerciales para estudiar y analizar la información disponible en Internet.

En esta tesis presentamos el sistema *SWMiner - Semantic Web Miner* como una herramienta que permite categorizar a un grupo de usuarios de Internet, analizando el contenido de las páginas web accedidas por los mismos. Para llevar a cabo dicha tarea analizamos el *log file* del servidor *Proxy* de una organización, en la que identificamos las acciones realizadas por cada uno de los usuarios, obtenemos las páginas a las cuales accedieron y las categorizamos semánticamente. Finalmente, obtenemos los perfiles de cada usuario basándonos en una jerarquía semántica de conceptos llamada *WordNet*.

Las principales contribuciones de esta tesis son:

- Proponemos una combinación de diferentes técnicas de tratamiento y análisis de documentos web que convergen a la identificación de conceptos de los documentos.
- Definimos fórmulas de similaridad semántica entre conjuntos de palabras, seleccionando una apropiada conceptualización de las mismas en una taxonomía jerárquica de conceptos.
- Presentamos un sistema que incluye distintas técnicas que pueden ser útiles para:
 - ✓ Realizar procesamientos lingüísticos de documentos.
 - ✓ Identificar idiomas de documentos.
 - ✓ Categorizar automáticamente documentos.

CAPÍTULO 1: Introducción

1.1 Introducción

Internet ha experimentado un crecimiento exponencial en estos últimos años, convirtiéndose en un gran almacén de información. Existen miles de bases de datos disponibles, periódicos, tutoriales, información técnica, empresarial, catálogos, etc. Toda esta información disponible convierte a Internet en una gigantesca biblioteca potencial en continuo crecimiento.

Existen muchas razones matemáticas, sociológicas y comerciales para estudiar y analizar la información disponible en Internet.

Actualmente se puede obtener en Internet información sobre casi todo, desde libros hasta tipos de pimientas existentes. Sin embargo, toda esta información puede llegar a ser inmanejable por el usuario. Sistemas automáticos que filtren dicha información y realicen recomendaciones a los usuarios pueden ser de mucha utilidad.

Un sistema de recomendaciones (SR) puede pensarse como un sistema que, a partir de un conjunto de elecciones del usuario, obtenga posibles intereses del mismo según conclusiones basadas en información obtenida de usuarios con elecciones similares. Un sistema de recomendaciones bien conocido puede encontrarse en *Amazon.com*. En este sitio web, los usuarios generan sus propios perfiles registrando los artículos que poseen, compraron o buscaron en el sitio en algún momento. Amazon realiza recomendaciones comparando con otros usuarios que compraron los mismos artículos, y sugiere los artículos que estos otros usuarios poseen, buscaron o compraron. A esta idea se la conoce como *collaborative filtering*. Existe otra aproximación en el diseño de SR llamada *content based recommendation*. En este caso el usuario necesita desarrollar una historia con el SR. Esta aproximación realiza las recomendaciones basándose en las elecciones anteriores de los usuarios, las cuales definen el perfil de los mismos.

Otra técnica que se utiliza para obtener información de Internet es *Web Mining* [CMS97]. Esta técnica aprovecha las técnicas de *Data Mining* [GON96] para descubrir conocimiento de la información disponible en Internet. La obtención de datos en *Web Mining* difiere con la de *Data Mining* en que analiza la información de servidores web, de servidores proxy, del browser o navegador del cliente, de bases de datos adicionales, etc. Los datos obtenidos no solo difieren en cuanto a ubicación de la fuente sino también en la clase de datos disponibles, el segmento de población del cual se obtienen los datos y los métodos para obtenerlos.

Dependiendo del tipo de información que se desee analizar u obtener de Internet, *Web Mining* tiende a complementarse con técnicas de *Information Extraction (IE)*, *Information Retrieval (IR)* y *Text Mining* [EIK99], las cuales deben ser modificadas para adaptarse al nuevo entorno. En este proyecto, utilizamos este conjunto de técnicas para complementar las de *Data Mining* aplicadas en el preprocesamiento y categorización de páginas web.

El objetivo de nuestro proyecto es analizar el comportamiento de los usuarios de una organización que acceden a Internet y así categorizarlos según sus preferencias, intereses, patrones de comportamiento, etc.. Para ello analizamos el contenido de las páginas web a las cuales acceden con el fin de categorizarlas semánticamente y obtener los tópicos de interés de los usuarios. El resultado es una herramienta denominada *SWMiner*.

SWMiner es una herramienta automática de representación y categorización semántica de la información que sigue un proceso de múltiples etapas, a saber;

- Filtrado y depuración de log files;
- Identificación de usuarios y transacciones;
- Procesamiento del texto contenido en las páginas web, obteniendo un conjunto de palabras relevantes como representación de cada página;
- Categorización semántica de páginas web, obteniendo clusters de palabras que comparten una conceptualización común.
- Asignación de categorías a los usuarios.

SWMiner, al igual que otros sistemas, utiliza heurísticas especializadas para reconocer y extraer las palabras relevantes de cada página web. Esta representación de páginas web como un conjunto de palabras, nos permite ubicarnos en el problema conocido como *Market Basket* (Problema del Supermercado), generando *itemsets frecuentes* o grupos de palabras que comúnmente aparecen juntas en las páginas web analizadas [AS94]. Luego, agrupamos los itemsets frecuentes que comparten una conceptualización común en clusters (grupos), utilizando la base jerárquica de conocimientos *WordNet* [W30]. Finalmente, asignamos categorías a los usuarios según la conceptualización semántica obtenida del conjunto de páginas accedidas por los mismos.

1.2 Motivación

La mayoría de los motores de búsqueda emplean una estructura jerárquica para agrupar los documentos web y utilizan un índice que se basa en un conjunto de palabras contenidas o relacionadas a los documentos para poder accederlos. Para agregar un nuevo documento a la estructura es necesario la intervención humana, quien decide en forma manual, a que categoría debe incorporarse el nuevo documento.

En este trabajo proponemos técnicas que mejoran el funcionamiento de los motores de búsqueda, categorizando automáticamente documentos web y simplificando el tedioso trabajo de incorporar un nuevo documento web a dichos motores.

Muchos buscadores utilizan conjuntos de palabras para hallar las páginas de interés en su estructura de índices, y este método no siempre obtiene buenos resultados. Por otra parte, puede ocurrir que el conjunto de palabras ingresadas manualmente al insertar un nuevo documento en la estructura de índices del motor de búsqueda no contenga ninguna de las palabras ingresadas por el usuario al submitir una consulta.

Finalmente, conocer los patrones de comportamiento de un usuario en Internet para personalizar automáticamente los sitios web permitirá simplificar la búsqueda de información, y generar mejores sistemas de recomendación.

El problema a resolver en nuestro proyecto es el siguiente: *Dado un conjunto de páginas web accedidas por un usuario, sobre qué temas tratan?*

Observando una página es sencillo asignarle un *tema* para categorizarla, pero realizar esta tarea automáticamente sin la intervención humana no es trivial, dado que las palabras extraídas de la página pueden tener diferentes significados según el contexto al cual se apliquen, o el tema principal de la página puede no estar mencionado en ella. Por ejemplo, al observar la página principal de *IBM Almaden Research Center* (www.almaden.ibm.com) nos damos cuenta que el *tema* sobre el cual trata es *Data Mining*, sin embargo esta frase no aparece en la página. Para solucionar este problema analizamos un conjunto acotado de páginas que rodean a aquellas accedidas por los usuarios. Como el contenido textual de la Web se enriquece con la estructura de *hiperlinks* que la rodea, ésta puede ser una fuente útil de búsqueda de información.

Una vez determinado el conjunto de páginas a analizar, debemos decidir cómo realizar el análisis, cuál es la información más relevante del conjunto de páginas que estamos

analizando y aplicar alguna técnica de *Data Mining* sobre dicha información para luego categorizar a los usuarios.

1.3 Contribuciones

Las principales contribuciones de esta tesis son:

- Proponemos una combinación de diferentes técnicas de tratamiento y análisis de documentos web que convergen a la identificación de conceptos de los documentos.
- Definimos fórmulas de similaridad semántica entre conjuntos de palabras, seleccionando una apropiada conceptualización de las mismas en una taxonomía jerárquica de conceptos.
- Presentamos un sistema llamado *SWMiner* que incluye distintas técnicas que pueden ser útiles para:
 - ✓ Realizar procesamientos lingüísticos de documentos.
 - ✓ Identificar idiomas de documentos.
 - ✓ Categorizar automáticamente documentos.

Algunas de las posibles aplicaciones de *SWMiner* son:

- Detectar el uso fraudulento de la red en una organización.
- Proveer de información adicional a los buscadores de internet, cuando los usuarios solicitan una búsqueda.
- Determinar dentro de qué categoría es vista una organización a través de su sitio web.
- Utilizarlo como parte de un sistema de recomendación.

1.4 Organización de la Tesis

En el capítulo 2 presentamos el estado del arte describiendo las metodologías y técnicas utilizadas en la actualidad para el preprocesamiento de texto y categorización de páginas web.

En el capítulo 3 introducimos el marco teórico, detallando las metodologías de *IR* (*Information Retrieval*), *IE* (*Information Extraction*), *Data Mining* y *Text Mining* utilizadas en *SWMiner* y presentamos las definiciones necesarias para la comprensión del presente trabajo.

En el capítulo 4 describimos los procesos involucrados en *SWMiner*.

En el capítulo 5 describimos la implementación del sistema, los algoritmos utilizados y presentamos algunos ejemplos del funcionamiento de los mismos.

En el capítulo 6 detallamos las pruebas realizadas con el sistema y los resultados alcanzados.

Finalmente, en el capítulo 7 discutimos las conclusiones y los posibles futuros trabajos que sigan la misma línea de investigación de *SWMiner*.

CAPÍTULO 2: Estado del Arte

2.1 Introducción

La mayoría de los métodos y técnicas que presentamos en este capítulo surgen como posibles soluciones a problemas relacionados con el aprovechamiento de la enorme cantidad de información disponible en WWW. El problema principal en este marco es encontrar páginas relevantes a una consulta determinada utilizando para ello un motor de búsqueda. Generalmente los motores de búsqueda basados en palabras claves (*Keyword Matching*) devuelven muchas páginas web, de las cuales una gran cantidad es irrelevante a la consulta inicial. La razón por la cual sucede esto es que los motores de búsqueda o *buscadores* mantienen índices relacionados a conjuntos acotados de palabras contenidas en las páginas web. Cuando un usuario realiza una consulta al motor de búsqueda, éste busca en el índice las palabras contenidas en la consulta y devuelve la lista de *URLs* (*Uniform Resource Locator*, dirección o localización de un documento en Internet) cuyas páginas contienen dichas palabras.

Para facilitar las búsquedas, los buscadores organizan las páginas mediante una clasificación jerárquica mantenida con intervención humana. Cuando alguien desea registrar una nueva página, los editores la examinan para determinar en forma manual en qué categoría debe incluirse. Clasificar páginas manualmente es muy costoso, requiere un gran esfuerzo intelectual, es difícil de mantener y no cubre todos los tópicos posibles; a pesar de ello, en la actualidad, la performance en cuanto a resultados obtenidos en las búsquedas es superior a la clasificación automática.

Motivados por este problema muchas comunidades científicas comenzaron a plantear posibles soluciones a la categorización automática de documentos web para mejorar el funcionamiento de los motores de búsqueda.

La categorización de documentos web es diferente a la categorización de colecciones de documentos clásicos. En esta, todos los documentos tienen una estructura similar, mientras que los documentos web son heterogéneos y pueden contener desde banners de publicidad hasta código de programación.

Existen muchos métodos para categorizar documentos web, algunos sólo analizan el contenido de los documentos involucrados sin tener en cuenta el contexto en el cual se encuentran y otros aprovechan la información que brinda el contexto para favorecer el análisis. Sabemos que el contenido textual de la web se enriquece con la estructura de hipervínculos que la rodea, la cual puede ser una fuente útil de búsqueda de información. Un hipervínculo entre dos documentos da la idea de que ambos documentos contienen información relacionada. En general, el contexto de un documento web se define como el conjunto formado por los documentos web que contienen hipervínculos que apuntan al documento en análisis más los documentos web apuntados por los hipervínculos incluidos en el mismo.

En nuestro caso particular la categorización de documentos web tiene suma importancia, ya que necesitamos inferir el tópico de las páginas a las cuales accedieron los usuarios de la organización para poder luego categorizarlos.

2.2 Categorización de Páginas Web

A continuación presentamos algunos de los métodos más utilizados en la actualidad para la categorización automática de documentos web.

2.2.1 Clasificación con Mega documentos

Este método [KF00] necesita conocer de antemano las categorías a las cuales pertenecerán los documentos.

El método inicialmente concatena el texto de todos los documentos pertenecientes a una categoría, generando un mega-documento, derivando un vector aplicando un método estándar para calcular el peso (*tf*) y la frecuencia (*idf*) de los términos. Finalmente se obtiene un mega-documento por categoría y el número total de mega documentos es igual al número de categorías.

Para clasificar un documento nuevo, se extraen todos los términos del documento, se seleccionan los mejores n términos y se los utiliza como una consulta para obtener el mega-documento más similar. La búsqueda de similitud se basa en el producto escalar, el cual es calculado usando el método de base de datos deductiva probabilística *HySpirit*.

Para pesar los términos de los mega documentos se utilizan los pesos estándar *tf* e *idf*. Para todo término en el documento se calcula el peso *tf* con la siguiente fórmula:

$$tf_{i,d} = 0.5(1 + f_{i,d} / m_d)$$

donde $f_{i,d}$ es la frecuencia del término t_i en el documento d y m_d es la frecuencia máxima de un término en este documento. La frecuencia inversa de un término es calculada con la siguiente fórmula:

$$idf_i = \log N / n_i$$

donde N es el número total de documentos en la colección y n_i es el número de documentos que contienen el término t_i .

El peso global calculado por *HySpirit* es:

$$w_{i,d} = tf_{i,d} \times idf_i$$

donde i denota el índice del término y d el índice del documento. Esta técnica asume que los términos son mutuamente independientes. Para incrementar la eficiencia del proceso de categorización no se consideran todos los términos de los documentos, se seleccionan los mejores x términos. Estos términos son rankeados de acuerdo a valores decrecientes de *idf*.

2.2.2 Clustering de Hipertexto en Aplicaciones de Búsqueda en la Web

Este método [MS96] propone un algoritmo que arma clusters de documentos de hipertexto usando las palabras contenidas en el documento, los links entrantes y los salientes. Esta técnica considera que dos documentos son semánticamente similares si comparten una o más palabras.

El método propone una codificación geométrica de la web, donde supone que W es la colección completa de documentos de la web y Q un subconjunto de documentos obtenidos con un motor de búsqueda, a los cuales les aplica el algoritmo de clustering.

Cada documento d de Q se representa como una tripla (D, F, B) donde D , F y B capturan respectivamente la información representada por las palabras contenidas en d , los links salientes y los links entrantes respectivamente.

La creación de D es un ejercicio estándar de *Text Mining* o de *Information Retrieval*. La idea básica es construir un diccionario de palabras de todas las palabras que aparecen en alguno de los documentos en Q y podar o eliminar las palabras del diccionario que no ayudan a discriminar semánticamente un cluster del otro. En este método se eliminan aquellas palabras que aparecen en menos de dos documentos, los tags html [W16] y las

stopwords [W29]. Luego asigna un identificador numérico unívoco a cada una de las palabras del diccionario y construye el vector D de la siguiente manera: la entrada i en D representa la frecuencia de aparición de la palabra con identificador i en el documento.

Para obtener F se construye un diccionario de links salientes con todos los documentos que son apuntados por links desde los documentos contenidos en Q , luego se eliminan los documentos del diccionario que son apuntados por menos de dos documentos de Q . A los documentos resultantes les asigna un identificador numérico y construye un vector donde la i -ésima entrada es la cantidad de links en la página d que apuntan a la página con identificación i .

La creación de B es similar a la de F pero con los links entrantes. El método normaliza con Euclides los componentes D , F y B de cada documento d de Q .

Finalmente, este método presenta una medida de similaridad que utiliza con el algoritmo *Toric K-means* [MS96] para agrupar documentos en clusters.

Dados dos documentos $x = (D_1, F_1, B_1)$ e $y = (D_2, F_2, B_2)$ la medida de similaridad entre ellos se define como la suma del producto entre los pesos de cada uno de sus componentes. Es decir,

$$S(x, y) = \alpha_1 D_1 D_2 + \alpha_2 F_1 F_2 + \alpha_3 B_1 B_2$$

donde α_1 , α_2 y α_3 son números no negativos tales que,

$$\alpha_1 + \alpha_2 + \alpha_3 = 1$$

Para categorizar un nuevo documento se calcula la medida de similaridad con al menos un documento de cada cluster para determinar en cual de ellos debería incluirse.

2.2.3 TopCat

TopCat [CCR00, HKK97, DAHKRV97] es una técnica para identificar tópicos en un documento. Primero identifica las entidades claves en un documento, luego agrupa estas entidades para encontrar tópicos y finalmente mapea los tópicos a los documentos.

TopCat consta de tres etapas: *Identificación de Entidades*, *Clustering de Tópicos* y *Mapeo de Tópicos*. La identificación de entidades utiliza técnicas de procesamiento del lenguaje natural para identificar entidades claves en un documento, permitiendo representar dicho documento como un conjunto de ítems.

En el contexto de *Data Mining* la identificación de tópicos se traduce en identificar grupos de ítems relacionados. Uno de los mecanismos utilizados es el de encontrar reglas de asociación (*Association Rules* [ASI93]) entre los ítems. Para realizar este proceso se utiliza el sistema Alembic [DAHKRV97]. La etapa de clustering consiste en agrupar los conceptos claves para encontrar tópicos. La técnica busca conjuntos de ítems frecuentes, es decir entidades claves que comúnmente aparecen juntas en varios documentos; descubrir conjuntos de ítems frecuentes es un problema conocido en el área de *Data Mining* como *Market Basket Association Rules Problem* [ASI93]. Para obtener los conjuntos frecuentes de ítems, TopCat utiliza una generalización del algoritmo de Association Rules llamado *Query Flocks* [TUSVMN97].

TopCat encuentra los clusters de ítems construyendo un hipergrafo con los conjuntos frecuentes de ítems y particiona este hipergrafo usando algún criterio de similaridad relacionado a la confianza de las reglas de asociación de estos conjuntos. Un hipergrafo $H = (V, E)$ consiste de un conjunto de vértices (V) y un conjunto de hiperejes (E). Un hipergrafo es una extensión de un grafo en el sentido que cada hipereje puede conectar más de dos vértices. En este modelo, el conjunto de vértices corresponde a los distintos ítems y los hiperejes corresponden a los conjuntos frecuentes de ítems. Para particionar el

hipergrafo se utiliza el algoritmo *Hmetis* [KK98] que produce k caminos particionados balanceados con k especificado por el usuario.

Una vez que obtiene los clusters de conjuntos frecuentes de ítems realizar un mapeo inverso para determinar a qué documentos corresponden los tópicos hallados. Para llevar adelante esta etapa utiliza una métrica *tfi.df* [SAB94] para generar una medida de distancia entre un documento y un tópico, luego elige el tópico más cercano para cada documento.

2.2.4 Método de Selección de Palabras Claves

Este método de selección de keywords [LK99] puede ser utilizado para sugerir términos descriptivos o etiquetas a grupos de documentos similares o clusters. Se enfoca en las distribuciones de las palabras que ocurren dentro de los grupos de documentos.

El método define la *Bondad de una Palabra Clave* (keyword) de la siguiente manera: "Un buen descriptor de un cluster caracteriza alguna propiedad excepcional del cluster en relación con el resto de la colección.". En otras palabras, para que una palabra w sea una buena keyword para un cluster de documentos C , w debe tener las siguientes propiedades:

- w debe ser prominente en C comparada con otras palabras en C .
- w debe ser prominente en C comparada con la ocurrencia de w en la colección completa.

Estos dos criterios pueden combinarse en una medida general con el propósito de ranquear las keywords,

$$G(W) = F^{clust}(w) * F^{col}(W)$$

donde F^{clust} describe la palabra w en relación con las otras palabras dentro del cluster y F^{col} relaciona la palabra w con la colección completa. F^{clust} es la frecuencia relativa de la palabra w y se define como

$$F_j^{clust} = F_j(w) = \frac{f_j(w)}{\sum_v f_j(v)}$$

donde $f_j(w)$ es el número de veces que la palabra w ocurre en el cluster j , es decir, la frecuencia de w en j , y $f_j(v)$ es la frecuencia de la palabra v en j . F^{col} mide la relación de la frecuencia de w en el cluster j contra la frecuencia que describe cuan típica es la palabra w en otras partes de la colección y se define como:

$$F_j^{col} = \frac{F_j(w)}{\sum_i F_i(w)}$$

Luego el método define la *Bondad de w en el cluster j* como:

$$G_0(w, j) = F_j(w) \frac{F_j(w)}{\sum_i F_i(w)}$$

2.2.5 Reputación de una Página sobre un Tópico

Rafiei y Mendelzon [MR00] presentaron un proceso para computar la reputación de una página para tópicos específicos e identificar aquellos tópicos para los cuales la página tiene una buena reputación. La entrada del proceso es una *url* y la salida es un conjunto ranqueado de tópicos sobre los cuales la página tiene reputación. El método define dos proporciones que relacionan una página p y un tópico t .

- La *Penetración* de una página p sobre un tópico t , $P_p(t)$ es la fracción de páginas sobre el tópico t que apuntan a p y
- el *Foco* de una página p sobre un tópico t , $F_t(p)$, es la fracción de páginas apuntando a p que referencian al tópico t .

Esto es, si $l(p,t)$ es el número de páginas que contienen t y apuntan a p , $ln(p)$ es el número de páginas que apuntan a p y $N(t)$ es el número de páginas que contienen t , luego

$$P_p(t) = \frac{l(p,t)}{N(t)} \quad \text{y} \quad F_t(p) = \frac{l(p,t)}{ln(p)}$$

Estas cantidades pueden ser interpretadas como probabilidades condicionales.

- $P_p(t)$ es la probabilidad condicional de que una página sobre el tópico t apunte a p y
- $F_t(p)$ es la probabilidad condicional de que una página que apunta a p sea sobre el tópico t .

Es importante aclarar que para el método, una página es sobre un tópico t simplemente si contiene el término o frase t .

No alcanza con utilizar estas proporciones como medida de reputación de una página sobre un tópico, ya que éstas pueden sobreestimar las reputaciones, por ello se define la *Reputación* de una página p sobre un tópico t como:

$$RM(p,t) = \frac{(P_p(t) - L(p))}{L(p)} \quad \text{o} \quad RM(p,t) = \frac{(F_t(p) - M(t))}{M(t)}$$

con

$$L(p) = \frac{ln(p)}{N_w}$$

donde N_w es el número de páginas en la Web y

$$M(t) = \frac{N(t)}{N_w}$$

El problema que surge con este método es ¿Cómo computamos los conjuntos de tópicos que tienen reputación alta para una página dada?

Una solución puede ser computar la reputación para todo término o frase que aparece en el documento, pero no siempre sirve ya que un documento puede ser una autoridad sobre un tópico cuyo término no aparece en el documento. Por ejemplo la página principal de *IBM Almaden Research Center* (www.almaden.ibm.com) tiene reputación sobre "Data Mining" y esta frase no aparece en la página principal.

Otra solución puede ser computar las reputaciones de las páginas sobre todos los términos o frases que apuntan a la página que está siendo analizada, pero acá surge otro problema: una página puede tener cientos de miles de páginas que la apuntan, por lo tanto analizar todas estas páginas sería demasiado costoso. La solución es computar las reputaciones de un conjunto acotado de páginas que apuntan a la página que está siendo analizada.

2.2.6 Centros y Autoridades

Kleinberg [RM00, KLE98, MR00] presenta el algoritmo *HITS* para solucionar el siguiente problema: dada una cadena σ determinar las páginas que tienen autoridad sobre esa cadena. Este método es inverso al anterior, pero puede ser adaptado para hallar la

categoría a la que pertenece una página. Para resolver el problema, Kleinberg realiza un análisis de la estructura de links de una colección de páginas vinculadas a través de hipervínculos de la web. Esta colección S_σ de páginas debe cumplir las siguientes propiedades:

- S_σ es relativamente pequeño.
- S_σ es rico en páginas relevantes.
- S_σ contiene muchas (o algunas) de las autoridades más fuertes.

Para hallar esta colección consulta la cadena σ en un motor de búsqueda y se queda con las t páginas con mayor ranking (el motor de búsqueda devuelve las páginas ordenadas de mayor a menor según su ranking).

El método se refiere a estas t páginas como el conjunto raíz R_σ . Este conjunto raíz satisface las propiedades anteriores. Luego utiliza un algoritmo para expandir R_σ y producir el conjunto de páginas $S_{\sigma'}$, este último se denomina *conjunto base*.

Con $S_{\sigma'}$ construye un grafo G_σ donde los nodos corresponden a las páginas y un eje entre p y q indica la presencia de un link de p a q .

Para Kleinberg existen dos tipos de páginas útiles: las autoridades y los centros.

- Una página es una *Autoridad* si contiene mucha información sobre el tópico en cuestión y
- un *Centro* o *Hub* es una página que contiene muchos links a páginas que contienen información sobre el tópico.

El problema ahora es extraer las autoridades de la colección completa de páginas a través de un análisis de los links de G_σ . Las páginas con autoridad revelan que la consulta inicial no debería tener solamente un gran grado de entrada (número de links que apuntan a ella), puesto que todas las autoridades sobre un tópico en común deberían tener un considerable solapamiento en los conjuntos de páginas que las apuntan. El principio básico dice que los centros y autoridades exhiben lo que podríamos llamar *Relación de Consolidación Mutua*: un buen centro es una página que apunta a muchas buenas autoridades y una buena autoridad es una página que es apuntada por muchos buenos centros. Claramente, si se desean identificar centros y autoridades en un subgrafo G_σ , se necesita un método para romper esta circularidad.

Kleinberg presenta un algoritmo iterativo que mantiene y actualiza pesos numéricos para cada página, es decir a cada página p le asocia un peso de autoridad x y un peso de centro y , que actualiza utilizando las fórmulas que surgen de la relación de consolidación mutua. Kleinberg expresa la relación de consolidación mutua entre centros y autoridades de la siguiente manera:

- ✓ Si una página p apunta a muchas páginas con un gran valor de x , luego debería recibir un gran valor de y
- ✓ Si p es apuntada por muchas páginas con un gran valor de y , luego debería recibir un gran valor de x .

Luego utiliza estas propiedades para definir las fórmulas que actualizan los pesos:

$$x(p) = \sum_{q:(q \rightarrow p)} y(q)$$

donde $q:(q \rightarrow p)$ indica que la página q apunta a la página p y

$$y(p) = \sum_{q:(p \rightarrow q)} x(q)$$

donde $q:(p \rightarrow q)$ indica que la página p apunta a la página q .

En cada iteración reemplaza el peso de autoridad de la página p por la suma de los pesos de centro de las páginas que apuntan a p y reemplaza el peso de centro de la página p por la suma de los pesos de autoridad de las páginas apuntadas por p .

Para obtener los pesos de centro y de autoridad, el algoritmo de Kleinberg aplica estas fórmulas en cada iteración hasta alcanzar un punto fijo.

Este método puede ser utilizado para encontrar las k páginas que apuntan a p , con las cuales podemos armar el conjunto raíz, expandirlo para encontrar el conjunto base, obtener el subgrafo inducido y finalmente computar los centros y autoridades para un conjunto de tópicos determinados. También puede ser utilizado para determinar la similitud entre páginas.

Bharat y Henzinger [BH98, DH99] descubrieron algunos casos en los cuales el algoritmo de Kleinberg no trabaja bien, por lo tanto presentan una modificación de éste algoritmo.

Para Bharat y Henzinger el algoritmo de Kleinberg tiende a fallar por tres razones:

- **Relación de Consolidación Mutua entre Host**

Algunas veces un conjunto de documentos en un host apunta a un único documento en otro host. Esto dirige el puntaje de centro de los documentos sobre el primer host y el puntaje de autoridad sobre el documento del segundo host. El caso inverso, donde hay un documento en el primer host que apunta a múltiples documentos en el segundo host, crea el mismo problema. Asumimos que el conjunto de documentos en cada host fue creado por el mismo autor u organización, estas situaciones le dan un peso indebido a la opinión de una persona.

- **Links Generados Automáticamente**

A menudo los documentos web generados por herramientas tienen links que fueron insertados por la herramienta. Por ejemplo, el sistema Hypernews que coloca artículos de noticias en páginas web, inserta automáticamente un link al site de Hypernews. En estos casos no es aplicable asumir que la opinión de un humano es representada por el link.

- **Nodos no relevantes**

A menudo encontramos que el grafo de vecinos contiene documentos que no son relevantes al tópico consultado. Si esos nodos están bien conectados, se origina un problema: las autoridades y centros más altamente rankeados tienden a no ser sobre el tópico original. Por ejemplo, si corremos el algoritmo con la consulta "jaguar and car" obtendremos páginas de diferentes fábricas de autos como autoridades top y una lista de fábricas de autos como centros top.

Para solucionar el primer problema Bharat y Henzinger dicen que si hay k links (ejes) en documentos que están en un host hacia un único documento en otro host, el peso de autoridad de cada eje debe ser $1/k$. Este peso es usado cuando se computa la autoridad de un documento en el segundo host. Y si hay l links de un simple documento en un host a un conjunto de documentos en otro host, el puntaje de centro de cada eje es $1/l$. Adicionalmente, descartan los nodos aislados del grafo y presenta una modificación del algoritmo de Kleinberg en la cual se tienen en cuenta los pesos de los ejes.

Para solucionar el problema de los nodos no relevantes presentan la definición de similitud de una consulta y un documento basándose en la normalización del coseno de las frecuencias de aparición de términos en el tópico consultado y en el documento. Con esta medida de similitud calculan el peso para cada documento y determinan según esos pesos que documentos son relevantes para el tópico y cuales no. Dean y Henzinger presentan dos algoritmos que solamente explotan la estructura de hiperlinks de la web y no examinan la información sobre el contenido o uso de las páginas.

Por último, Rafiei y Mendelzon [RM00,MR00] presentaron un método en el cual generalizan el método de centros y autoridades. El método dice que cuando un usuario está en una página p tiene dos alternativas:

- tomar un link de p ;
- aleatoriamente elegir una página q que tiene un link hacia p y tomar un link de q .

Es decir, el usuario sigue links *forward* (hacia fuera de la página p) y *backward* (dentro de la página q). Si el usuario continua el camino hacia delante, luego el número de visitas hacia delante que realice a la página es la *Reputación de Autoridad* y el número de visitas que realice hacia atrás a la página es su *Reputación de Centro*.

La reputación de autoridad de una página p sobre un tópico t depende no solo del número de páginas sobre el tópico t que apuntan a p , sino también de las reputaciones de centro de esas páginas sobre el mismo tópico t . Similarmente, la reputación de centro de una página p sobre un tópico t no sólo depende del número de páginas apuntadas por p que son sobre el tópico t sino también de las reputaciones de autoridad de estas páginas.

Rafiei y Mendelzon dan una definición de camino aleatorio y definen formalmente la reputación de autoridad de una página p sobre un tópico t como la probabilidad que el usuario que navega aleatoriamente buscando el tópico t haga una visita hacia delante a la página p y la reputación de centro de una página p sobre un tópico t haga una visita hacia atrás a la página p . Luego presentan un algoritmo para cada uno de los métodos.

El algoritmo presentado por Rafiei y Mendelzon no sólo computa las reputaciones sino también identifica los tópicos de las reputaciones. Si en el algoritmo fijamos la página p a una página específica encontraremos la reputación de todo tópico t posible, pudiendo luego reportar los tópicos con reputación más alta para categorizar la página.

En la Universidad de Toronto se desarrolló un prototipo llamado *TOPIC - TORonto Page Influence Computation* [W36] que implementa este algoritmo presentado por Rafiei y Mendelzon para calcular reputaciones de páginas web. Dada la URL de una página, el sistema busca un conjunto de páginas que la apuntan (inlinks) utilizando los motores de búsqueda *Altavista* y *Lycos*. Por cada inlink, el sistema extrae palabras y frases del abstract devuelto por el motor de búsqueda, evitando de esta manera el overhead adicional necesario para descargar la página correspondiente, asumiendo que el abstract devuelto por el motor de búsqueda es representativo del tópico de la página. El usuario puede especificar un término o frase adicional, en cuyo caso la reputación de la página se mide solamente sobre ese tópico particular.

2.3 Resumen

El método de *Clasificación con Megadocumentos* [KF00], no nos sirve para obtener los tópicos de las páginas accedidas por los usuarios de la organización porque no conocemos de antemano las categorías de los documentos, pero resulta útil para tener en cuenta al calcular el peso de las palabras en los documentos.

La metodología de *Clustering de Hipertexto en Aplicaciones de Búsqueda en la Web* [MS96], no devuelve los tópicos correspondientes a cada cluster, sólo agrupa los documentos basándose en una medida de similaridad; de todos modos para nuestro proyecto nos aporta la idea de utilizar los links salientes y entrantes como fuente adicional de información.

Topcat [CCR00, HKK97, DAHKRV97] es la base principal de nuestro proyecto, de él tomamos la idea de utilizar itemsets frecuentes y clustering, aunque el algoritmo de clustering que utilizamos no es el mismo que utiliza esta técnica.

En nuestro proyecto utilizamos la fórmula de bondad presentada en el Método de Selección de Palabras Claves [LK99] para determinar cual es el mejor concepto para los clusters correspondientes a cada usuario.

De los métodos de *Reputación de una página sobre un tópico* [MR00] y los métodos de *Centros y Autoridades* [KLE98, RM00] tomamos la idea de trabajar no sólo con el contenido textual de las páginas web sino también con un conjunto acotado de páginas que pertenecen a su entorno. Utilizamos la herramienta *TOPIC* [W36] desarrollada en la Universidad de Toronto para comparar los tópicos identificados con nuestra herramienta con los tópicos obtenidos por *TOPIC*.

CAPÍTULO 3: Marco Teórico

3.1 Introducción

El objetivo del trabajo presentado en esta tesis es categorizar los usuarios de una organización, analizando el contenido de las páginas web accedidas por los mismos. Para llevar a cabo dicho objetivo, analizamos el *log file* del servidor *Proxy* de la organización a través del cual los usuarios tienen acceso a Internet: identificamos las acciones realizadas por cada uno de los usuarios, obtenemos las páginas a las cuales accedieron, procesamos el texto de las mismas y las categorizamos semánticamente para obtener los tópicos que describen los perfiles de cada usuario.

En este capítulo presentamos las definiciones teóricas necesarias para una mejor comprensión del trabajo. Detallamos qué es un log file y los diferentes tipos y formatos que existen; más adelante presentamos el proceso de *Data Cleaning* sobre un log file. Luego definimos qué páginas web, de las accedidas por los usuarios, se analizan. A continuación, enumeramos los problemas que surgen al procesar páginas web y sus posibles soluciones (identificación de idiomas, stemming, lematización, stop words, part of speech, selección y prioridad de tags html). Finalmente presentamos un conjunto de metodologías y técnicas utilizadas para categorizar semánticamente páginas web.

3.2 Log Files

Internet se basa en el modelo *cliente-servidor*; al usuario que envía desde su browser un requerimiento a Internet se lo denomina *cliente*. La comunicación *cliente-servidor* se realiza con pares requerimiento-respuesta (request-response) y siempre es iniciada por el *cliente*. El protocolo utilizado es el protocolo HTTP que corre sobre TCP. Cuando un cliente tiene un requerimiento para un servidor de internet, primero debe establecer una conexión TCP con dicho servidor, una vez establecida la conexión el cliente podrá enviar el requerimiento y cuando el servidor haya finalizado su respuesta, se cerrará la conexión entre ambos. Este proceso se repite cada vez que un cliente desea acceder a una página web.

Cada página web accedida puede contener múltiples archivos y cada uno de ellos se convierte en un requerimiento separado para el servidor. Los servidores pueden ser configurados para registrar información sobre los requerimientos que realizan todos sus clientes y esta información se almacena en archivos denominados *log files*.

La figura 3.1 muestra una típica taxonomía de la red donde se encuentra el servidor proxy del cual analizaremos los log files [W3]:

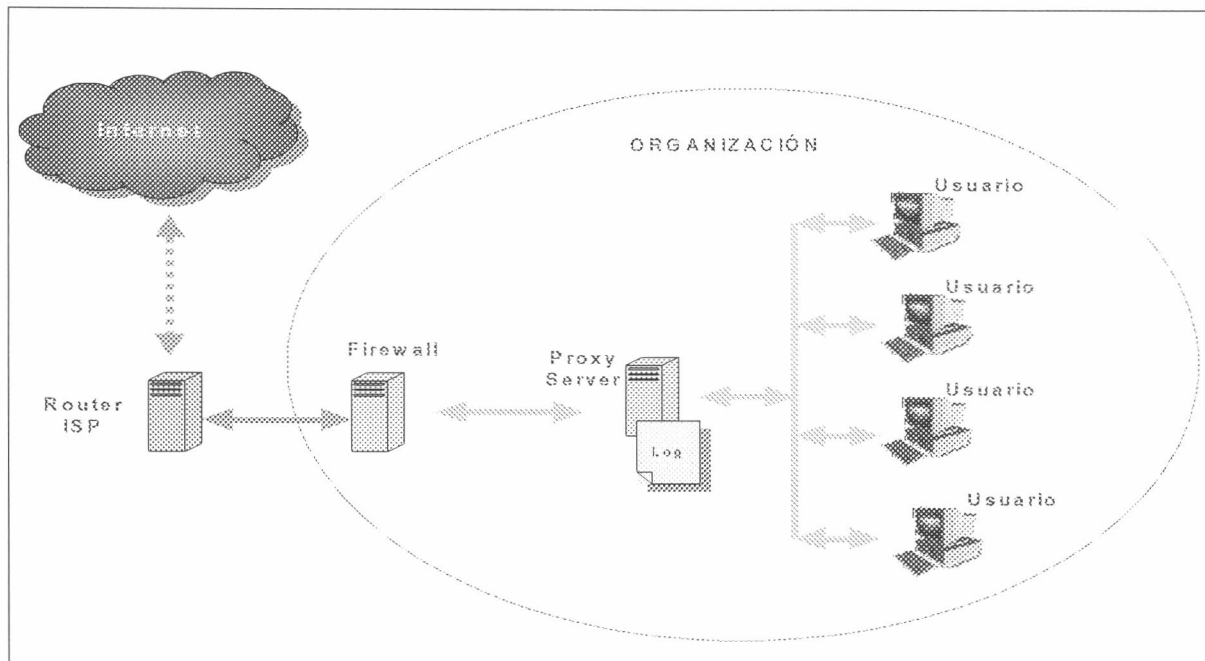


Figura 3.1: Taxonomía de la red donde se encuentra el servidor proxy

Cuando un usuario realiza un requerimiento desde su browser, éste es enviado al servidor proxy donde es registrado en el log file, luego el proxy envía el requerimiento al firewall y finalmente éste se conecta con el ISP para obtener las páginas requeridas. El firewall y el proxy podrían estar sobre la misma máquina, pero decidimos separarlos para mayor claridad.

Al tomar los logs antes de que pasen por el firewall de la organización, podemos identificar unívocamente a las máquinas desde las cuales se realizan los requerimientos por su dirección IP, ya que aún no se ha aplicado el proceso de *NAT (Network Address Translation)*. *NAT* es el mapeo de una dirección IP usada en una red por otra dirección IP de otra red. Generalmente, una organización mapea las direcciones IP de su red interna en unas pocas direcciones IP para salir a Internet y realiza el mapeo inverso cuando recibe información desde Internet, de esta manera la taxonomía de la red interna no queda expuesta. Si en cambio, tomamos los logs después de pasar por el firewall, todos los usuarios accederían a Internet utilizando unas pocas direcciones IP, por lo tanto sería necesario identificar a cada usuario de alguna otra forma y no mediante su dirección IP. Más adelante analizaremos con más detalle cómo identificar las transacciones realizadas por cada usuario independientemente de la IP o de si se autenticaron o no.

El formato de log elegido por la mayoría de los servidores proxy es el *Extended Common Log Format*, del cual hablaremos mas adelante.

3.2.1 Tipos de Log File

Todos los servidores de Internet generan log files o al menos tienen la opción de generarlos. Generalmente existen dos tipos diferentes de log file[W2,W3,W8]:

Access log

El log de accesos es un archivo de texto donde cada línea constituye un *hit*. Cada hit contiene información tal como el nombre de la URL accedida, el status del requerimiento, la fecha y hora del requerimiento y el nombre de la máquina cliente que realiza el requerimiento.

Error log

El log de errores es un archivo de texto que contiene una línea por cada error, la fecha y hora en que ocurrió el error y un mensaje que describe el error ocurrido.

Algunos servidores de Internet registran información sobre los links entre las páginas web y los documentos locales, las keywords usadas por los motores de búsqueda, el browser y el sistema operativo utilizados en la máquina cliente; esta información se denomina Referrer Information [W1] y en algunos casos es almacenada en dos log files llamados Referrer Log y Agent Log [W3]. En síntesis, los log files son archivos de texto o bases de datos como en *MS Proxy Server*, que contienen un registro histórico de la actividad en un servidor.

En el presente trabajo, utilizamos el log de accesos de un servidor proxy y de ahora en adelante lo llamaremos log file o log de accesos de manera indistinta, ya que gran parte de las referencias los utilizan como sinónimos.

3.2.2 Formatos de Log File

Los cuatro estándares de formatos de log files más populares son: *NCSA*, *W3SVC*, *Microsoft IIS 3.0* y *Website de O'Reilly*[W11]. Los demás formatos son variaciones de estos cuatro. Todos estos estándares se basan en el *Common Log Format*[W1].

Common Log Format (CLF)

El *Common Log Format* [W2,W3,W4,W5,W6,W7,W8,W13] es el formato estándar de logs de accesos en el cual sólo se registra una porción de la información contenida en los mensajes *HTTP*. Los campos que contiene son los siguientes:

Campos	
Host / IP	Es la máquina cliente que realiza el requerimiento. Si el server no puede resolver el DNS Name, coloca la dirección IP.
Ident	Es el identificador RFC 931 del usuario, si es conocido.
Authuser	Si se realiza autenticación en este campo se coloca el nombre con el cual el usuario se logueó.
Timestamp	Es la fecha y hora de realización del requerimiento.
Request	Es el requerimiento tal como lo envía el cliente. Contiene el método usado (GET, POST, HEAD, PUT or DELETE), la URL requerida y el protocolo utilizado.
Status	Es el código de estado que indica si el documento fue retornado con éxito o no.
Bytes	Es el tamaño del documento transferido en bytes.

Tabla 3.1 – Campos del Common Log Format

En el caso de que la información de algún campo no se encuentre disponible, se coloca un guión (-) en su lugar.

El formato de registro de cada línea del log file es el siguiente:

Host Ident Authuser [Timestamp] "Request" Status Bytes

En la Figura 3.2 se observa un ejemplo del *Common Log File*:

```

client - rich [09/May/1996:13:43:20 -0400] "GET / HTTP/1.0" 200 477
client - rich [09/May/1996:13:45:11 -0400] "GET /product1.html HTTP/1.0" 404
client - rich [09/May/1996:14:01:41 -0400] "GET /product.html HTTP/ 1.0" 200 204

```

Figura 3.2: Ejemplo de Common Log Format

La Figura 3.2 muestra un ejemplo en el cual un usuario que se ha autenticado como *rich*, conectándose desde el host llamado *client*, el 9 de Mayo de 1996 a las 13:43 hs. El primer requerimiento realizado fue la página home del servidor (GET/) y el documento que mide 477 bytes se obtuvo en forma exitosa(200). Dos minutos más tarde, el usuario intentó acceder a la página *product1.html* pero obtuvo el código de error 404(página no encontrada) y finalmente accedió a la página *product.html*.

Observando las tres líneas del log file podemos pensar que siempre se trata de la misma persona, pero podrían ser hits correspondientes a dos o mas personas que se autenticaron como *rich* usando la misma máquina, para poder evitar este tipo de problemas deberíamos contar con información adicional o agregar campos al log hasta convertirlo en un *Extended Common Log Format* [W3, W4, W6, W7,W9, W10,W12, W13].

Extended Common Log Format (ECLF)

```

#Version: 1.0
#Date: 12-Jan-1996 00:00:00
#Fields: time cs-method cs-uri00:34:23
GET /foo/bar.html12:21:16
GET /foo/bar.html12:45:52
GET /foo/bar.html12:57:34
GET /foo/bar.html

```

Figura 3.3: Ejemplo de Extended Common Log Format

Existen dos formatos de *ECLF*. El primer formato, posee un encabezado que especifica el tipo de datos de cada campo del log file; cada línea del encabezado se denomina *directiva* y comienza con #.

En la Figura 3.3 se muestra un ejemplo de *ECLF* en el que hay tres directivas: la versión del formato de log extendido, la fecha del requerimiento y los campos que se desean registrar; en este caso, se registra la hora en que se realizó el requerimiento, el método utilizado desde el cliente y la URI. Las siguientes cuatro líneas corresponden a requerimientos realizados por uno o varios usuarios.

El segundo formato es similar al *CLF* con algunos campos extras. En la Tabla 3.2 detallamos el formato *ECLF* del log del servidor proxy [W33] utilizado en el presente trabajo, conocido como *Web Proxy Server 2.0 Log File Format*.

Campos	
Host	Es la máquina cliente que realiza el requerimiento. Si el server no puede resolver el DNS Name, coloca la dirección IP.
Ident	Es el identificador RFC 931 del usuario.
Browser	Tipo de browser del cliente. Cuando el servidor proxy toma paginas de cache, este campo se completa con <i>MSProxy/2.0</i>
Authuser	Indica si el cliente ha sido autenticado por el servidor proxy o no. (Y/N)
Date	Fecha de realización del requerimiento

Time	Hora de realización del requerimiento
Service	Indica el servicio del proxy que esta usando el cliente
ServerName	Nombre del servidor proxy
ReferredServer	Si el proxy se encuentra en una red y éste recibe requerimientos de otros servidores, en este campo se coloca el nombre del servidor que realizó el requerimiento
Referrer	URL desde la cual se accedió a la URL actual
DestHostIP	Dirección IP del servidor destino
DestHostPort	Port TCP o UDP del servidor destino
ProcessingTime	Fecha y hora de la conexión del cliente con el proxy, en milisegundos
BytesSend	Número de bytes enviados al servidor remoto por el proxy
BytesRecvd	Número de bytes recibidos por el proxy desde el servidor remoto
Protocol	Protocolo de aplicación usado para la conexión (HTTP, FTP, Gopher, Secure)
Protocol Transport	Especifica el protocolo de transporte utilizado para la conexión (UDP o TCP)
Operation	Especifica el método de aplicación usado (GET, PUT, POST, etc)
Request	URL Destino. En CLF también contiene la operación.
MimeType	Multipurpose Internet Mail Extension del objeto requerido (Text/Plain, Image/GIF, etc)
ObjectSource	Indica el origen del objeto requerido (0 – Información no disponible, Cache – el origen es el proxy, Inet – el origen es Internet, etc)
Status	Código de estado de la transacción
Bytes	Es el tamaño del documento transferido en bytes.

Tabla 3.2: Campos del Extended Common Log Format

Cada línea del log contiene los campos de la tabla anterior en el mismo orden, con el cual han sido enunciados. Veamos un ejemplo de este formato.

```

10.4.14.15, mrodriguez, -, Y, 23/01/2001, 10:46:29, 1, -, -, www.cisco.com, -, 80, 8172, 55210, 748, http, -, -,
http://www.cisco.com/univercd/cc/td/doc/cisintwk/ics/cs012.htm, -, Inet, 200, 8388608

10.4.14.15, mrodriguez, -, Y, 23/01/2001, 10:46:29, 1, -, -, www.cisco.com, -, 80, -, 94, 417, http, -, -,
http://www.cisco.com/univercd/cc/lib/docstrip.gif, -, NotModified, 0, 409810.4.14.15, mrodriguez, -, Y, 23/01/2001, 10:46:29, 1, -, -,
www.cisco.com, -, 80, -, 94, 417, http, -, -, http://www.cisco.com/univercd/cc/lib/cisco/home.gif, -, NotModified, 0, 4098

10.4.14.15, mrodriguez, -, Y, 23/01/2001, 10:46:29, 1, -, -, www.cisco.com, -, 80, -, 94, 417, http, -, -,
http://www.cisco.com/univercd/cc/lib/cisco/toc.gif, -, NotModified, 0, 4098

10.4.14.15, mrodriguez, -, Y, 23/01/2001, 10:46:29, 1, -, -, www.cisco.com, -, 80, -, 94, 416, http, -, -,
http://www.cisco.com/univercd/cc/lib/cisco/prev.gif, -, NotModified, 0, 4098

10.4.14.15, mrodriguez, -, Y, 23/01/2001, 10:46:29, 1, -, -, www.cisco.com, -, 80, -, 94, 420, http, -, -,
http://www.cisco.com/univercd/cc/lib/cisco/gloss.gif, -, NotModified, 0, 4098

10.18.4.1, mrodriguez, -, N, 12/11/2000, 03:47:10, 1, -, -, www.musicworld.nu, -, 80, 1672, 1829, 144, http, -, -,
http://www.musicworld.nu/rock.html, -, VCache, 304, 0

```

Figura 3.4: porción de log file del proxy server

En este ejemplo (Figura 3.4), el usuario *mrodriguez* desde una PC con IP *10.4.14.15* solicitó la página *www.cisco.com/univercd/cc/td/doc/cisintwk/ics/cs012.htm*, por lo tanto en el log se registró una línea para la página html y una línea para cada una de las imágenes que la componen.

¿Porqué se registraron varias entradas en el log file para la página solicitada? Cada página que se muestra en un *browser* puede contener varias piezas separadas y cada pieza causa que se establezca una nueva conexión http, un nuevo *hit*, es decir cada una de estas líneas o entradas del log file representa una conexión http, que se utiliza para bajar un gráfico, una página de texto o un archivo. La registración en el log file de cada pieza de las páginas que se solicitan incrementa el tamaño del log, por lo tanto es necesario realizar algún tipo de depuración según la información que se desee obtener.

3.2.3 Data Cleaning de Log Files

La mayoría de las herramientas comerciales que analizan log files realizan una tarea de preprocesamiento eliminando la información irrelevante al problema que se desea resolver y aquella considerada inconsistente para evitar obtener conclusiones erróneas [KR98]; esta tarea se denomina *Data Cleaning* y es una de las etapas del proceso de *Knowledge Discovery in Databases (KDD)*. Este proceso consiste en la "extracción de conocimiento no trivial, implícito, previamente desconocido y potencialmente útil de la información disponible" [GON96].

Cuando un cliente realiza un requerimiento de una determinada página web, genera varias entradas en el log file, dado que por ejemplo, los archivos de gráficos y scripts contenidos en la página web requerida se bajan automáticamente junto con ella. Esta es una de las razones del incremento del volumen de información contenida en los log files, motivo por el cual es necesario realizar algún tipo de depuración.

Las acciones que se pueden tomar para llevar a cabo la limpieza del log file dependen fuertemente del tipo de análisis que se desee realizar. A continuación detallamos algunas de las acciones más comunes [RH00]:

- Eliminar las entradas correspondientes a archivos de gráficos, sonido o video.
- Eliminar las entradas correspondientes a requerimientos fallidos.
- Eliminar las entradas cuya cantidad de campos no coincida con la cantidad total de campos del formato del log file.
- Eliminar las entradas generadas por agentes automáticos como Web Spiders, indexadores, chequeadores de links u otros agentes inteligentes.
- Unificar las URLs.
- Una vez identificadas las transacciones de cada usuario, se pueden eliminar los requerimientos repetidos.
- Eliminar las entradas que contengan algún campo cuyo tipo no coincide con el tipo o el rango de valores no concuerde con los especificados en el formato del log file.

En el presente trabajo, se tratan las entradas del log file correspondientes a documentos *html* accedidos por los usuarios del servidor proxy y se descartan las demás entradas.

3.3 Identificación de Transacciones

Para categorizar usuarios según el contenido de las páginas web a las que acceden, es necesario identificar unívocamente a cada usuario y luego agrupar el log file en unidades lógicas correspondientes a la secuencia de requerimientos realizados por cada usuario en un lapso de tiempo. A cada una de estas unidades lógicas se la denomina *Sesión* o *Transacción*. De aquí en adelante, utilizaremos el término *Transacción* para hacer referencia a éste concepto.

La identificación unívoca de un usuario no es una tarea sencilla. Los proveedores de servicios de Internet (ISPs) utilizan servidores proxy a través de los cuales los usuarios acceden a Internet. Como consecuencia, las verdaderas direcciones IP de los usuarios no se hallan disponibles para los servidores web. Los requerimientos de diferentes usuarios que acceden a un sitio web a través del mismo servidor proxy, se registran en el log file con la misma dirección IP, la del servidor proxy. Un caso similar es el de las organizaciones comerciales, que utilizan firewalls o servidores proxy para acceder a Internet.

Para resolver el problema de las direcciones IP enmascaradas por servidores proxy y firewalls, generalmente se requiere la registración de los usuarios o la utilización de cookies entre el servidor web y el browser de los usuarios. Pero la mayoría de los usuarios desean mantener su privacidad y permanecer en el anonimato dentro de lo posible, lo que obliga a los servidores web a no requerir registración alguna ni utilizar cookies. Sin esta información disponible, es necesario realizar un detallado y profundo análisis del log file, utilizando algún tipo de heurística, como por ejemplo combinar la información de la dirección IP, el nombre de la máquina, el agente browser y la página referrer, entre otras, para identificar usuarios.

3.3.1 Modelos de Identificación de Transacciones

El objetivo en la identificación de transacciones es construir *Clusters* significativos de páginas referenciadas por cada usuario. Este objetivo puede ser llevado adelante mediante dos modelos: el utilizado por la herramienta *SpeedTracer* [W14, MCS97] al cual denominaremos *Modelo Referrer* y el utilizado por el sistema *WebMiner* [W15] denominado *Modelo Navegacional*.

Modelo Referrer

Este modelo, a partir de la información disponible en el log file, reconstruye los caminos recorridos por los usuarios para identificar las transacciones de cada usuario. Este modelo utiliza información del log file tal como la dirección IP, el *timestamp*, la URL de la página requerida, la página *referrer* y el *agente*.

A partir de la página *referrer* y la URL de la página requerida se forma un par ordenado de requerimientos que representa un paso en el camino del usuario. Cada uno de estos pares ordenados es utilizado para reconstruir el camino realizado por el usuario e identificar de esta manera las distintas transacciones. La idea básica es que la secuencia de pares de requerimientos constituye un camino conectado durante toda una transacción de usuario. La identificación de transacciones se traduce entonces, en particionar el log file en grupos, de tal forma que la secuencia de pares de requerimientos dentro de cada uno constituya un camino conectado.

Modelo Navegacional

Este modelo distingue las referencias a páginas web con propósito de *navegación* solamente, de aquellas referencias a páginas web de *contenido*. Las páginas de *navegación* son aquellas utilizadas por los usuarios en la transición para llegar a las páginas que les interesan; y las páginas de *contenido* son las páginas de interés para los usuarios.

Antes de dividir el log file en transacciones debemos identificar unívocamente a cada uno de los usuarios; para ello es necesario analizar cuidadosamente las entradas del log file para poder determinar los campos que identifican a un usuario. Luego de analizar varios log files de un servidor proxy, asumimos que los campos *IP* e *Ident*, si se informan, identifican unívocamente a un usuario y cuando no se informa el campo *Ident* todos los requerimientos realizados por la misma IP los consideramos realizados por el mismo usuario.

Modelo General de Transacciones

Sea L un conjunto de entradas en un log file. Una entrada $l \in L$ incluye la dirección IP del usuario ($l.ip$), la identificación del usuario ($l.uid$), la URL de la página accedida ($l.url$) y la fecha y hora de acceso ($l.time$).

Definición 3.1 [Transacción General]: Una transacción general t es una 3-upla

$$t = \langle ip_t, uid_t, \{(l_1^t.url, l_1^t.time), \dots, (l_m^t.url, l_m^t.time)\} \rangle$$

con $1 \leq k \leq m, l_k^t \in L, l_k^t.ip = ip_t, l_k^t.uid = uid_t$.

Esta definición asume que cada transacción se compone de todas las referencias a páginas web realizadas por un usuario. Por ejemplo, si tomamos las líneas del log de la figura 3.4 como los únicos requerimientos realizados por el usuario *mrodriguez*, la transacción general correspondiente a este usuario sería:

$t = \{10.4.14.15, mrodriguez, \{(http://www.cisco.com/univercd/cc/td/doc/cisintwk/ics/cs012.htm, 23/01/2001 10:46:29), (http://www.cisco.com/index.html, 23/01/2001 10:47:19)\}\}$

A continuación se describen 3 métodos para identificar las transacciones: *Reference Length*, *Maximal Forward Reference* y *Time Window*.

Reference Length

Este método asume que la cantidad de tiempo invertida por un usuario en una página permite clasificar a la página como de *navegación* o *contenido*. Se supone que el usuario permanece poco tiempo en las páginas de *navegación*, e invierte un tiempo mayor en las páginas de *contenido*.

Este método requiere una nueva definición de transacción en la cual se tiene en cuenta el tiempo que permanece el usuario en cada página a la que accede.

Definición 3.2 [Transacción Reference Length]: Una transacción reference length t es una 4-upla

$$t = \langle ip_t, uid_t, \{(l_1^t.url, l_1^t.time, l_1^t.length), \dots, (l_m^t.url, l_m^t.time, l_m^t.length)\} \rangle$$

con $1 \leq k \leq m, l_k^t \in L, l_k^t.ip = ip_t, l_k^t.uid = uid_t$.

Si continuamos con el ejemplo anterior, para esta definición la transacción t sería

$t = \{10.4.14.15, mrodriguez, \{(http://www.cisco.com/univercd/cc/td/doc/cisintwk/ics/cs012.htm, 23/01/2001 10:46:29, 50), (http://www.cisco.com/index.html, 23/01/2001 10:47:19, 0)\}\}$

Este tiempo, denominado *longitud de la referencia*, se estima tomando la diferencia entre la fecha y hora de la próxima referencia y la fecha y hora de la referencia actual. Obviamente, la última página referenciada no tiene una próxima referencia para estimar su longitud, por lo tanto este método asume que todas las últimas páginas referenciadas son páginas de *contenido* e ignora su longitud. Dicha suposición puede introducir errores, por ejemplo, si un usuario utiliza una página de *navegación* para abandonar un sitio web o si se produce una interrupción en la navegación tal como un corte de energía. Se debe establecer una cota de tiempo para la *longitud de referencia*.

Sea C una cota de tiempo para la longitud de referencia. Para identificar transacciones del tipo *navegación* y *contenido* se agregan 2 condiciones a la definición 3.2 [Transacción Reference Length]:

- para $1 \leq k \leq (m-1) : l_k^t.length \leq C$
- para $k = m : l_k^t.length > C$

Para identificar transacciones del tipo *contenido* solamente se agrega la siguiente condición a la definición 3.2:

- para $1 \leq k \leq (m-1) : l_k^t.length > C$

Maximal Forward Reference

Este método define cada transacción como el conjunto de páginas referenciadas en el camino realizado por un usuario desde la primer página requerida por el mismo, hasta la página anterior a una página ya existente en dicho camino. La próxima página referenciada, no incluida en el conjunto de páginas de la transacción actual, marca el comienzo de una nueva transacción. La última página de cada transacción es considerada una página de *contenido* y las demás son páginas de *navegación*.

Por ejemplo, la secuencia de páginas $A B C D C B E F E G$ podría dividirse en tres transacciones: $A B C D$, $A B E F$ y $A B E G$. Las páginas de *contenido* en este ejemplo serían D , F y G .

En este método se utiliza la definición 3.1 de transacción general.

Time Window

Este método divide el log file por usuario en intervalos de tiempo no mayores a un parámetro de entrada especificado denominado *Time Window* o *Ventana de Tiempo*. Este método asume que las transacciones significativas poseen una longitud promedio asociada.

Sea W la longitud de la ventana de tiempo. Para identificar transacciones se agrega la siguiente condición a la definición 3.1:

$$(l_m^t.time - l_1^t.time) \leq W$$

En el presente trabajo, identificamos a cada usuario unívocamente mediante la combinación de los campos Host + Ident del log file. Implementamos el Modelo Navegacional de identificación de transacciones utilizando el método Reference Length para agrupar el log file en transacciones del tipo contenido solamente, descartando aquellas páginas del tipo navegacional y agrupando todas las transacciones detectadas para un mismo usuario en una única transacción.

3.4 Preprocesamiento de Documentos Web

La mayoría de las metodologías para la obtención de los tópicos correspondientes a un documento web utilizan técnicas de *Information Extraction (IE)* [EIK99] para extraer información de los documentos que luego categorizan semánticamente.

En nuestro proyecto hemos distinguido dos etapas bien marcadas: una etapa que denominamos *Preprocesamiento de Documentos Web*, en la cual realizamos el procesamiento del texto de cada documento web obteniendo una representación de cada documento a partir de un conjunto de palabras significativas extraídas del mismo. La otra etapa que denominamos *Categorización Semántica*, consiste en el análisis de las relaciones semánticas existentes entre las palabras que representan cada documento con el fin de obtener los tópicos que conceptualizan al documento.

Las técnicas de IE que se utilizan para extraer información de los documentos web no tratan de entender el significado del texto de los documentos, sólo analizan las porciones que se especifican de los documentos web y devuelven dicha información.

Un sistema *IE* puede pensarse como un intento de convertir la información de diferentes documentos de texto en entradas de una base de datos. *IE* es la tarea de localizar información específica de un documento en lenguaje natural y es una subárea del procesamiento del lenguaje natural.

3.4.1 Information Extraction e Information Retrieval

El objetivo de Information Retrieval (*IR*) es seleccionar un conjunto de documentos relevantes de una gran colección basándose en una consulta del usuario. El usuario luego debe navegar por los documentos para obtener la información deseada. *IR* devuelve documentos relevantes de colecciones mientras que *IE* extrae información relevante de documentos escritos en un lenguaje natural. Las dos técnicas son complementarias y usadas en combinación pueden proveer poderosas herramientas para el procesamiento de texto [EIK99].

En nuestro proyecto necesitamos extraer información de los log files y de la páginas html accedidas por los usuarios, para ello debemos identificar primero que tipo de texto son, para poder aplicar las reglas de extracción de texto o patrones de extracción que utilizan los sistemas IE para identificar la información relevante a ser extraída.

3.4.2 Tipos de Documentos

Texto Libre

El texto libre es texto continuo, no formateado, no estructurado.

Texto Estructurado

El texto estructurado es definido como información textual en una base de datos o archivo siguiendo un formato estricto y predefinido, tal información puede fácilmente ser correctamente extraída usando la descripción del formato.

Texto Semiestructurado

Los datos semiestructurados son un punto intermedio entre las colecciones no estructuradas de documentos de texto y tuplas completamente estructuradas de tipos de datos. Tales textos caen entre el texto estructurado y el texto libre. Para este tipo de textos se pueden usar patrones de extracción basados en tokens y delimitadores, como por ejemplo los tags de Html o los atributos y tags de XML.

Los log files del servidor proxy que analizamos en el presente trabajo son de tipo estructurado, por lo tanto sólo necesitamos utilizar la descripción del formato para extraer la información que deseamos.

Ahora bien, ¿qué pasa con los documentos web?, ¿De qué tipo son?. Los documentos web son diferentes que los documentos tradicionales usados en *IE*; el volumen es muy grande, aparecen nuevos documentos, los documentos cambian a menudo, una gran porción de los documentos contiene texto estructurado o semiestructurado y los documentos pueden contener hiperlinks. Usualmente, los documentos web generados automáticamente son muy estructurados mientras que los generados *en forma manual* son menos estructurados.

En nuestro proyecto las reglas o patrones de extracción de información consisten en extraer el texto asociado a los tags html donde html es un lenguaje estructurado.

Antes de comenzar a analizar las páginas html que se obtienen del log file, tenemos que encontrar las repuestas a varias preguntas que van surgiendo. Por ejemplo, ¿Podemos analizar cualquier página?, ¿Debemos extraer todas las palabras contenidas en los documentos?, ¿Todas las palabras extraídas nos sirven para el análisis semántico?

3.4.3 ¿Podemos analizar cualquier página web?

La respuesta a esta pregunta depende de la información que querramos extraer de cada página. Por ejemplo, si nos interesa obtener información estadística relacionada a los tags html, como ser la frecuencia de aparición de los mismos, no nos interesa de donde provengan las páginas ni el idioma en que fueron escritas ya que todas utilizan el lenguaje html.

En cambio, si queremos relacionar semánticamente las palabras contenidas en las páginas, es inevitable identificar el idioma con el cual fueron escritas, o traducir las páginas a un idioma común, ya que la mayoría de las herramientas que existen hoy en día para realizar el este tipo de análisis trabajan con un único idioma o con un grupo de idiomas, tal es el caso de las bases de datos jerárquicas semánticas *WordNet* [W30] y *EuroWordNet* [GGV96] que se utilizan como bases de conocimiento.

3.4.4 ¿Qué es mejor: traducir los documentos o inferir su idioma?

Para traducir todos los documentos a un idioma en común, es necesario utilizar un traductor altamente confiable y rápido, estas dos características son muy importantes y dependen del traductor que se utilice. El traductor debe ser confiable, porque de no serlo existe una alta posibilidad de analizar palabras erróneamente traducidas, lo que afectaría el resultado final del análisis.

Existen muchísimos traductores on-line en la web [W17, W18, W19, W20], la gran mayoría son herramientas complementarias de los motores de búsqueda y casi todos necesitan que se les indique cual es el idioma origen y cual es el destino, por lo tanto no podemos utilizar este tipo de traductores, ya que desconocemos el idioma origen.

Los traductores que no necesitan conocer el idioma origen utilizan alguna metodología lingüística para inferirlo, la más comúnmente utilizada se basa en Ngrams (ver más adelante) [CT94, CAV94, HSU98]. Ahora bien, si los traductores utilizan esta metodología para inferir el idioma de un documento, porque no utilizarla directamente en nuestro proyecto? Decidimos utilizar esta metodología basada en Ngrams porque según Cavnar-Trenkle [CT94], “es rápida, robusta y trabaja muy bien para la clasificación de lenguajes”. Cavnar-Trenkle testearon su sistema con artículos de noticias escritos en diferentes lenguajes, de los cuales el 99.8% fue correctamente clasificado.

En este trabajo identificamos el idioma de las páginas utilizando la metodología de Ngrams, para restringir el análisis semántico sólo a aquellas páginas escritas en inglés, ya que para realizar dicho análisis utilizamos la base de datos jerárquica y semántica WordNet [W30] que es en inglés.

3.4.5 Categorización de Texto usando Estadísticas de Frecuencias de NGrams

Los Ngrams son habitualmente pares, ternas, etc. de palabras consecutivas o no. Para Cavnar y Trenkle [W21, CT94, CAV94, W22] un Ngram es una secuencia consecutiva de letras de diferentes longitudes a las cuales se les agrega un caracter blanco () al principio y al final de la cadena original.

Veamos cuales son los ngrams de longitud 2 (bi-grams), 3 (tri-grams) y 4 (quad-grams) correspondientes a la palabra “Text” (tabla 3.3):

Ngram	
Bi-grams	_ T, TE, EX, XT, T_
Tri-grams	_TE, TEX, EXT, XT_, T _ _
Quad-grams	_TEX, TEXT, EXT_, XT_ _, T _ _ _

Tabla 3.3: Ngrams de la palabra "Text"

En general, una cadena de longitud k con blancos en los extremos, tendrá $k+1$ bi-grams, $k+1$ tri-grams, $k+1$ qua-grams, etc.

Los lenguajes humanos tienen algunas palabras con mayor frecuencia de ocurrencia que otras. Una de las maneras más comunes de expresar esta idea se conoce como Ley de Zipf: "La n -ésima palabra común en un texto en lenguaje humano ocurre con una frecuencia inversamente proporcional a n ". La implicación de esta ley es que siempre hay un conjunto de palabras que predominan más que otras palabras del lenguaje en términos de frecuencia de uso y que si comparamos documentos de la misma categoría de lenguaje, éstos tendrán distribuciones de frecuencias de ngrams similares.

Veamos gráficamente en que consiste la metodología de Cavnar y Trenkle.

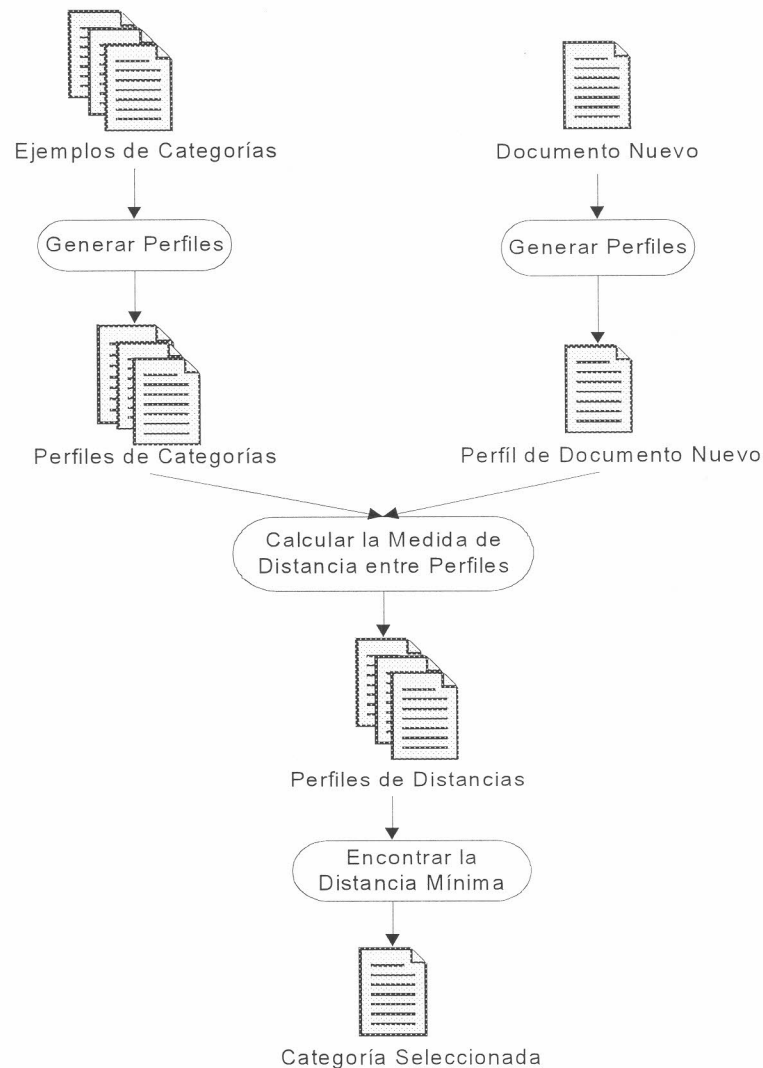


Figura 3.5: Proceso de Identificación de Idiomas con NGrams

El proceso de generación de perfiles es muy simple, consiste en leer texto, contar las ocurrencias de todos los ngrams y ordenar los ngrams y sus frecuencias en orden inverso al número de ocurrencias.

Para calcular la medida de distancia se toma uno de los perfiles de las categorías de lenguajes y el perfil del documento que se desea categorizar y se calcula una medida estadística llamada "*fuera de lugar*" por Cavnar-Trenkle para cada uno de los ngrams del documento. Esta estadística indica qué tan fuera de lugar se encuentra un ngram en el documento que se desea categorizar respecto a la categoría actual. Veamos un ejemplo del cálculo de esta estadística:

	Perfil de la Categoría		Perfil del Documento	Fuera de Lugar
Mayor Frecuencia	TH	—	TH	0
	ER	—	ING	3
	ON	—	ON	0
	LE	—	ER	2
	ING	—	AND	1
	AND	—	ED	No match = max
Menor Frecuencia	...		...	

Sum = Medida de Distancia

Figura 3.6: Ejemplo Medida de Distancia

Luego, se suman las medidas estadísticas "*fuera de lugar*" de cada ngram del documento y se obtiene la medida de distancia entre el perfil y el documento. El idioma seleccionado para el documento será el correspondiente al perfil cuya distancia con el documento sea la más pequeña.

3.4.6 ¿Debemos extraer todas las palabras contenidas en los documentos?

En las metodologías tradicionales de *IE* todas las palabras del documento son consideradas elementos a recuperar, es decir que deben ser extraídos. No ocurre lo mismo si lo que queremos es analizar el contenido de documentos html, ya que de nada nos serviría, por ejemplo, analizar sus tags html.

La idea es extraer la información contenida en los tags html, pero acá surge una nueva pregunta: ¿Debemos extraer la información de todos los tags?

La prioridad de los tags depende de la relevancia de la información que contienen, por ejemplo, por lo general el título de un documento web posee información que está muy relacionada al tópico del documento, por lo tanto en nuestro proyecto el tag *Title* tiene mayor prioridad que otros tags.

Otro ejemplo similar se da con los tags *Meta*, que se utilizan para especificar información sobre la página, como ser el autor, descripción del contenido, palabras claves que se pueden utilizar en los buscadores para hallar la página, etc. La información contenida en algunos de estos tags *Meta* puede ser muy útil para obtener el tópico de la página, pero no siempre ocurre esto, ya que algunas veces es utilizada para manipular el ranking de la

página mediante palabras o frases muy populares que no se relacionan con el contenido de la página pero que permiten que la página sea parte de la mayoría de las búsquedas realizadas por los usuarios en un motor de búsqueda.

Resumiendo, en nuestro trabajo hemos asignado un orden de prioridades a los tags html de los cuales extraemos las palabras que luego analizamos, basándonos en heurísticas obtenidas a partir del análisis de una gran cantidad de páginas web. El orden de prioridades de base definido es: TITLE, METADATA, H1, H2, H3, BIG, STRONG, B, I, EM, CITE, BLOCKQUOTE, H4, KBD, H5, H6, A, IMG

3.4.7 ¿Todas las palabras extraídas sirven para la categorización semántica?

No. Por ejemplo, dependiendo del tipo de análisis que se desee realizar sobre las palabras extraídas, es necesario realizar un preprocesamiento para distinguir las variaciones morfológicas de una misma palabra. Palabras tales como "computing" y "computation" no serán reconocidas como equivalentes sin algún tipo de procesamiento previo.

En la mayoría de los casos, las variantes morfológicas de las palabras tienen interpretaciones semánticas similares y pueden ser consideradas equivalentes.

Los sistemas actuales de IR representan el contenido de los documentos identificando las palabras y normalizándolas.

Reconocer los límites de una palabra en inglés es casi trivial: las palabras están separadas por espacios en blanco o puntuaciones. La normalización morfológica de las palabras es importante porque incrementa la medida de Recall que se utiliza para evaluar el resultado de los sistemas IR.

A continuación presentamos las técnicas de normalización morfológica de palabras más utilizadas en la actualidad: *Stemming* y *Lemmatización*.

Stemming [W23, W24, W25]

Los algoritmos de *stemming* o *stemmers* intentan reducir una palabra a su raíz o *stem*. Por ejemplo, las palabras "computing" y "computation" serán transformadas en "comput". Como puede apreciarse, "comput" no es una forma estándar de palabra o lema, por consiguiente el uso de este algoritmo dependerá del propósito del proyecto en el cual se aplique.

Los algoritmos de *stemming* aplican reglas morfológicas que reemplazan sufijos por otros en forma sucesiva hasta obtener la raíz de una palabra. El algoritmo de *stemming* comúnmente usado para el idioma Inglés es el de *Porter* [W24].

Este algoritmo considera que cualquier palabra o parte de palabra tiene la siguiente forma $[C](VC)^m[V]$ donde C es una secuencia de consonantes, V es una secuencia de vocales y m se define como la medida de cualquier palabra o parte de palabra cuando la misma se representa de esta forma.

Las reglas del algoritmo para remover sufijos se describen de la siguiente manera:

(condición) $S1 \rightarrow S2$

Esto significa que si una palabra termina con el sufijo $S1$ y el stem de $S1$ satisface la condición dada, $S1$ se reemplaza por $S2$. La condición generalmente se da en términos de m. Por ejemplo, $(m > 1)$ "ement $\rightarrow$ " donde $S1$ es *ement* y $S2$ es *null*; esta regla reemplazará *replacement* por *replac* dado que *replac* es una parte de palabra para la cual $m=2$.

La condición puede contener lo siguiente:

*s indica que el stem termina con s (similar para las otras letras).

- *v* indica que el stem contiene una vocal.
- *d indica que el stem termina con una consonante doble (por ej. -tt, -ss).
- *o indica que el stem termina en "cvc" (c es consonante y v es vocal), donde la segunda consonante no es ni w ni x ni y (por ej. -wil, -hop).

El algoritmo consta de los siguientes pasos:

1.

a)

- "sses → ss"
- "ies → i"
- "ss → ss"
- "s → "

b)

- (m>0) "eed → ee"
- (*v*) "ed → "
- (*v*) "ing → "

Si se aplica la segunda o tercer regla, se continúa aplicando las siguientes reglas:

- "at → ate"
- "bl → ble"
- "iz → ize"
- (*d and not(*l or *s or *z)) " → letra simple"
- (m=1 and *o) " → e"

c)

- (*v*) "y → i"

2.

(m>0 para todas las reglas) "ational→ate", "tional→tion", "enci→ence", "anci→ance", "iser→ize", "abli→able", "alli→al", "entli→ent", "eli→e", "ization→ize", "ation→ate", "aliti→al", "alism→al", "iveness→ive", "fulness→ful", "ousness→ous", "ivity→ive", "bility→ble", "ator→ate", "ousli→ous".

3.

(m>0 para todas las reglas) "icate→ic", "ative→", "alize→al", "iciti→ic", "ical→ic", "ful→", "ness→"

4. Elimina los siguientes sufijos de las palabras:

(m>1 para todas las reglas) "al", "ance", "ence", "er", "ic", "able", "ible", "ant", "ement", "ment", "ent", "sion", "tion", "ou", "ism", "ate", "iti", "ous", "ive", "ize", "ise"

5.

(m>1) "e → "

(m=1 and not *o) "e → "

(m>1 and *d and *l) " → letra simple"

Motores de búsqueda tales como Lycos y Google utilizan algoritmos de *stemming* para optimizar las búsquedas.

En nuestro proyecto no nos sirve utilizar este tipo de algoritmo porque las palabras que se extraen de las páginas deben ser lemas que existan en la base de datos semántica WordNet [W30], por lo tanto lo descartamos.

Lematización [W26, W27, W28]

Un algoritmo que transforma una palabra en su forma estándar o *lema* se llama *lemmatiser*. Este tipo de algoritmo también se basa en reglas morfológicas tales como:

1. Si la palabra termina en "ies", "ed" o en "s", aplicar la regla correspondiente: "ies→y", "ed→ ", "s→ "
2. Excepciones: "sheep→sheep", "feet→foot", "es→e"
3. Reducir los verbos conjugados al forma infinitiva ("sought→seek", "be, am, was, are, is, were, will→ be")

Al aplicar este algoritmo a las palabras "computing" y "computation" se obtiene como *lema* la palabra "compute".

El algoritmo de lematización presenta algunos inconvenientes que pueden afectar el resultado final del proyecto en el cual se aplica.

Por ejemplo, si consideramos la palabra inglesa "quail", que significa codorníz (sustantivo) o abatirse, deprimirse, desanimarse (verbo) y queremos obtener un mismo *lema*, probablemente no podamos distinguir los distintos significados de la palabra. La lematización del texto crea este tipo de ambigüedad, que puede solucionarse incorporando información léxica al lema. Por ejemplo, la palabra "quail" podría representarse como *quail*<N> si es un sustantivo y *quail*<V> para los verbos, donde la N (noun) y la V (verb) denotan categorías sintácticas (part of speech [W31]).

El algoritmo de lematización que utilizamos en nuestro proyecto se llama lemmatiser [W28] y se basa en una lista de 14000 lemas y 1500 excepciones que se aplican sucesivamente a las palabras a través de un conjunto de reglas. Este algoritmo sirve como herramienta para aplicaciones tales como: indexación automática, bases de datos documentales, preprocesamiento y categorizaciones semánticas de texto.

Además de normalizar las palabras, debemos seleccionar aquellas que estén relacionadas al tópico principal del documento. Para ello es necesario depurar las palabras que aparecen con mayor frecuencia, ya que éstas no nos permiten distinguir un documento de otro. También se debe determinar el sentido semántico de las palabras para poder extraer las que realmente representan al documento.

La mayoría de los sistemas de extracción de información de la actualidad utilizan *Stop Words* y *Part of Speech* [W31, W32] para llevar a cabo esta selección. En nuestro caso particular, también utilizaremos *WordNet* [W30] para continuar analizando sólo aquellas palabras que están presentes en esta base de conocimiento semántica que se utilizará en la etapa de *Categorización Semántica* para obtener los tópicos de los documentos.

Part of Speech (o Categorías Sintácticas)

Una categoría sintáctica define un conjunto de palabras de un lenguaje que muestran un comportamiento sintáctico similar. 3 categorías importantes son: sustantivos, verbos y adjetivos.

Por ejemplo, si tenemos la siguiente frase

"Many lights will light the playroom so that the light people can play"

y deseamos saber la frecuencia de aparición de cada palabra, un algoritmo que realice este cálculo sin tener en cuenta la categoría sintáctica asociada a las palabras nos dirá que la palabra *light* tiene frecuencia 3, sin embargo si tiene en cuenta la categoría sintáctica nos dirá que la palabra *light* se utiliza como sustantivo 1 vez, como verbo 1 vez y como adjetivo 1 vez.

Identificar o no categorías sintácticas depende del tipo de análisis que se desee realizar al texto. En nuestro caso particular, decidimos aplicar un proceso que le asigne a cada palabra extraída del texto de los documento html la categoría sintáctica correspondiente para evitar ambigüedades durante el análisis semántico.

Los sistemas que etiquetan cada palabra de una oración con su correspondiente categoría sintáctica se los conoce como *taggers*. Existen dos tipos de taggers: los que se basan en reglas y los que se basan en probabilidades.

El primer paso de cualquier tipo de tagger es buscar la palabra que esta siendo analizada, de aquí en más *token*, en un diccionario; si el token no se encuentra en el diccionario, el tagger debe tener un mecanismo de soporte tal como un componente morfológico o algún método heurístico que le sirva de soporte en este caso. Aquí es donde difieren los dos tipos de taggers.

El tagger basado en reglas intentará aplicar algún conocimiento lingüístico (generalmente codificado en reglas) para eliminar todas las posibilidades de combinaciones de tags ilegales, mientras que un tagger probabilístico determinará cual de las posibles secuencias es más probable, usando un modelo basado en las frecuencias de transiciones entre los diferentes tags. En otras palabras, si el token no se encuentra en el diccionario, un tagger probabilístico le asignará la categoría sintáctica más probable y uno basado en reglas le aplicará la regla que mejor se adapte al token.

El tagger que utilizamos es *QTAG* [W31, W32]. Este tagger opera con probabilidades que se adquieren a partir de un conjunto de entrenamiento.

StopWords [W29]

Cuando analizamos texto notamos que contiene muchas palabras que no se desean analizar como ser: *he, she, do, why, is, are, the, and, some, who*, etc, estas palabras se conocen como *stop words*.

La mayoría de los sistemas de *IR* utilizan *Stop Words* para eliminar estas palabras porque como tienen una gran frecuencia de aparición en los textos, no sirven para discriminar unos de otros.

Existen muchas listas de stop words conteniendo entre 100 y 400 palabras y generalmente se agregan palabras que no están relacionadas al tema de investigación. Por ejemplo, si se está analizando texto sobre deporte se pueden agregar a la lista palabras relacionadas con otros temas que no aportan información relevante al análisis, como ser, palabras relacionadas con informática.

La motivación para el uso de estas listas se basa puramente en razones de eficiencia:

- ✓ eliminarlas no afecta la calidad del sistema
- ✓ utilizarlas reduce el tamaño de la información a analizar y por consiguiente, se reduce el tiempo de análisis

WordNet (WN) [W30]

WN es una base de datos léxica que consiste de relaciones semánticas entre palabras, las cuales pueden ser accedidas como un diccionario de sinónimos.

En el presente trabajo utilizamos WN en dos ocasiones: para verificar la existencia de las palabras que representan a las páginas y para obtener los tópicos durante la categorización semántica.

Por ahora sólo diremos esto sobre WN y más adelante lo detallaremos mejor.

Como dijimos antes, nuestro sistema consta de una etapa de preprocesamiento de documentos web y otra de categorización semántica. A continuación detallamos los conceptos teóricos necesarios para la comprensión de esta última etapa.

3.5 Categorización Semántica

La etapa de *Categorización Semántica* de nuestro proyecto consiste en obtener los tópicos que generalizan las palabras extraídas de las páginas, aplicando clustering sobre los mismos y luego mapeando dichos tópicos a los usuarios que accedieron a esas páginas. Para realizar esta etapa decidimos subdivirla en 4 subetapas: *obtención de itemsets frecuentes*, *clustering de itemsets frecuentes*, *selección del tópico de mayor bondad* y *mapeo de tópicos-usuarios*. Para poder comprender cada una de ellas introduciremos algunos conceptos teóricos.

3.5.1 Reglas de Asociación (Association Rules)

La primera de estas subetapas es un problema conocido en el área de Data Mining como *Market Basket Association Rules Problem* [GON96, ASI93]. *Association Rules* consiste en descubrir patrones frecuentes, reglas de asociación y/o correlaciones entre conjuntos de ítems u objetos de una base de datos o repositorio.

En el problema del supermercado se buscan descubrir reglas sobre un conjunto de transacciones correspondientes a las compras realizadas por los clientes en dicho supermercado. Estas reglas suelen ser utilizadas por los administradores del supermercado para tomar decisiones de negocios tales como estrategias de Marketing.

Un ejemplo de tales reglas puede ser: "el 98% de las personas que compran neumáticos y accesorios de auto, realizan el service de su auto".

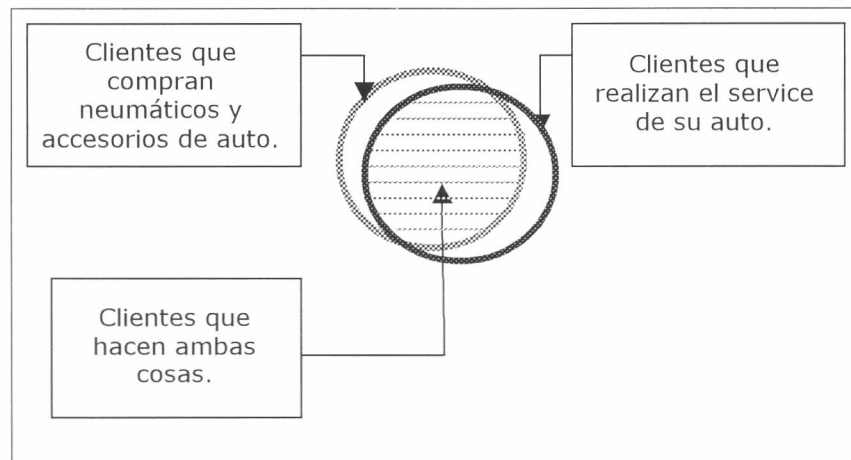


Figura 3.7: Ejemplo Association Rules

En nuestro trabajo, el problema del supermercado se traduce en encontrar conjuntos de palabras (*ítems*) que ocurren juntas con frecuencia en las páginas web accedidas por los usuarios (*transacciones*).

3.5.2 Definición Formal del Problema

Sea $I = I_1, I_2, \dots, I_m$ un conjunto de atributos binarios llamados ítems. Sea T una base de datos de transacciones. Cada transacción t es presentada como un vector binario donde,

$$t[k]=1 \text{ si en } t \text{ existe el ítem } I_k, \text{ es decir si } t \text{ compró el ítem } I_k$$

$$t[k]=0 \text{ en otro caso.}$$

Si trasladamos esta definición a nuestra categorización de usuarios, las transacciones son los documentos web que están siendo analizados y los ítems son las palabras extraídas de dichos documentos.

El problema de descubrir reglas en un conjunto de transacciones puede ser descompuesto en dos subproblemas:

1. Generar todas las combinaciones de ítems que tiene un *soporte* de transacciones por encima de una cota, llamada *soporte mínimo*. A estas combinaciones se las denomina *large itemsets* y todas las otras combinaciones que no superan la cota se las denomina *small itemsets*.
2. Para un *itemset* dado $Y = I_1, I_2, \dots, I_k$, con $k \geq 2$, generar todas las reglas (al menos k reglas) que usen los ítems de $I_1, I_2, \dots, I_k$.

Para comprender mejor estos dos subproblemas debemos definir qué es una *Regla* y cuales son las medidas de calidad que se utilizan para evaluarla: *Soporte* y *Confianza*.

Se llama Regla de Asociación (Association Rule) a una implicación de la forma: $X \rightarrow I_j$, donde X es un conjunto de algunos de los ítems de I y I_j es un ítem en I que no aparece en X .

Las medidas de calidad de una regla de asociación son:

- La **Confianza** de una regla $X \rightarrow I_j$ indica qué porcentaje de las transacciones que satisfacen X también satisfacen I_j .
Este es un concepto básico en Data Mining; por ejemplo, si X aparece en 200 transacciones y sólo en una se encuentra I_j , entonces esa regla no va aportar ningún conocimiento útil ya que se dice que la regla no es confiable.
- El **Soporte** de una regla $X \rightarrow I_j$ indica el porcentaje de transacciones del total en las cuales aparecen todos los ítems involucrados en la regla.

Siguiendo el ejemplo anterior, si X e I_j aparecen en 50 transacciones pero se cuenta con 1 millón, lo más probable es que la regla tampoco aporte información relevante.

3.5.3 Itemsets Frecuentes

El problema central es hallar los itemsets frecuentes, pero ¿Qué es un itemset frecuente? Un *Itemset frecuente* es un conjunto de ítems que aparecen con frecuencia en distintas transacciones.

- Definición 3.3 [Soporte de un Itemset]: el **Soporte** de un itemset es la cantidad de transacciones que contienen al itemset
- Definición 3.4 [Tamaño de un Itemset]: el **Tamaño** de un itemset es la cantidad de ítems del itemset.

Un itemset de tamaño k se denomina k -itemset.

3.5.4 Algoritmos de Reglas de Asociación

Los algoritmos para descubrir itemsets frecuentes hacen múltiples pasadas sobre los datos. En la primer pasada, se cuenta el soporte de los ítems individualmente y se determina cuales de ellos son frecuentes, es decir tienen mínimo soporte.

En cada pasada subsiguiente, se parte del conjunto de itemsets "semilla" descubierto en el paso anterior. Se utiliza este conjunto "semilla" para generar potenciales itemsets frecuentes, llamados itemsets candidatos, y se cuenta el soporte actual de estos itemsets candidatos durante la pasada sobre los datos.

Al final de la pasada, se determina cuales de los itemsets candidatos son frecuentes, y los mismos se convierten en "semilla" para la próxima pasada. Este proceso continúa hasta que no se encuentren nuevos itemsets frecuentes.

Con un soporte mínimo bajo se evitan desechar aquellos itemsets que ocurren en pocas transacciones por un lado, pero se generan muchos itemsets frecuentes por el otro; con lo cual se necesita establecer una relación costo-beneficio.

El algoritmo que utilizamos en nuestro proyecto se denomina *Apriori* [AS94], este algoritmo se basa en el principio de que "cualquier subconjunto de un itemset frecuente debe ser frecuente también".

3.6 WordNet

WordNet (WN) [W30] es un sistema de referencia léxica cuyo diseño está inspirado en teorías psicolinguísticas de la memoria léxica humana. Los sustantivos, verbos, adjetivos y adverbios en inglés son organizados en conjuntos de sinónimos, cada uno representando un concepto léxico.

La diferencia principal entre WN y un diccionario estándar es que WN clasifica el significado de las palabras en cuatro categorías léxicas: sustantivos, verbos, adjetivos y adverbios, que están organizadas en conjuntos de sinónimos llamados Synsets (unidad básica en WN).

La categorización sintáctica de WN introduce redundancias, palabras como "back" caen en más de una categoría. Pero la ventaja es que la organización semántica de estas categorías sintácticas puede ser claramente identificada.

Los sustantivos son organizados en una memoria léxica como una jerarquía de tópicos, los verbos son organizados con una variedad de relaciones y los adjetivos y los adverbios son organizados como hiperespacios n-dimensionales. Cada una de estas estructuras léxicas refleja una manera diferente de categorización.

La palabra "*palabra*" es comúnmente usada para referirse a la expresión y a su concepto asociado; discusiones de esta asociación léxica son vulnerables a confusión terminológica. En orden de reducir ambigüedades, utilizaremos "*forma de palabra*" para referirnos a la expresión física o inscripción y "*significado de la palabra*" para referirnos al concepto lexicalizado que una forma puede usar para expresar el significado. Luego el punto de comienzo para la semántica léxica puede decirse que es el mapeo entre las formas y los significados.

La característica más ambiciosa de WN es su intento de organizar la información léxica en términos de significados de palabras, en vez de formas de palabras. WN está organizado por relaciones semánticas. Una relación semántica es una relación entre significados de palabras y los significados pueden ser representados por Synsets.

Para acceder a WN se dispone de 4 tipos diferentes de relaciones semánticas entre synsets: *Sinonimia*, *Hiponimia/Hiperonimia*, *Antonimia* y *Merominia/Holonimia*.

3.6.1 Tipos de Relaciones

Sinonimia (Synonymy)

La *Sinonimia* o *Similitud de Significados* es una relación implícita en *WN*. De acuerdo a la definición, dos expresiones son sinónimos si la substitución de una por la otra nunca cambia el valor de verdad de la sentencia en la cual se realiza la substitución.

Hiponimia/Hiperonimia (Hyponymy/Hypernymy)

Relación *IS-A* o *ISA*, también conocido como subconjunto/superconjunto o relación *Es-Un*. Esta relación produce una organización jerárquica de los synsets para toda categoría. Es una relación semántica entre significados de palabras: por ejemplo, "árbol" es un *hyponym* de "planta" y "planta" es un *hypernym* de "árbol".

Meronomia/Holonomia (Meronymy/Holonymy)

Relación *HAS-A* o *HASA* o part-whole. Un *meronym* es una parte, un miembro o una sustancia de su *holonym*. Es la relación parte de o tiene, por ejemplo "volante" es parte de "auto" o "auto" tiene un "volante".

Antonimia (Antonymy)

WordNet considera la oposición de significados, a través de esta relación – la cual se define entre formas de palabras y no entre conjuntos de sinónimos. El antónimo de una palabra *x* es lo mismo que *no-x*, pero no siempre. Por ejemplo, rico y pobre son antónimos, pero decir que alguien no es rico no implica que deba ser pobre.

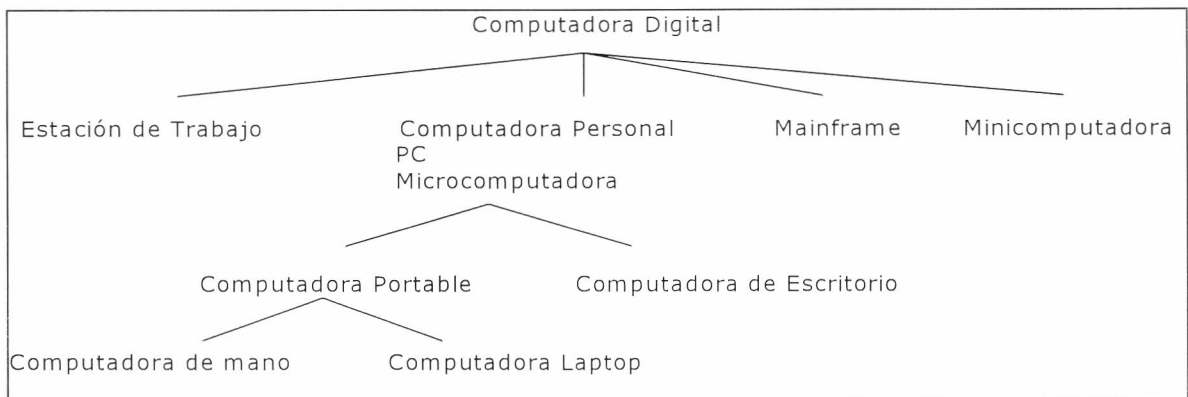


Figura 3.8: Ejemplo de jerarquía para el concepto *Computadora Digital*

En la Figura 3.8 mostramos un ejemplo de la jerarquía posible para el concepto *Computadora Digital*, utilizando la relación *Hiponimia/Hiperonimia*. De acuerdo a esta jerarquía, si en un texto se encuentran las palabras *Computadora de mano* y *Laptop*, se puede inferir que el texto trata acerca de *Computadoras portables* que es el concepto padre.

Las relaciones semánticas entre sustantivos y verbos pueden ser muy útiles para muchas aplicaciones de una base de datos semántica como *WN*. Dado un problema de encontrar todos aquellos documentos de una colección relevantes a una consulta formulada en lenguaje natural, una red semántica ofrece al menos dos ventajas evidentes:

- La posibilidad de discriminar sentidos en preguntas y documentos. Así puede, evitarse por ejemplo, encontrar documentos sobre agricultura cuando se pregunta por abonos para asistir a todas las sesiones de un festival de música.

- La oportunidad de buscar documentos que no contienen los términos originales de la pregunta. Por ejemplo, permite relacionar documentos sobre estiércol o mantillo con una pregunta sobre abono.

WN contiene aproximadamente:

- 91591 synsets con 126520 palabras o ítems léxicos, donde las palabras son definidas como claves de acceso a la base de datos.
- 95.600 formas de palabras diferentes, 70.100 significados de palabras, con lo que se cubre el rango completo del vocabulario común.
- 57000 sustantivos en 48800 conjuntos de sinónimos que se dividen en 5 archivos de tópicos.
- La jerarquía de sustantivos de WordNet tiene un promedio de profundidad de 20, bajo varios conceptos top.
- 21000 formas de palabras de verbos en 8400 conjuntos de sinónimos.

En el anexo WordNet mostramos el formato de los archivos de datos de WN.

En nuestro proyecto sólo utilizaremos la relación Hiponímia/Hiperonímia, más conocida como relación Es-Un, que produce una organización jerárquica de los synsets. En la etapa de preprocesamiento de texto verificamos la existencia en WN de todas las palabras extraídas de los documentos para asegurarnos que en la etapa de categorización semántica pueda utilizarse WN para generalizar los conceptos utilizando su taxonomía conceptual jerárquica.

3.6.2 Generalización de Tópicos

El algoritmo de generalización de tópicos que utilizamos corresponde al algoritmo de conteo de conceptos de la Tesis "Robust Automated Topic Identification"[LIN97] con algunas modificaciones. Este algoritmo accede a una taxonomía jerárquica de conceptos *WordNet* con los ítems obtenidos de las páginas web accedidas por los usuarios y obtiene los tópicos en común más bajos en la jerarquía, es decir, la generalización de los ítems. El algoritmo obtiene la generalización más apropiada de los ítems en la jerarquía semántica de *WordNet* basándose en una proporción de la frecuencia de aparición de conceptos (R) denominada *Proporción de la Rama (Branch Ratio)*.

Definición 3.5 [Branch Ratio]: sea R la proporción para un nodo tal que:

$$R = \frac{\text{MAX(Peso de todos los hijos inmediatos)}}{\text{SUM(Peso de las hojas del árbol cuya raíz es el nodo actual)}}$$

Definición 3.6 [Peso de un Ítem]: sea P el peso de un nodo tal que:

$$P = \begin{cases} \text{Frecuencia de aparición del ítem en el} \\ \text{conjunto de páginas del usuario} & \text{si el nodo es hoja} \\ \\ \text{Suma de los pesos de todas las hojas} \\ \text{cuya raíz es el nodo actual} & \text{si el nodo es intermedio} \end{cases}$$

La Definición 3.5 [Branch Ratio] de la proporción R es una manera de identificar el grado de generalización, cuanto más alta es la proporción, menor es el poder de generalización que un nodo padre tiene sobre sus hijos inmediatos.

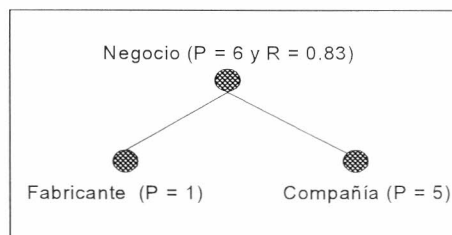


Figura 3.9: Ejemplo de proporción de rama para el concepto *Negocio*

La Figura 3.9 muestra un ejemplo del cálculo del peso y la proporción del concepto *Negocio*; este concepto tiene como hijos a *Fabricante* y a *Compañía*, con pesos 1 y 5 respectivamente.

La proporción de rama del concepto *Negocio* es $R = 5 / 6 \approx 0.83$. En este caso, se debe tomar como tópico *Compañía* porque la diferencia del peso entre *Negocio* y *Compañía* es muy chica; además *Compañía* es más específico que *Negocio* y el 83 % del peso del concepto *Negocio* viene del hijo *Compañía*.

Veamos otro ejemplo, en el cual el nodo $N1$ tiene como hijos a los nodos $N2$, $N3$, $N4$, $N5$ y $N6$, cada uno de ellos con peso 2. En este caso la proporción de $N1$ es $R = 2/10 \approx 0.20$. El tópico que deberíamos tomar en este ejemplo es $N1$, porque todos sus hijos aportan la misma información por lo tanto no podemos decidir cual de ellos es el mejor.

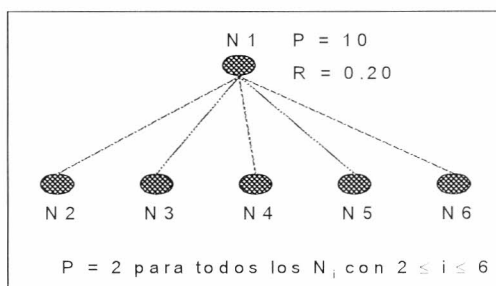


Figura 3.10: Ejemplo de proporción de la rama para el concepto $N1$

Para determinar que tópico elegir, se establece un límite de la proporción de la rama, al que denotamos R_t (*Branch Ratio Threshold*), que sirve como cota para definir el grado de generalización. Un nodo de interés es aquel cuya proporción R es menor que R_t .

Existen 2 casos especiales que se deben tener en cuenta:

- Si se tiene un padre con un solo hijo, se debe tomar como tópico el hijo, porque es más específico que el padre.
- Para computar la proporción de un nodo hoja, se agrega un nodo *dummy* conectado a cada hoja, ya que no podemos utilizar la fórmula para calcularlo. Cada nodo hoja siempre tiene proporción 1.

3.6.3 Ambigüedad Sintáctica y Semántica

El algoritmo de generalización de conceptos funciona bien cuando todo concepto tiene un único concepto padre; pero una palabra puede tener varios sentidos y seguir múltiples categorías sintácticas, es decir puede tener más de un padre en la jerarquía de generalización.

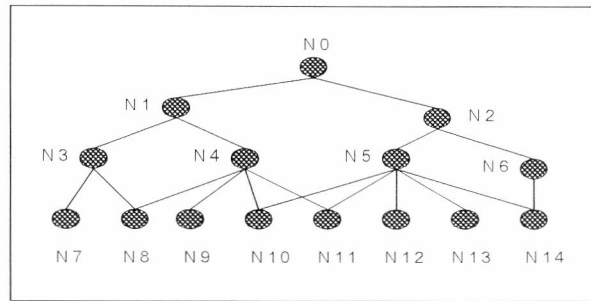


Figura 3.11: Ejemplo de una jerarquía de conceptos con ambigüedades

Existen dos problemas cuando consideramos las ambigüedades sintácticas y semánticas:

Conteo del peso de un concepto múltiples veces

Un concepto hoja puede tener múltiples padres y estos padres, a su vez, tienen un padre en común. En la Figura 3.11, los nodos hoja *N8*, *N10*, *N11* y *N14* presentan este problema.

En la siguiente tabla se muestran dos formas de calcular el peso y la proporción de un concepto. El método *Naive* calcula el peso y la proporción de un concepto basándose en los hijos inmediatos, mientras que el método denominado *Improved*, los calcula totalizando los valores de las hojas.

Nodo	Naive		Improved	
	Peso	Proporción	Peso	Proporción
N0	12	$6/12 \approx 0.5$	8	$5/8 \approx 0.62$
N1	6	$4/6 \approx 0.67$	5	$4/5 \approx 0.8$
N2	6	$5/6 \approx 0.83$	5	$5/5 \approx 1$
N3	2	$1/2 \approx 0.5$	2	$1/2 \approx 0.5$
N4	4	$1/4 \approx 0.25$	4	$1/4 \approx 0.25$
N5	5	$1/5 \approx 0.2$	5	$1/5 \approx 0.2$
N6	1	1	1	1

Tabla 3.4: Ejemplo de diferentes cálculos de proporciones

En el ejemplo de la figura 3.11, los nodos hojas *N7* a *N14* tienen peso y proporción 1. Con el método *Naive*, los nodos *N8*, *N10*, *N11* y *N14* incrementan el peso de los nodos *N0*, *N1* y *N2*. El método que utilizamos en las fórmulas enunciadas al comienzo denominado *Improved* soluciona este problema, pues sólo trabaja con los pesos de las hojas del subárbol cuya raíz es el nodo al cual se le está calculando el peso y la proporción. Como puede observarse en la tabla 3.4, los nodos con más de un padre no incrementan el peso del nodo *N0*.

Conceptos mutuamente relacionados

Supongamos que el algoritmo de generalización de conceptos obtiene como nodos de interés a los nodos *N3*, *N4*, *N5* y *N14*. En este caso los nodos *N3* y *N4* comparten al nodo *N8* y los nodos *N4* y *N5* comparten los nodos *N10* y *N11*. Para calcular el peso de *N4* y *N5* se utilizan los pesos de *N10* y *N11*, por eso decimos que existen conceptos mutuamente relacionados.

Para solucionar este problema, el algoritmo de generalización de conceptos no debe seleccionar los conceptos *N4* y *N5*, sino que debe elegir uno de ellos. *El algoritmo de generalización de conceptos que utilizamos en el presente trabajo, selecciona el nodo de mayor peso.* Con *N3* y *N4* ocurre lo mismo, el algoritmo selecciona el nodo de mayor peso.

La tabla 3.5 muestra un ejemplo de una palabra con más de un sentido sintáctico y semántico en WordNet.

Whale	
POS	Hypernym
N	Heavyweight
N	whale
V	Hunt for whale

Tabla 3.5: Lista de significados sintácticos y semánticos de la palabra *whale*

Con los itemsets frecuentes obtenidos aplicamos un algoritmo de clustering que agrupa los itemsets frecuentes basándose en una medida de similaridad que utiliza el algoritmo de generalización de tópicos.

3.7 Clustering

Clustering también conocido como *Segmentación*, es un proceso de descubrimiento de conocimiento que agrupa conjuntos de datos en clusters o grupos de tal manera que maximiza la similaridad de los datos dentro de cada cluster (intra-cluster) y minimiza la similaridad entre los datos de distintos clusters (inter-cluster).

Un *Cluster* es un conjunto de elementos, *itemsets frecuentes* en el presente trabajo, agrupados según alguna medida de similaridad definida entre ellos.

Los clusters obtenidos se utilizan para explicar las características de la distribución de los datos. Los grupos o clusters son desconocidos a priori, no existen grupos predefinidos, y el propósito del análisis de datos es determinar los clusters desde los datos mismos, de modo que los datos dentro de un mismo grupo sean más similares u homogéneos que aquellos que pertenecen a grupos diferentes.

Los clusters que se obtiene deben satisfacer dos criterios principales:

- Cada grupo o cluster debe ser homogéneo, todos los datos agrupados en un mismo cluster deben ser similares.
- Cada grupo o cluster debe ser diferente a los otros clusters, los datos que caen en un cluster deben ser diferentes a los datos que caen en otros clusters.

Dependiendo de la técnica de *Clustering* utilizada, los clusters pueden ser expresados de diferentes maneras:

- Los clusters pueden ser *exclusivos*, como los utilizados en el presente trabajo, es decir *cada dato cae en un único cluster*.
- Puede existir solapamiento, es decir un dato puede caer en más de un cluster.
- Los clusters pueden tener una estructura jerárquica que dividan los datos en clusters y subclusters.

En muchas aplicaciones de negocios se utilizan técnicas de Clustering para agrupar a los clientes y permitir a las empresas construir soluciones a la medida de cada uno de sus clientes o predecir patrones de comportamiento de los clientes basándose en el perfil de los clusters a los cuales pertenecen.

En este proyecto utilizamos el denominado Clustering Conceptual, muy utilizado en el área de Machine Learning. Este tipo de Clustering, además de generar los clusters o grupos de datos basado en alguna métrica o medida de similaridad, intenta encontrar una descripción significativa para cada cluster; para ello utilizamos el método Attribute-Oriented Induction[HM96], el cual va reemplazando los datos por una conceptualización superior en la jerarquía de conceptos WordNet [W30], que se obtiene utilizando el algoritmo de generalización de tópicos.

3.7.1 Medidas de Similaridad

Los algoritmos de Clustering utilizan medidas de similaridad o distancia entre los datos para determinar los clusters. Las medidas de similaridad más populares incluyen la distancia de *Minkowski*, la distancia de *Manhattan* y la distancia *Euclidiana*. Esta última es la más usada, pero por lo general se encuentra muy relacionada con los datos que están siendo analizados.

En nuestro trabajo definimos de manera heurística una medida de similaridad propia basada en nociones de conjuntos y conceptos semánticos.

La similaridad de palabras entre itemsets se especifica en función de los ítems que los componen. Se tiene en cuenta la cantidad de ítems resultantes de la intersección de los itemsets.

Definición 3.7 [Similaridad de palabras entre Itemsets]:

Sean dos itemsets I_i, I_j la similaridad de palabras entre I_i e I_j denotada $C(I_i, I_j)$, se define como:

$$C(I_i, I_j) = 2 * \#(I_i \cap I_j) / \#(I_i) + \#(I_j)$$

Por ejemplo, si $I_1 = \{\text{monitor}\}$ y $I_2 = \{\text{monitor}, \text{cpu}\}$ entonces $C(I_1, I_2) = 2 * 1 / (1 + 2) = 2 / 3 \approx 0.67$.

La similaridad semántica entre itemsets se establece en función de la conceptualización jerárquica de los ítems que los componen. Se tiene en cuenta la cantidad de conceptos que definen a los ítems, así como también la profundidad de los conceptos dentro de la jerarquía semántica WordNet.

Definición 3.8 [Similaridad Semántica entre Itemsets]:

Sean I_i, I_j itemsets, $GetTopics(I_i, I_j)$ el conjunto de conceptos en común entre I_i, I_j en la jerarquía semántica WordNet y $Depth(c)$ la profundidad del concepto c dentro de la jerarquía. La similaridad semántica entre I_i e I_j denotada $S(I_i, I_j)$ se define como:

$$S(I_i, I_j) = \#GetTopics(I_i, I_j) + \sum_{c \in GetTopics(I_i, I_j)} Depth(c)$$

Los conceptos o tópicos se obtienen con el algoritmo de generalización de tópicos detallado en el capítulo de implementación.

Continuando con el ejemplo anterior, los conceptos que se obtienen con el algoritmo de generalización de tópicos que accede a WordNet [W30] son:

$GetTopics(I_1, I_2) = \{\text{electronic equipment}\}$, de lo que se desprende que $S(I_1, I_2) = 1 + 5 = 6$.

La similaridad entre clusters se especifica en función de la similaridad de palabras y la similaridad semántica de todos los itemsets que los componen.

Definición 3.9 [Similaridad entre Clusters]:

Sean C_i, C_j dos clusters y $Itemset(C_i)$ el conjunto de itemsets pertenecientes a C_i . La similaridad entre C_i y C_j denotada $D(C_i, C_j)$ se define como:

$$D(C_i, C_j) = 1 - (\#(C_i) + \#(C_j)) / \sum_{m \in \text{Itemset}(C_i), n \in \text{Itemset}(C_j)} (C(m, n) + S(m, n))$$

Supongamos que tenemos dos clusters C_1 y C_2 compuestos por los itemsets I_1 y I_2 respectivamente del ejemplo anterior, entonces $D(C_1, C_2) = 1 - (1 + 1) / (C(I_1, I_2) + S(I_1, I_2)) \approx 1 - 2 / (0.67 + 6) \approx 0.7$.

Ya sabemos que fórmulas de similaridad vamos a usar al realizar clustering, sólo nos falta definir el algoritmo de clustering a aplicar.

3.7.2 Clasificación de Algoritmos de Clustering

Los algoritmos de Clustering pueden clasificarse en Algoritmos Jerárquicos y Algoritmos Particionados [FP98].

Algoritmos Jerárquicos

Los algoritmos de clustering jerárquico toman como entrada un conjunto de datos y construyen un árbol que agrupa elementos con características similares. Estos métodos permiten partir el conjunto de entrada en distintas cantidades de clusters dependiendo del nivel de abstracción deseado.

Existen dos tipos de algoritmos jerárquicos: el *Algoritmo Jerárquico Aglomerativo* y el *Algoritmo Jerárquico Divisivo*.

- *Algoritmo Jerárquico Aglomerativo*

El algoritmo jerárquico aglomerativo forma un cluster por cada dato del conjunto y luego une estos clusters atómicos en clusters cada vez más grandes hasta obtener la cantidad de clusters deseada o hasta que no se puedan agrupar más clusters. Es el algoritmo de clustering más utilizado comúnmente.

El algoritmo general consta de los siguientes pasos,

1. Encontrar los 2 objetos más similares y agruparlos en un mismo cluster.
2. Encontrar y agrupar en un mismo cluster los próximos 2 objetos más similares, donde un objeto puede ser, en este punto, un objeto individual o un cluster de objetos.
3. Si existe más de 1 cluster ir al paso 2.

Los algoritmos específicos se caracterizan por la definición de la función de similaridad utilizada para identificar el par de objetos mas similares.

- *Algoritmo Jerárquico Divisivo*

El algoritmo de clustering jerárquico divisivo forma un solo cluster con todos los datos del conjunto de muestras y luego va subdividiendo dicho cluster hasta obtener la cantidad de clusters deseada o hasta que no se pueda subdividir más.

Algoritmos Particionados

Los algoritmos particionados dividen los datos en k subconjuntos llamados clusters, de forma tal que los datos dentro de un cluster son más similares entre sí que al compararlos con los datos de los otros clusters. Cada cluster se representa por un centro de gravedad o por uno de los datos del cluster ubicado cerca del centro. k es un parámetro de entrada para este tipo de algoritmo, o sea, se requiere cierto

conocimiento del dominio. Un ejemplo de este tipo de algoritmos es el conocido algoritmo *K-means*[JD98].

En el presente trabajo aplicamos un algoritmo jerárquico y aglomerativo. Es aglomerativo en el sentido que comienza con un conjunto de itemsets frecuentes independientes y los va agrupando en clusters basados en su semejanza con los miembros de cada cluster. Es jerárquico porque los clusters resultantes quedan estructurados jerárquicamente según la jerarquía semántica de WordNet.

3.7.3 Peso de un Cluster

Una vez obtenidos los clusters de itemsets frecuentes, a cada cluster se le asigna un peso que depende de la cantidad de páginas accedidas por el usuario que caen dentro del cluster. Esto indica el nivel de importancia que tienen los tópicos de cada cluster en la categorización del usuario.

Las páginas que caen dentro de cada cluster son aquellas páginas que contienen los itemsets frecuentes del cluster.

Definición 3.10 [Peso de un Cluster]:

Sea un cluster c , el peso de c se denota $P(c)$ y se define como:

$$P(c) = \frac{CP(c)}{T}$$

donde $CP(c)$ es la cantidad de páginas accedidas por el usuario que caen dentro del cluster c y T es el total de páginas accedidas por el usuario.

Una vez obtenidos los clusters de itemsets frecuentes con su respectivo peso y la lista de tópicos asociada, falta ver como decidir cual de todos estos tópicos es el que mejor lo representa. Para ello, calculamos la bondad de cada tópico [LK99] y asignamos el tópico de mayor bondad como descriptor del cluster.

3.8 Bondad de un Tópico

Las definiciones que presentamos en esta sección se basan en la metodología presentada por Krista Lagus and Samuel Kaski [LK99], descrita en el capítulo 2 - Estado del Arte.

Para definir la bondad de un tópico en un determinado cluster debemos definir primero qué significa la frecuencia de un tópico en un cluster en nuestro trabajo.

Definición 3.11 [Frecuencia de un Tópico en un Cluster]:

Sean un cluster c y un tópico t , la frecuencia de t en c se denota $F(t,c)$ y se define como:

$$F(t,c) = \sum_{I \in c} \sum_{i \in I} P(t,i)$$

siendo

$$P(t,i) = \begin{cases} 1 & \text{si } t \text{ es padre en común de los items de } i \text{ en WordNet} \\ 0 & \text{si } t \text{ no es padre en común de los items de } i \end{cases}$$

Definición 3.12 [Frecuencia Relativa de un Tópico en un Cluster] [LK99]:

Sean c un cluster y t un tópico, la frecuencia relativa de t en c se denota $Fr(t,c)$ y se define como:

$$Fr(t,c) = \frac{F(t,c)}{\sum_{d \neq t} F(d,c)}$$

La frecuencia relativa del tópico t en el cluster c se define como la suma de las frecuencias de todos los demás tópicos ($t \neq d$) en c , dividido la frecuencia de t en c . De esta manera podemos definir la bondad de un tópico t en un cluster c como sigue.

Definición 3.13 [Bondad de un Tópico en un Cluster] [LK99]:

Sean c un cluster y t un tópico, la bondad de t en c se denota $Bondad(t,c)$ y se define como:

$$Bondad(t,c) = Fr(t,c) * \left(\frac{Fr(t,c)}{\sum_{k \in c} Fr(t,k)} \right)$$

En la etapa de análisis semántico se utilizan estas fórmulas para determinar cuál es el tópico que mejor define a cada cluster, el que se utiliza como descriptor del cluster.

3.9 Métricas de Evaluación

Para evaluar la performance de las distintas etapas del *SWMiner* utilizamos las métricas estándares de IR: *Precision* y *Recall*. En la tarea de extracción de información, *Recall* puede ser interpretado como la fracción de información relevante que ha sido obtenida y *Precision* como la fracción de información obtenida que es relevante.

Precision y *Recall* se definen de la siguiente manera [EIK99, BG94]:

Definición 3.14 [Precision]:

$$Precision = \frac{\# \text{Respuestas Correctas}}{\# \text{Respuestas Producidas}} = \frac{\# \text{Respuestas Correctas}}{\# \text{Respuestas Correctas} + \# \text{Respuestas Incorrectas}}$$

Definición 3.15 [Recall]:

$$Recall = \frac{\# \text{Respuestas Correctas}}{\# \text{Total Posibles Correctas}} = \frac{\# \text{Respuestas Correctas}}{\# \text{Respuestas Correctas} + \# \text{Respuestas Correctas No Obtenidas}}$$

Las Figuras 3.12 y 3.13 muestran las interpretaciones gráficas de estas fórmulas.

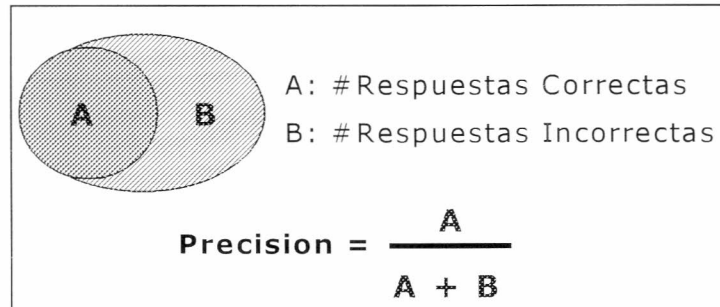


Figura 3.12: Precision

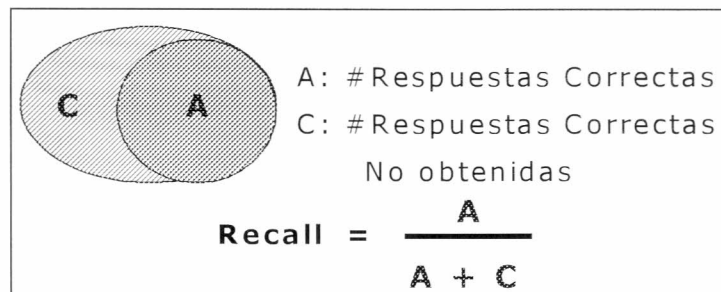


Figura 3.13: Recall

Para concluir si la performance del sistema es buena o mala debemos ver que tan cercanas a la unidad son estas medidas. Cuanto más se acercan a la unidad, mejor es la performance del sistema.

3.9.1 Problemas de Precision y Recall

Recall es difícil de estimar, pues no es fácil conocer cual es el conjunto completo de respuestas correctas del total de repuestas. En cambio *Precision* es un poco mas fácil de estimar, aunque precisar cuándo una respuesta es correcta o no depende de la persona que está evaluando el sistema, ya que una respuesta puede ser correcta para una persona e incorrecta para otra.

3.9.2 Definiciones de Precision y Recall para SWMiner

A continuación presentamos las diferentes definiciones de "Respuesta Correcta", "Respuesta Incorrecta" y "Respuesta Correctas No Obtenidas" que utilizamos para medir la performance del SWMiner en general y en cada una de sus etapas.

De aquí en adelante llamaremos *Hits* a las "Respuestas Correctas", *Mistakes* a las "Respuestas Incorrectas" y *Misses* a las "Respuestas Correctas No Obtenidas", por lo tanto las fórmulas de *Precision* y *Recall* se redefinen de la siguiente manera:

Definición 3.16 [Precision]:

$$\text{Precision} = \frac{\text{Hits}}{\text{Hits} + \text{Mistakes}}$$

Definición 3.17 [Recall]:

$$\text{Recall} = \frac{\text{Hits}}{\text{Hits} + \text{Misses}}$$

Definición 3.18 [Hits, Misses y Mistakes para Identificación de idiomas]:

- Hits: cantidad de páginas en inglés identificadas por el algoritmo.
- Misses: cantidad de paginas en inglés no identificadas por el algoritmo.
- Mistakes: cantidad de páginas en otro idioma identificadas en inglés por el algoritmo.

Definición 3.19 [Hits, Misses y Mistakes para Lemmatiser]:

- Hits: cantidad de lemas identificados por el algoritmo.
- Misses: cantidad de lemas no identificados por el algoritmo.

- Mistakes: cantidad de lemas erróneamente identificados por el algoritmo.

Definición 3.20 [Hits, Misses y Mistakes para Part Of Speech]:

- Hits: cantidad de sustantivos identificados por el algoritmo.
- Misses: cantidad de sustantivos no identificados por el algoritmo.
- Mistakes: cantidad de palabras identificadas erróneamente como sustantivos.

Definición 3.21 [Hits, Misses y Mistakes para Identificación de Tópicos]:

- Hits: cantidad de tópicos identificados por el sistema.
- Misses: cantidad de tópicos no identificados por el sistema.
- Mistakes: cantidad de tópicos erróneamente identificados por el sistema.

3.10 Resumen

En este capítulo hemos presentado los conceptos teóricos necesarios para la comprensión de nuestro proyecto. Describimos el proceso para determinar las transacciones realizadas por cada usuario, agrupando los requerimientos realizados por cada uno de ellos. Explicamos qué páginas analizamos junto con las páginas accedidas por los usuarios, qué información se puede extraer de ellas y cómo utilizar dicha información para categorizar las páginas. Luego describimos brevemente el funcionamiento del algoritmo *Apriori*, el cual utilizamos en *SWMiner* para obtener los itemsets frecuentes de palabras y los distintos tipos de algoritmos de clustering que se pueden utilizar para agrupar dichos itemsets. Describimos qué es WordNet, cómo se compone y cómo la utilizamos en *SWMiner*. Definimos los conceptos de similaridad de palabras y similaridad semántica utilizadas en *SWMiner* para obtener los clusters de itemsets y la técnica utilizada para generalizar las palabras contenidas en los itemsets accediendo a WordNet. Finalmente especificamos las métricas que utilizamos para medir la performance y eficacia de *SWMiner*.

CAPÍTULO 4: Arquitectura de SWMiner

4.1 Introducción

En este capítulo presentamos el proceso general de SWMiner; este proceso está compuesto por tres módulos: *Depurador*, *Preprocesador* y *Categorizador*.

El *Depurador* es el encargado de realizar el *Data Cleaning* sobre el log file del servidor proxy. Por ejemplo, una de sus responsabilidades es eliminar todas las entradas del log que contengan nombres de archivos con extensiones gif o jpeg, ya que en este caso, las imágenes no aportan información adicional (a menos que se haga un análisis de su contenido, lo cual escapa a los alcances de esta Tesis).

El *Preprocesador* se encarga de acceder a Internet y obtener las páginas a las cuales accedieron los usuarios; luego obtiene de ellas todas las palabras que las componen y les aplica un preprocesamiento hasta obtener las palabras representativas de cada una de estas páginas.

El *Categorizador* aplica los algoritmos de *Association Rules* y *Clustering* a las palabras relevantes obtenidas en el paso anterior y generaliza conceptos accediendo a la base de datos jerárquica semántica WordNet [W30] para obtener los tópicos correspondientes. Finalmente realiza un mapeo inverso para asignar estos tópicos a los usuarios.

En la primera sección de este capítulo presentamos el proceso general del sistema y en las secciones siguientes detallamos cada subproceso por separado.

4.2 Proceso General de SWMiner

La Figura 4.1 describe el proceso general de *SWMiner*, mostrando la interacción de las distintas etapas.

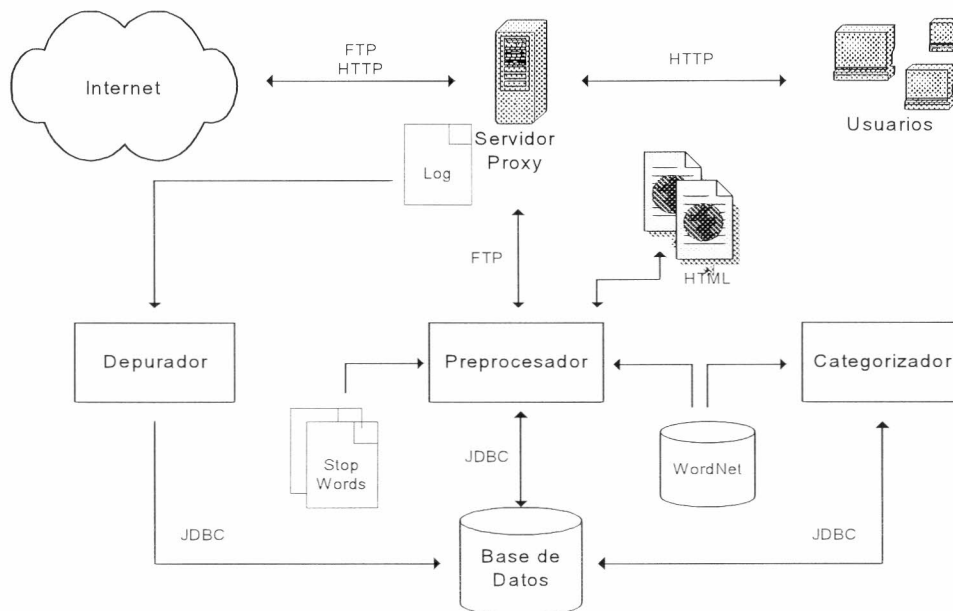


Figura 4.1: Proceso General del SWMiner

Los empleados de la organización, es decir los usuarios de *SWMiner*, acceden a Internet a través del servidor proxy utilizando un browser de internet para obtener las páginas por las cuales desean navegar. Cada uno de los requerimientos realizados por los usuarios es almacenado en el log file del servidor proxy, junto con cada elemento que compone la página requerida.

El depurador lee el log file, lo depura, determina las transacciones realizadas por cada usuario y almacena los requerimientos y los usuarios en la base de datos.

El preprocesador accede a la base de datos y por cada usuario obtiene los respectivos requerimientos, realiza el ftp de cada uno de ellos y actualiza la base de datos con la información obtenida (si pudo o no realizar el ftp de cada requerimiento, la ubicación y nombre del archivo correspondiente a cada requerimiento, etc). Luego identifica el idioma de las páginas html y continúa analizando sólo aquellas páginas escritas en inglés. Por cada una de estas páginas, aplica técnicas de procesamiento del lenguaje natural para identificar y extraer los sustantivos existentes en WordNet, de los cuales selecciona los más relevantes, según un criterio definido, para almacenar en la base de datos.

El categorizador accede a la base de datos y por cada usuario aplica el algoritmo de association rules para obtener los itemsets frecuentes sobre todos los conjuntos de palabras correspondientes a los requerimientos del usuario; almacena estos itemsets frecuentes en la base de datos y luego los agrupa en clusters según la similaridad de palabras y la similaridad semántica definida, obteniendo para cada cluster una conceptualización semántica de WordNet. Finalmente, las categorías que asigna a cada usuario son los tópicos resultantes de esta conceptualización semántica del conjunto de clusters al cual pertenecen las páginas accedidas por el usuario.

4.2.1 Depurador

La Figura 4.2 describe el proceso seguido por el Depurador.

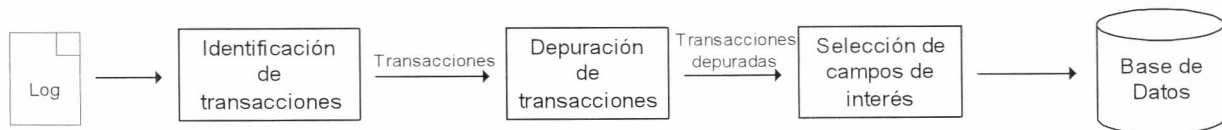


Figura 4.2: Depurador

El depurador lee el log file del servidor proxy y agrupa los requerimientos por usuario identificando las transacciones realizadas por éstos.

Las transacciones son depuradas hasta obtener sólo los requerimientos de interés, en nuestro caso los requerimientos que contienen las urls de las páginas html accedidas por el usuario, descartando el resto de los requerimientos del log como se especifica en la sección 3.2.3; por ejemplo las entradas correspondientes a archivos gráficos, los requerimientos fallidos, etc.

De los requerimientos obtenidos se extraen los campos que el *SWMiner* necesita para categorizar a los usuarios y se los almacena en la base de datos.

4.2.2 Preprocesador

El preprocesador consta de varias etapas, que describimos en figuras diferentes para su mejor comprensión. Veamos primero una breve descripción de cada una de estas etapas.

Etapas I: Distinción de Tipo de Página

Lo primero que hace es recorrer los requerimientos realizados por cada usuario para determinar si son de contenido o navegacionales. Descarta los requerimientos navegacionales y continúa analizando los requerimientos de contenido, ya que asumimos que los requerimientos navegacionales sólo se utilizaron para llegar a los requerimientos de interés.

Etapas II: Identificación de Idiomas

Esta etapa se encarga de obtener vía ftp las páginas html de los requerimientos identificados como de contenido de cada usuario e identificar el idioma de cada una de ellas, continuando sólo con los requerimientos de las páginas escritas en inglés. Esta restricción tiene su origen en la utilización de la base de datos jerárquica WordNet [W30] en el categorizador; WordNet es una base de datos léxica que consiste de relaciones semánticas entre palabras en inglés.

Etapas III: Obtención de Páginas Vecinas

Esta etapa obtiene vía ftp la vecindad o conjunto acotado de páginas escritas en inglés que rodean a cada requerimiento.

Para cada requerimiento se obtienen las k páginas que apuntan al requerimiento (inlinks) y las m páginas que son apuntadas por el requerimiento (outlinks). Este proceso se realiza n niveles hacia delante y hacia atrás, en donde por cada inlink obtenido en cada nivel se obtienen las t páginas que lo apuntan y por cada outlink se obtienen las h páginas que son apuntadas por él. Los valores k , m , t , h y n son parámetros del sistema.

Por cada página obtenida se identifica el idioma y si la misma está escrita en inglés se almacena el link correspondiente en la base de datos; caso contrario se descarta y se busca otra página.

Etapas IV: Representación de Páginas

Para cada una de las páginas html obtenidas vía ftp correspondientes a los requerimientos y sus vecindades, se aplican técnicas de procesamiento del lenguaje natural para identificar y extraer los sustantivos que no pertenezcan a las listas de StopWords definidas y que figuren en la base de datos WordNet (Part of Speech [W31, W32], StopWords [W29], Lemmatización [W26, W27, W28] y WordNet [W30]), de entre los cuales se extrae un conjunto acotado de palabras que representan a la página y que se almacena en la base de datos; dicho conjunto queda formado por aquellas palabras más relevantes según un criterio definido que se basa en prioridades en función de la posición que ocupan las palabras dentro de la estructura del archivo html.

4.2.2.1 Preprocesador - Distinción de Tipo de Página

La Figura 4.3 describe el proceso de la Etapa I del Preprocesador.

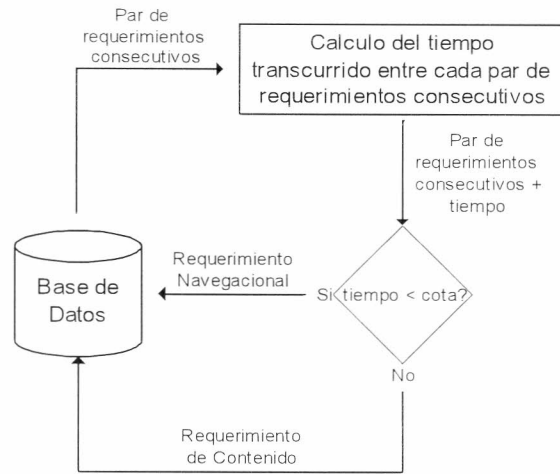


Figura 4.3: Etapa I del Preprocesador - Distinción de Tipo de Página

El preprocesador lee los pares de requerimientos consecutivos y calcula el tiempo transcurrido entre ellos basándose en la fecha y hora de realización del requerimiento por parte del usuario.

Si el tiempo transcurrido entre los requerimientos es menor que una cota de tiempo especificada en el sistema, se actualiza el tipo del primer requerimiento del par como *Navegacional* y si el tiempo transcurrido es mayor o igual a la cota especificada el tipo del requerimiento se actualiza como de *Contenido*.

4.2.2.2 Preprocesador -Identificación de Idiomas

La Figura 4.4 describe el proceso de la Etapa II del Preprocesador.

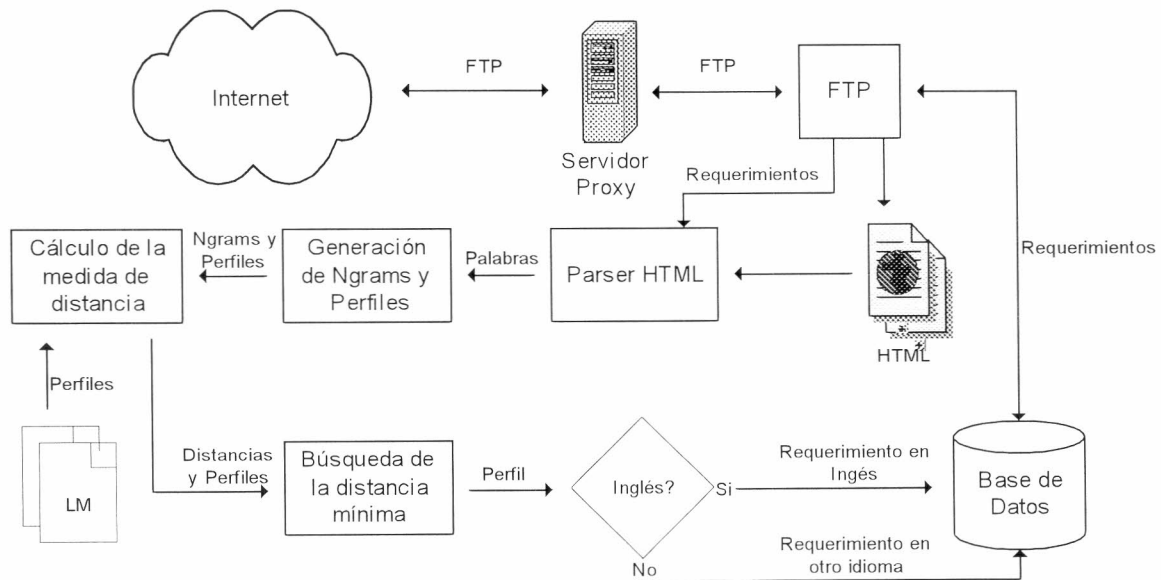


Figura 4.4: Etapa II del Preprocesador - Identificación de Idiomas

El primer paso es tomar los requerimientos de tipo de *Contenido* de la base de datos y obtener vía ftp los documentos html correspondientes.

Los documentos html que se obtienen correctamente, se parsean y de ellos se extraen las palabras contenidas en los tags html.

Con 10 de las palabras obtenidas se generan los ngrams y se los ordena según su frecuencia de aparición en los documentos de mayor a menor para construir los perfiles de los documentos html que están siendo analizados.

Con los perfiles obtenidos se calcula la medida de distancia utilizando para ello los perfiles de los diferentes idiomas disponibles. Contamos con 76 perfiles de idiomas diferentes.

De las medidas de distancia calculadas se obtiene el perfil con la menor distancia. Si el idioma correspondiente a dicho perfil es *Inglés* se actualiza el idioma del requerimiento con *EN*, de lo contrario se actualiza con *OT*.

4.2.2.3 Preprocesador – Obtención de Páginas Vecinas

La figura 4.5 describe el proceso de la Etapa III del Preprocesador.

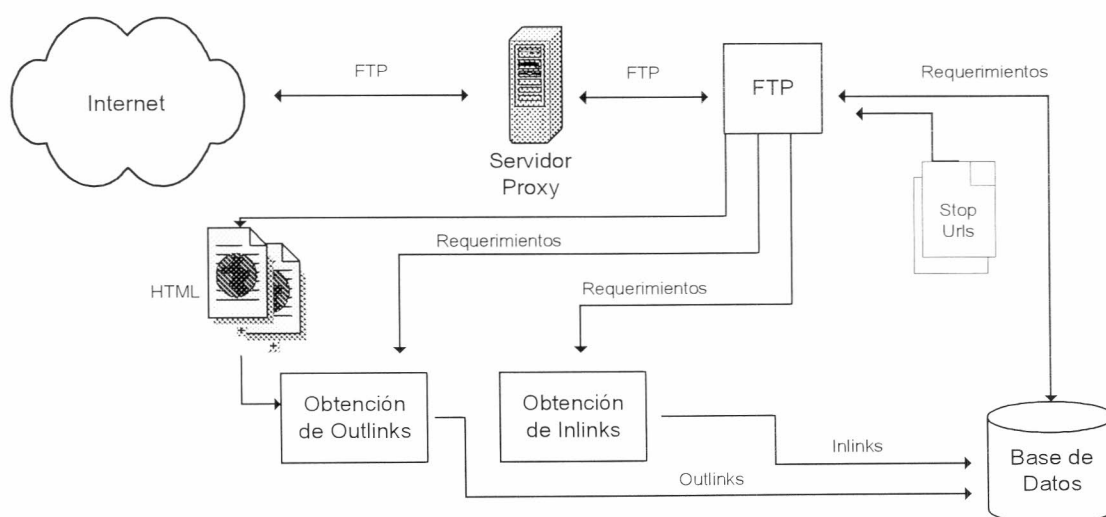


Figura 4.5: Etapa III del Preprocesador – Obtención de Páginas Vecinas

Se obtienen todos los requerimientos realizados por los usuarios y para cada uno de ellos, se obtiene una vecindad o conjunto acotado de páginas que apuntan a estos requerimientos, de aquí en más inlinks.

Para obtener este conjunto de páginas se invoca a un motor de búsqueda indicándole la url de la página a la que deben apuntar las páginas resultantes de la búsqueda. Los motores de búsqueda *Altavista* y *Google* utilizan el comando *link:url* para obtener páginas que apuntan a la url indicada.

Para cada requerimiento se obtienen los k inlinks que apuntan al requerimiento. Este proceso se realiza n niveles hacia atrás, en donde por cada inlink obtenido en cada nivel se obtienen las t páginas que lo apuntan. Los valores k , t y n son parámetros del sistema.

Por cada inlink obtenido se identifica el idioma y si el mismo es inglés se almacena en la base de datos; caso contrario se descarta y se busca otro inlink.

La figura 4.6 muestra un ejemplo en el cual se obtienen 3 inlinks, *Link1*, *Link2* y *Link3*, de un requerimiento *Req1* y 2 inlinks para cada uno de ellos, *Link4*, *Link5*, *Link6*, *Link7*, *Link8* y *Link9*. Es decir se llega hasta el nivel 2.

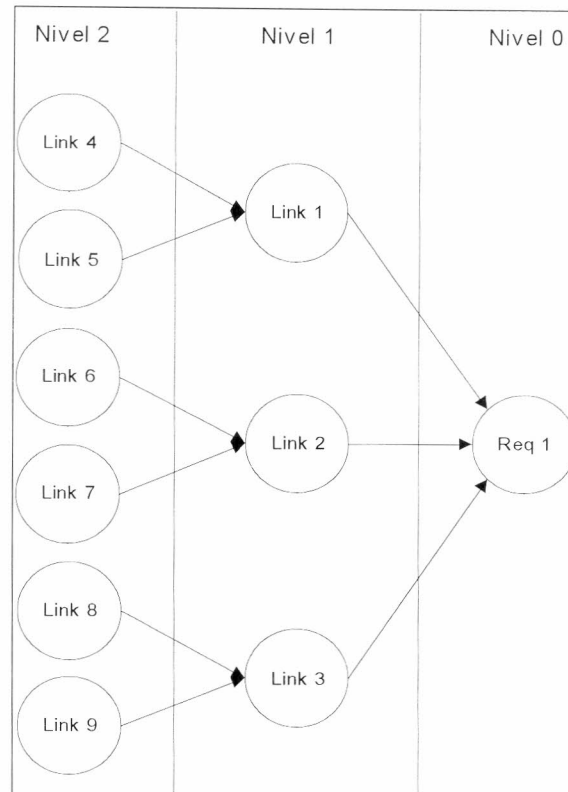


Figura 4.6: Ejemplo Inlinks

Se determinan los links contenidos en las páginas html correspondientes a los requerimientos y con la url de estos links se buscan las páginas html referenciadas. A estos links los llamamos outlinks.

Se descartan las urls que pertenecen a la lista de *StopUrls* o que contengan alguna palabra de la lista de *StopWordUrl*. Por ejemplo, todas las urls que contengan la palabra *yahoo* son descartadas porque no nos interesa analizar páginas correspondientes a un motor de búsquedas. La lista de urls *StopUrls* contiene urls que no deseamos analizar, como por ejemplo *www.hotmail.com*.

Inicialmente se buscan las páginas html referenciadas que corresponden a un dominio diferente del dominio del requerimiento o link que esta siendo analizado, pero si la cantidad de páginas obtenidas no alcanza el total de páginas indicada por el usuario como parámetro del sistema, sigue buscando páginas html del mismo dominio del requerimiento o link. Esto sucede tanto con los inlinks como con los outlinks.

Por cada outlink obtenido se identifica el idioma y si el mismo es inglés se almacena en la base de datos; caso contrario se descarta y se busca otro outlink.

Para cada requerimiento se obtienen los m outlinks que son apuntados por el requerimiento. Al igual que con los inlinks, este proceso se realiza n niveles pero hacia delante en este caso, en donde por cada outlink obtenido en cada nivel se obtienen las h páginas que son apuntadas por él. Los valores m , h y n son parámetros del sistema.

La figura 4.7 muestra el ejemplo del requerimiento *Req1*, el cual tiene 2 outlinks, *Link10* y *Link 11*, y cada uno de ellos posee 2 outlinks, *Link12*, *Link 13*, *Link 14* y *Link15*. Se llega hasta el nivel 2.

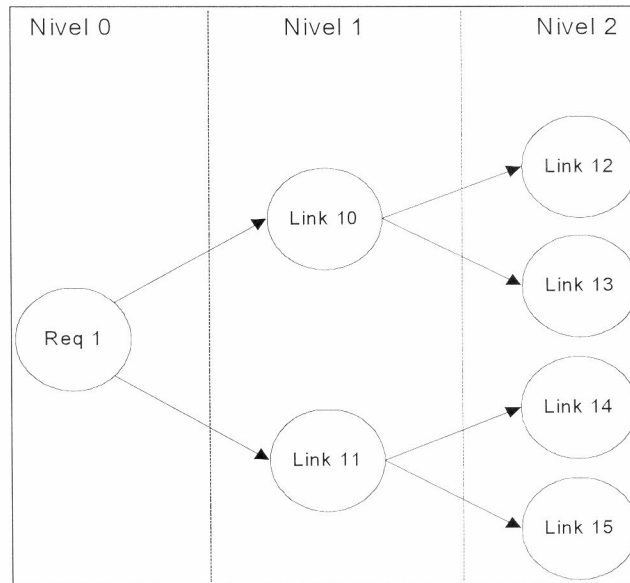


Figura 4.7: Ejemplo Outlinks

Finalmente, el grafo completo de inlinks y outlinks para el ejemplo del requerimiento *Req1* se muestra en la figura 4.8.

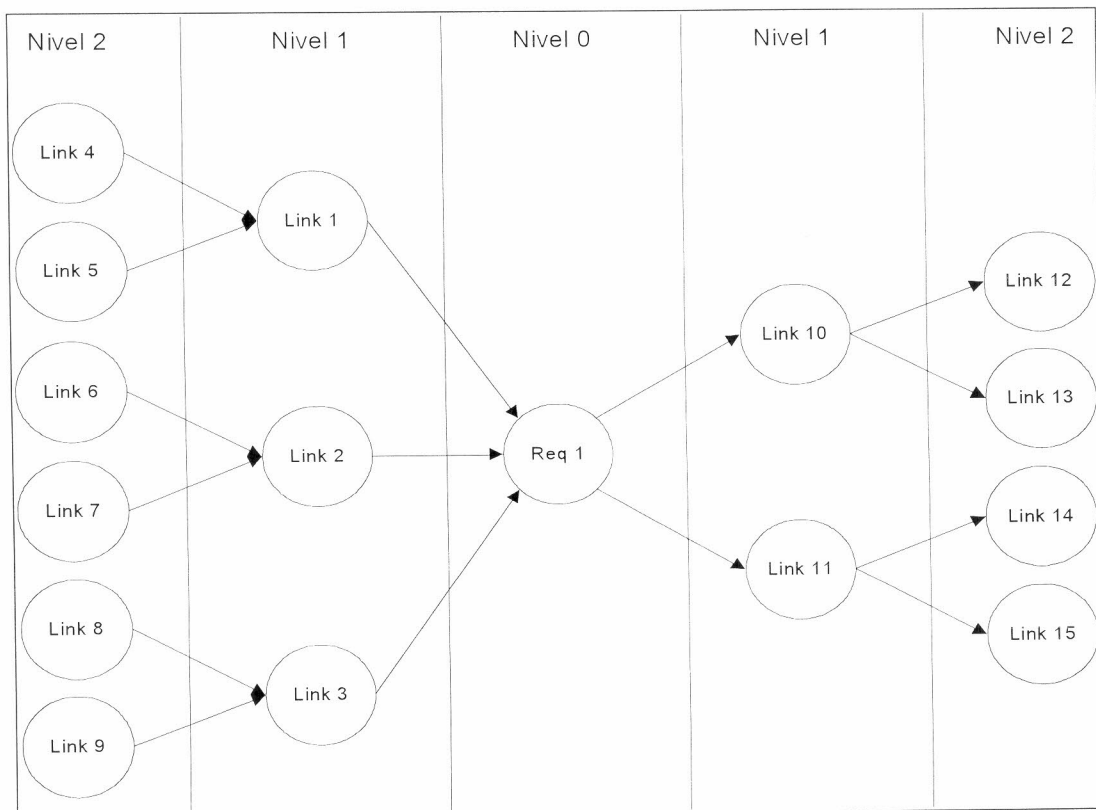


Figura 4.8: Ejemplo inlinks y outlinks

Este proceso se realiza para todos los requerimientos de todos los usuarios que están siendo analizados.

4.2.2.4 Preprocesador - Representación de Páginas

La Figura 4.9 describe el proceso de la Etapa IV del Preprocesador.

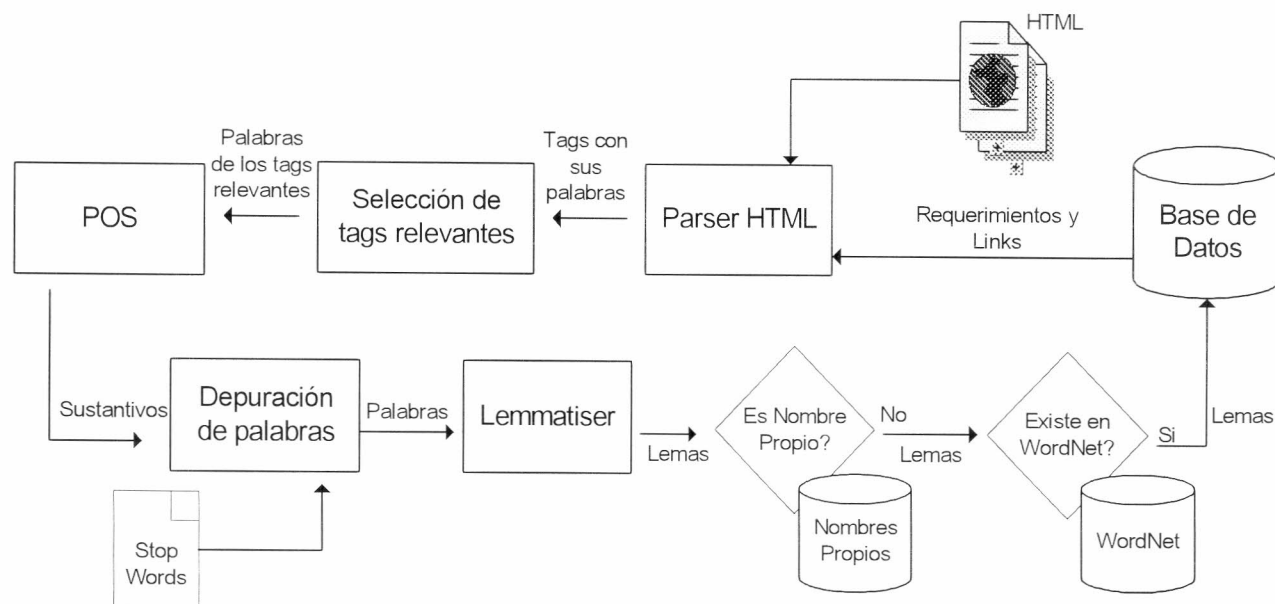


Figura 4.9: Etapa IV del Preprocesador – Representación de Páginas

Se obtienen los requerimientos y links de la base de datos y se accede a los correspondientes archivos html que son procesados por un parser.

Del parser se obtienen los tags html que componen cada archivo junto con las palabras contenidas en los tags y se realiza un proceso de selección de los mismos.

Cabe destacar que después de analizar gran cantidad de páginas html en forma manual, hemos llegado a la conclusión de que no todos los tags tienen la misma prioridad para obtener el tópico de las páginas. Por ejemplo, el tag *Title* de una página está muy relacionado con el contenido de la misma en la mayoría de los casos, por lo tanto las palabras contenidas en dicho tag se pueden utilizar para inferir el tópico. Por esta razón decidimos definir una prioridad entre tags y obtener las palabras que estén contenidas en ellos de acuerdo a esta prioridad.

De los tags seleccionados obtenemos sus palabras y les aplicamos un proceso de depuración para eliminar los caracteres especiales y los números, ya que éstos no aportan información relevante para el tipo de análisis que estamos realizando; y se eliminan todas las palabras contenidas en el archivo *StopWords* [W29]. Este archivo contiene palabras comunes que se repiten con gran frecuencia en el idioma inglés y que no aportan información relevante al análisis.

El proceso *POS* (*Part of Speech*) obtiene la categoría sintáctica asociada a cada palabra según el texto original y se queda sólo con los sustantivos. Es decir, analiza las oraciones del texto original y determina para cada palabra su categoría sintáctica (*V para verbos, N para sustantivos, etc.*) para poder luego seleccionar los sustantivos.

La importancia de conocer la categoría sintáctica asociada a cada palabra que se va a analizar radica en poder distinguir el significado de cada una de ellas. En nuestro proyecto,

sólo analizamos sustantivos debido a que la jerarquía de WordNet esta completa para los mismos.

A las palabras resultantes de los procesos anteriores se les aplica el proceso *Lemmatiser* en el cual se obtienen los lemas de las palabras.

Recordemos con algunos ejemplos a qué nos referimos cuando hablamos de lemas de las palabras. Si tenemos las palabras: "computers", "churches", "goes", "babies", "knives", "eating" y "men" después de aplicar este proceso obtendremos "computer", "church", "go", "baby", "knife", "eat" y "man" respectivamente.

Una vez obtenidos los lemas de las palabras se descartan aquellos que son nombres propios y se verifica la existencia de los mismos en *WordNet* [W30], y se almacenan en la base de datos aquellas palabras que existen en *WordNet*.

4.2.3 Categorizador

El objetivo central del *Categorizador* es identificar los conjuntos de palabras que ocurren juntas con frecuencia en las páginas web accedidas por cada usuario y agrupar dichos conjuntos en clusters de conceptos o tópicos utilizando la jerarquía semántica y conceptual *WordNet*, identificando de esta forma los tópicos que caracterizan a cada usuario.

El categorizador consta de varias etapas, que se repiten por cada usuario a analizar. Veamos una breve descripción de cada una de estas etapas.

Etapas I: Obtención de Itemsets Frecuentes

En esta primer etapa agrupa juntas aquellas palabras (ítems) que ocurren juntas con una determinada frecuencia en el conjunto de páginas web accedidas por el usuario y sus vecindades, empleando el algoritmo *Apriori*.

Etapas II: Clustering de Itemsets Frecuentes

En esta etapa trabaja sobre los itemsets frecuentes obtenidos en el paso anterior, agrupándolos en clusters de conjuntos de palabras frecuentes según una medida de similaridad definida en términos de cantidad de palabras y relaciones semánticas entre las mismas, empleando un algoritmo de *Clustering Conceptual* del tipo *Aglomerativo* y *Jerárquico* [HM96].

Cada cluster se define por el conjunto de tópicos resultantes de la conceptualización jerárquica que se obtiene al acceder a *WordNet* con los ítems que pertenecen al cluster.

Etapas III: Selección del Tópico de Mayor Bondad

Una vez obtenidos todos los grupos o clusters de itemsets frecuentes, se determina, para cada cluster, cuál es el tópico de mayor bondad que lo define, es decir, el tópico que mejor describe al cluster empleando las fórmulas de bondad presentadas en el capítulo 3.

Etapas IV: Mapeo de Tópicos-Usuarios

En esta última etapa, se realiza la asignación de tópicos (categorías) a cada usuario en función de los descriptores obtenidos para cada cluster y el peso que cada cluster posee para el usuario.

La Figura 4.10 ilustra el proceso del Categorizador.

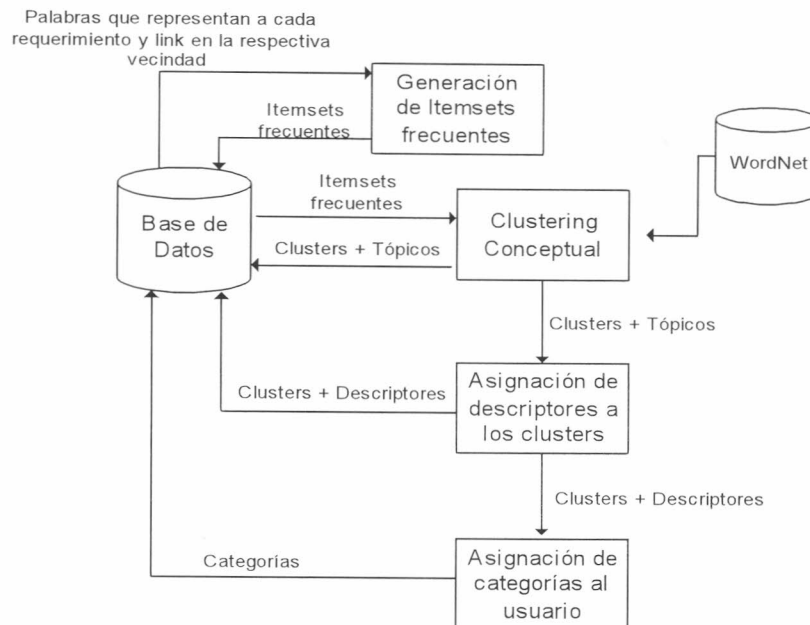


Figura 4.10: Proceso General del Categorizador

Los pasos del categorizador son los siguientes:

Obtiene los conjuntos de palabras que representan a los requerimientos y sus vecindades, para cada usuario.

Del conjunto de palabras obtiene los itemsets frecuentes, aplicando el algoritmo *Apriori*[AS94].

Luego agrupa los itemsets frecuentes en clusters utilizando las fórmulas de similaridad de palabras y similaridad semántica presentadas en el capítulo 3 y el algoritmo de generalización de tópicos que accede a WordNet.

Una vez obtenidos los clusters y los tópicos que los describen, obtiene para cada cluster el tópico de mayor bondad que será el descriptor de dicho cluster.

Finalmente los tópicos que describen los clusters serán las categorías correspondientes al usuario que realizó los requerimientos de los cuales se obtuvieron dichos clusters.

4.3 Resumen

En este capítulo hemos presentado la arquitectura del *SWMiner*, describiendo los diferentes procesos que realizan sus componentes: el *Depurador*, el *Preprocesador* y el *Categorizador*.

CAPÍTULO 5: Implementación

5.1 Introducción

En este capítulo detallamos cómo se implementó cada uno de los módulos que componen el sistema *SWMiner*: comenzamos con el *Depurador*, luego continuamos con el *Preprocesador* y finalmente presentamos el *Categorizador*.

El *Depurador* realiza el *Data Cleaning* de un log file, en este caso un log file de un servidor Proxy con formato *Extended Common Log Format (ECLF)*, identifica cada usuario presente en el mismo y las transacciones realizadas por cada uno de ellos.

El objetivo del *Preprocesador* es obtener las palabras relevantes de las páginas de contenido accedidas por los usuarios y de las páginas vecinas a éstas. Es decir, el *Preprocesador* se encarga de preprocesar el texto de los documentos y transformarlo en una forma estructurada para reducir la información del documento a una estructura tabular que será luego categorizada semánticamente.

El objetivo del *Categorizador* es identificar conjuntos de palabras que ocurren juntas con frecuencia en las páginas web accedidas por cada usuario y agrupar dichos conjuntos en clusters de conceptos o tópicos utilizando una taxonomía jerárquica conceptual, identificando de esta forma los tópicos que caracterizan a cada usuario.

5.2 Requerimientos del Sistema

Para el funcionamiento del sistema son necesarios los siguientes requisitos:

- Microprocesador Pentium 300MHz ó superior.
- 128 Mb de memoria RAM ó superior.
- 1Gb de espacio en disco rígido o superior.
- Windows 2000.
- Microsoft SQL Server 7.0.
- Acceso al directorio donde se almacenan los log files del servidor Proxy.
- Acceso a Internet.

5.3 Estructura de Directorios

El sistema arma automáticamente una estructura de directorios para almacenar las páginas que se analizan. Esta estructura de directorio se genera en el directorio del cual se toman los log files del servidor proxy.

El directorio *Paginas* almacena la estructura de directorios correspondiente a cada uno de los usuarios obtenidos de los log files analizados.

Veamos un ejemplo de la estructura de directorios que crea el sistema al obtener del log file 2 usuarios.

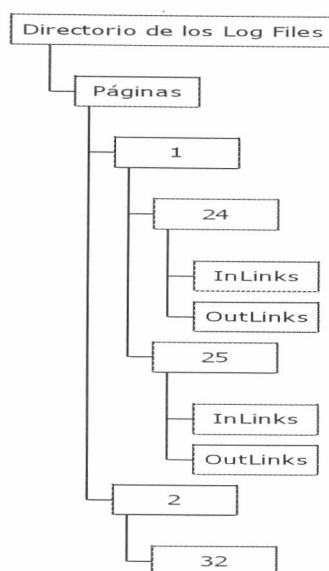


Figura 5.1: Ejemplo Estructura de Directorios de los Requerimientos de los Usuarios 1 y 2

Cada usuario tiene asignado un subdirectorio con el *id* del usuario en la base de datos, en el ejemplo de la figura 5.1 son los subdirectorios 1 y 2. Cada uno de estos subdirectorios posee un subdirectorio para cada requerimiento realizado por el usuario, que se identifica con el *id* del requerimiento en la base de datos. En la figura 5.1 para el usuario 1 existen dos subdirectorios de requerimientos: 24 y 25, y para el usuario 2, el subdirectorio 32.

Todos los subdirectorios que representan requerimientos almacenan:

- un archivo html correspondiente al requerimiento obtenido vía ftp al cual se le han eliminado los scripts y los comentarios. El nombre de este archivo es el *id*.html, donde *id* es el identificador unívoco del requerimiento en la base de datos.
- los archivos html correspondientes a las búsquedas de los inlinks. El nombre de estos archivos es: *busqueda**id*_n.html, donde *id* es el id del requerimiento en la base de datos y *n* es el contador de archivos de búsqueda. Cada archivo de búsqueda posee 10 inlinks (los motores de búsqueda que utilizamos (Altavista y Google) devuelven, por defecto, 10 links por consulta), por lo tanto si se analizaron esta cantidad de inlinks y es necesario analizar más, necesitamos realizar otra búsqueda y para no sobrescribir el archivo de búsqueda se utiliza el contador.
- Un archivo temporal:
 - ✓ *id*.txt, donde *id* es el identificador unívoco del requerimiento. Este archivo almacena todas las palabras contenidas en el archivo html del requerimiento en el orden en el que aparecen en él. Este archivo se utiliza para obtener las categorías sintácticas asociadas a las palabras.

Estos subdirectorios, a su vez, contienen 2 subdirectorios:

- *Inlinks*: archivos html de los inlinks del requerimiento, de los inlinks de los inlinks y los archivos de búsqueda para obtener los inlinks.
- *Outlinks*: archivos html de los outlinks del requerimiento y los outlinks de los outlinks.

5.4 Implementación del Depurador

Es importante realizar una primera aclaración: los log files con formato *Extended Common Log Format* que utilizamos en nuestra implementación son generados por el servidor proxy de una empresa cuyos empleados acceden a Internet solamente a través de éste; por lo tanto, en estos log files se registran todos los requerimientos realizados por dichos usuarios. Estos log files son el input del *Depurador*.

El *Depurador* efectúa una única pasada sobre cada log file que encuentra en el directorio de input especificado como parámetro del sistema. Esta pasada consiste en depurar las entradas del log file, almacenar las entradas válidas y los usuarios que las realizaron en una base de datos.

Depurar las entradas del log file implica aplicar el proceso de *Data Cleaning* sobre cada una de ellas. Durante este proceso, el *Depurador* analiza cada una de las entradas del log file, verificando si cumplen las condiciones necesarias para ser almacenadas en la base de datos como requerimientos y usuarios. Las acciones realizadas para tomar estas decisiones se detallan en el *Proceso de Data Cleaning*.

Durante el Proceso de Data Cleaning, el *Depurador* también realiza la identificación de los usuarios que generaron las entradas, basándose en la definición 3.2 de transacción enunciada en el capítulo 3. Esta definición surge como combinación del *Modelo Navegacional* con *Reference Length*.

A continuación detallamos las acciones que se realizan en el proceso de depuración o proceso de *Data Cleaning* sobre cada una de las entradas de los log files del servidor proxy y luego presentamos el pseudo código correspondiente a este proceso.

5.4.1 Data Cleaning

El Depurador necesita determinar qué entradas del log son válidas para poder luego almacenar en la base de datos estas entradas como requerimientos y usuarios. A continuación listamos las reglas que utiliza para determinar que la entrada que esta analizando es inválida.

- ✓ *Entradas correspondientes a archivos de gráficos, sonido o video*

Decidimos no analizar estas entradas porque consideramos que la información que podemos obtener, por ejemplo el nombre del archivo que contiene la imagen, no aporta información relevante para el análisis semántico.

- ✓ *Entradas correspondientes a archivos con extensiones: cab, js, vs, css*

Estas entradas corresponden a archivos de scripts, archivos de configuración de hojas de estilo, etc., los cuales no nos aportan nada a nuestra investigación.

- ✓ *Entradas correspondientes a requerimientos fallidos*

Determinar si una entrada corresponde o no a un requerimiento fallido, dependerá del código de estado asociado a la entrada.

Consideramos que las entradas cuyo código de estado sea alguno de los siguientes, son requerimientos fallidos.

206 – Requerimiento incompleto.

400 – Requerimiento incorrecto realizado por el cliente.

401 – Se requiere autorización para acceder al documento.

403 – El acceso al documento está prohibido.

404 – No se encuentra el documento.

501 – El método *GET* o *POST* es incompleto.

- ✓ *Entradas cuya cantidad de campos no coincida con la cantidad total de campos del formato de log file.*

Sólo consideramos inválidas a aquellas entrada en las cuales los campos ausentes sean relevantes para nuestro análisis. Tomamos esta decisión ya que no contamos con información adicional para completar estos campos.

- ✓ *Entradas generadas por agentes automáticos como Web Spiders, indexadores, chequeadores de links u otros agentes inteligentes.*

Asumimos que al trabajar sobre un log file de un Proxy Server de una empresa, en el cual se loguean los requerimientos realizados por los empleados de la empresa a Internet, no existirán entradas correspondientes a agentes automáticos porque no involucran la voluntad de los mismos, por lo tanto, el *Depurador* no verifica la existencia de agentes automáticos.

- ✓ *Entradas correspondientes a FTP*

Las entradas en las cuales el protocolo es ftp son consideradas inválidas.

- ✓ *Entradas generadas por el Proxy al cachear páginas*

Las entradas del log file cuyos campos *Ident* o *User Agent* sean iguales a *SYSTEM* o incluyan la palabra "proxy" serán consideradas inválidas porque dichas entradas representan requerimientos realizados por el servidor proxy para el mantenimiento de su cache.

- ✓ *Entradas correspondientes a accesos internos de la empresa*

Las entradas cuyo campo Request corresponda a alguna de las direcciones IP de la empresa son consideradas inválidas porque sólo nos interesa analizar el comportamiento de los usuarios en Internet y no en la intranet de la empresa.

- ✓ *Entradas con URLs no permitidas*

Consideramos inválidas aquellas entradas cuyo campo Request corresponda una Url que se encuentre en el archivo de Url's no permitidas (StopUrls.txt). Ej: url correspondiente a un motor de búsqueda.

- ✓ *Entradas con URLs con palabras no permitidas*

Las entradas cuyas URLs contengan alguna de las palabras del archivo StopWordsInUrls.txt son consideradas inválidas. Ejemplo: mail.

5.4.2 Pseudo código del Proceso de Data Cleaning

El proceso de *Data Cleaning* es realizado por el algoritmo *datacleaning* que detallamos a continuación.

Algoritmo datacleaning

```

Input:      log file L del servidor Proxy
Output:     -
Comienzo
  Abrir archivos L para lectura
  Para cada entrada e del log file con formato ECLF hacer
    Si datacleaningEntrada(e) entonces
      Insertar e en la base de datos
    Fin Si
  Fin Para
  Cerrar L
Fin
  
```

Veamos el pseudo código del algoritmo *datacleaningEntrada*.

Método *datacleaningEntrada*

Input: entrada e del log file con formato ECLF

Output: boolean indicando si la entrada e es válida o no

Comienzo

```
Boolean bRes = Not (tieneImagen(e)) And Not (tieneVideo(e))
bRes = bRes And Not (tieneSonido(e)) And Not (tieneScript(e))
bRes = bRes And Not (tieneStyleSheet(e)) And Not (isStopUrl(e))
bRes = bRes And Not (falloRequerimiento(e)) And Not (esFTP(e))
bRes = bRes And Not (generadaPorProxy(e)) And (entradaCompleta(e))
bRes = bRes And Not (isStopWordInUrl(e)) And Not (iPEmpresa(e))
Devolver bRes
```

Fin

En la Figura 5.2 se observa un ejemplo de log file antes de aplicarle el proceso de *Data Cleaning*. En este trozo de log file se registraron los requerimientos realizados por el usuario *krodriguez*.

```
10.18.4.1, krodriguez, -, N, 23/12/2000, 02:57:34, 1, -, -, www.musicworld.ru, -, 80, 150, 1645, 72, http, -, -,
http://www.musicworld.ru/index.html, -, Inet, 200, 0

10.18.4.1, krodriguez, -, N, 23/12/2000, 02:58:57, 1, -, -, www.musicworld.ru, -, 80, 20, 1645, 79, http, -, -,
http://www.musicworld.ru/images/search.gif, -, Inet, 200, 0

10.18.4.1, krodriguez, -, N, 23/12/2000, 03:01:31, 1, -, -, www.musicworld.ru, -, 80, 110, 1645, 74, http, -, -,
http://www.musicworld.ru/images/5.gif, -, Inet, 200, 0

10.18.4.4, roviela, -, N, 23/12/2000, 03:05:36, 1, -, -, www.rngroup.com, -, 80, 1382, 403, 166, http, -, -,
http://www.rngroup.com/images/arrow_yellow.gif, -, VCache, 304, 0

10.18.4.4, roviela, -, N, 23/12/2000, 03:05:49, 1, -, -, www.rngroup.com, -, 80, 1402, 1931, 144, http, -, -,
http://www.rngroup.com/rn_style.css, -, VCache, 304, 0

10.18.4.1, krodriguez, -, N, 23/12/2000, 03:06:26, 1, -, -, www.musicworld.ru, -, 80, 1362, 2742, 152, http, -, -,
http://www.musicworld.ru/images/11.gif, -, VCache, 304, 0

10.18.4.1, krodriguez, -, N, 23/12/2000, 03:06:39, 1, -, -, www.mp3delivery.com, -, 80, 1382, 1997, 154, http, -, -,
http://www.mp3delivery.com/albummenu.htm, -, VCache, 304, 0

10.18.4.1, krodriguez, -, N, 23/12/2000, 03:07:01, 1, -, -, www.mp3delivery.com, -, 80, 1402, 372, 148, http, -, -,
http://www.mp3delivery.com/center.gif, -, VCache, 304, 0

10.18.4.1, krodriguez, -, N, 23/12/2000, 03:07:22, 1, -, -, www.mp3delivery.com, -, 80, 1382, 1442, 146, http, -, -,
http://www.mp3delivery.com/tfade.gif, -, VCache, 304, 0

10.18.4.1, krodriguez, -, N, 23/12/2000, 03:07:58, 1, -, -, www.mp3delivery.com, -, 80, 4336, 862, 146, http, -, -,
http://www.mp3delivery.com/header.js, -, VCache, 304, 0
```

Figura 5.2: Ejemplo Log File antes de aplicar el Proceso de Data Cleaning

Luego de aplicar el proceso de *Data Cleaning* se obtiene el siguiente log de entradas inválidas (Figura 5.3).

```

10.18.4.1, krodriquez, -, N, 23/12/2000, 02:58:57, 1, -, -, www.musicworld.ru, -, 80, 20, 1645, 79, http, -, -,
http://www.musicworld.ru/images/search.gif, -, Inet, 200, 0

10.18.4.1, krodriquez, -, N, 23/12/2000, 03:01:31, 1, -, -, www.musicworld.ru, -, 80, 110, 1645, 74, http, -, -,
http://www.musicworld.ru/images/5.gif, -, Inet, 200, 0

10.18.4.4, roviela, -, N, 23/12/2000, 03:05:36, 1, -, -, www.rngrp.com, -, 80, 1382, 403, 166, http, -, -,
http://www.rngrp.com/images/arrowyellow.gif, -, VCache, 304, 0

10.18.4.4, roviela, -, N, 23/12/2000, 03:05:49, 1, -, -, www.rngrp.com, -, 80, 1402, 1931, 144, http, -, -,
http://www.rngrp.com/rn_style.css, -, VCache, 304, 0

10.18.4.1, krodriquez, -, N, 23/12/2000, 03:06:26, 1, -, -, www.musicworld.ru, -, 80, 1362, 2742, 152, http, -, -,
http://www.musicworld.ru/images/11.gif, -, VCache, 304, 0

10.18.4.1, krodriquez, -, N, 23/12/2000, 03:07:01, 1, -, -, www.mp3delivery.com, -, 80, 1402, 372, 148, http, -, -,
http://www.mp3delivery.com/center.gif, -, VCache, 304, 0

10.18.4.1, krodriquez, -, N, 23/12/2000, 03:07:22, 1, -, -, www.mp3delivery.com, -, 80, 1382, 1442, 146, http, -, -,
http://www.mp3delivery.com/tface.gif, -, VCache, 304, 0

10.18.4.1, krodriquez, -, N, 23/12/2000, 03:07:58, 1, -, -, www.mp3delivery.com, -, 80, 4336, 862, 146, http, -, -,
http://www.mp3delivery.com/header.js, -, VCache, 304, 0

```

Figura 5.3: Log file de entradas inválidas

Como puede observarse en la Figura 5.3, las entradas de este log file corresponden todas las imágenes que componen las páginas a las cuales accedió el usuario. De este trozo de log se obtuvo un usuario (Tabla 5.1) y 2 requerimientos (Tablas 5.2 y 5.3), los cuales se muestran en las Figuras 5.1 a 5.3.

Usuario	
Id	2
Ident	Krodriquez
Host	10.18.4.1
Fecha_Proceso	23/12/2000 02:57:34

Tabla 5.1: Usuario obtenido del Log File analizado

Requerimiento 1	
IdRequerimiento	3
FechaHora	23/12/2000 02:57:34
Request	http://www.musicworld.ru/index.html
Status	200

Tabla 5.2 : Primer requerimiento obtenido del Log File analizado

Requerimiento 2	
IdRequerimiento	4
FechaHora	23/12/2000 03:06:39
Request	http://www.mp3delivery.com/albummenu.htm
Status	200

Tabla 5.3: Segundo requerimiento obtenido del Log File analizado

Que se hayan obtenido dos requerimientos al aplicar el proceso de *Data Cleaning* al log file de la figura 5.2, no quiere decir que ambos sean analizados, ya que aún falta aplicar el preprocesamiento en el cual el requerimiento puede ser descartado, por ejemplo, por considerarse navegacional o por contener texto escrito en un idioma que no sea inglés.

5.5 Implementación del Preprocesador

Recordemos que el preprocesador se divide en 4 etapas:

- determinar si la página que está siendo analizada es *Navegacional* o de *Contenido*.
- obtener todas las palabras del documento html asociado a cada página de contenido e inferir el idioma con el cual fue escrito el documento.
- obtener las páginas vecinas a las páginas de contenido en inglés, que también deben estar en inglés.
- obtener las palabras relevantes de los documentos html de todas las páginas obtenidas en la etapa anterior, según su ubicación en el documento y almacenarlas en la base de datos para que sean analizadas por el *Categorizador*.

Veamos la implementación de cada una de estas etapas en detalle.

5.5.1 Distinción de Tipo de Página

Esta etapa consiste en establecer el tipo de las páginas accedidas por cada usuario según sea de *Contenido* o *Navegacional*. Para ello se utiliza el concepto de transacción (Definición 3.2) junto con las dos condiciones introducidas con el método *Reference Length* [W15] en el capítulo 3.

Para determinar si una página es de Contenido o Navegacional, el *Preprocesador* tiene en cuenta el tiempo transcurrido en la página, calculado como la diferencia en segundos entre el campo *Timestamp* (*Date y Time en ECLF*) del requerimiento actual y el campo *Timestamp* del siguiente requerimiento; si esta diferencia es menor a 180 segundos (3 minutos), el requerimiento es considerado de tipo *Navegacional*, de lo contrario se lo considera de tipo *Contenido*. Veamos el pseudo código asociado a esta etapa.

Algoritmo getPaginasDeContenido

Input: aReqs Array de requerimientos

Output: aContenido Array de páginas de contenido

Comienzo

```

    Para cada par de requerimientos consecutivos r1 y r2 ∈ aReqs hacer
        Si la fecha de r1 y r2 es la misma entonces
            Si la diferencia de las fechas de r1 y r2 en segundos es
                menor a Cota_Tiempo entonces
                    r1 es Navegacional
            Sino
                r1 es de Contenido y Agregar r1 al array aContenido
            Fin Si
        Sino
            Si las fechas de r1 y r2 difieren en 1 día y la hora de r2
                comienza con 23 entonces
                    r1 es Navegacional
            Sino
                r1 es de Contenido y Agregar r1 al array aContenido
            Fin Si
        Fin Si
    Fin Para
    Devolver aContenido

```


5.5.2 Identificación de Idiomas

En esta etapa se recorren todas las páginas de contenido. Para cada una de ellas se obtiene el archivo html vía ftp y se analiza el idioma correspondiente al texto contenido en dicho archivo.

Para la identificación de idiomas el *Preprocesador* cuenta con un módulo de identificación de idiomas que aplica la técnica de *Categorización de Texto usando Estadísticas de Frecuencias de Ngrams* [W21, CT94, CAV94, W22] a un conjunto acotado de palabras extraídas de las páginas. Esta técnica se detalla en el capítulo 3.

Para determinar el idioma de una página se obtienen las primeras 25 palabras del texto contenido en los tags del archivo html que representa la página. Se toman luego las primeras 10 palabras de longitud mayor a 2. Las cotas de palabras 25 y 10 se fijaron luego de realizar varias pruebas.

Uno puede preguntarse porqué la longitud de las palabras tiene que ser mayor a 2. Esto se debe a que si tomamos palabras de longitud 1 se analizarían las vocales, y la frecuencia de aparición de éstas en el texto resulta muy alta; en este análisis las vocales no aportan ninguna información para identificar el idioma ya que las mismas aparecen en la mayoría de los idiomas. Con las palabras de longitud 2 sucede algo similar, por ejemplo la palabra "No" puede pertenecer a varios idiomas (inglés, español, etc).

Una vez seleccionadas las 10 palabras, el *Preprocesador* obtiene los ngrams correspondientes a cada una de ellas, calcula la frecuencia de aparición de los ngrams en el documento, ordena los ngrams de mayor a menor según su frecuencia y calcula el puntaje de estos ngrams respecto a los 76 idiomas disponibles. El idioma que tenga menor puntaje es el que corresponde a la página web que está siendo analizada. Veamos el pseudo código asociado a esta etapa.

Algoritmo getPaginasEnIngles

Input: aReqs Array de requerimientos de contenido

Output: aReqEnIngles Array de requerimientos en inglés

Comienzo

Para cada requerimiento de contenido $r \in aReqs$ hacer

//obtiene el documento html correspondiente al requerimiento r

docHtml = ftp(r)

//verifica si el archivo html contiene un código de estado erróneo

//(es decir, distinto de 200)

Si sinError(docHtml) entonces

// Verifica si el idioma es Inglés

Si documentoEnIngles(docHtml) entonces

Agregar r a aReqEnIngles

Fin Si

Fin Si

Fin Para

Devolver aReqEnIngles

Veamos el pseudocódigo del método *documentoEnIngles*.

Método documentoEnIngles

```

Input: Documento Html doc
Output: Boolean indicando si el idioma es inglés o no
Comienzo
Array aItems, aNgrams
Matriz mIdiomas
    //parsea el documento Html y obtiene las palabras que contienen los
    //tags
    aItems = getItems(doc)

    //obtiene los ngrams de las palabras
    aNgrams = getNgrams(aItems)

    //Ordena el array de ngrams según su frecuencia de aparición en
    //el documento de mayor a menor
    aNgrams = ordenarPorFrecuencia(aNgrams,doc)

    //obtiene una matriz mIdiomas, puntaje de los ngrams
    mIdiomas = getPuntajeNgrams(aNgrams)

    //Obtiene el idioma de menor puntaje de mIdiomas
    idioma = getIdiomaMenorPuntaje(mIdiomas)
        Si idioma es Inglés entonces
            Devolver Verdadero
        Sino Devolver Falso
    Fin Si
Fin

```

Veamos el pseudo código de los métodos *getNgrams* y *getPuntajeNgrams*

Método getNgrams

```

Input: aItems Array de items (palabras)
Output: aNgrams Array de Ngrams
Comienzo
Array aNgramsItems
    Para cada item i ∈ aItems hacer
        //obtiene los ngrams del item i
        aNgramsItems = getNgramItem(i)
        Agregar los elementos de aNgramsItems a aNgrams
    Fin Para
Devolver aNgrams

```

Método getNgramsItem

```

Input: i Item
Output: aNgrams Array de ngrams
Comienzo
Array aPares
Contante BLANCO = _
String item
    item = BLANCO + i + BLANCO
    Para n=1 hasta CotaLongNgram hacer
        //obtiene un array de pares de letras consecutivas de longitud n
        aPares = getParesLong(n,item)
        Agregar aPares a aNgrams
    Fin Para
Devolver aNgrams
Fin

```

Para finalizar con esta etapa presentamos el método *getPuntajeNgrams*.

Método getPuntajesNGrams

```

Input:  aNGrams Array de ngrams ordenado de mayor a menor según su
        frecuencia de aparición en el documento
Output: mIdiomas Matriz de idioma-puntaje correspondiente al puntaje
        total de todos los ngrams del array en cada idioma
Comienzo
Entero Puntaje
Array aIdiomas = getIdiomas()
Para cada ngram e aNGrams hacer
    //obtiene el puntaje de ngram en cada uno de los idiomas, es decir el
    //nro de fila donde se encuentra el ngram en el archivo de c/idioma
    Para cada idioma hacer
        Puntaje = IndiceNgramEnArray - PosicionNgramEnArchivoIdioma
        Agregar el Puntaje a un array de puntajes del idioma

    Fin Para
Fin Para
Para cada idioma hacer
    Sumar los puntajes agregados al idioma
    Agregar el idioma y la suma obtenida a la matriz mIdiomas
Fin Para
Devolver mIdiomas
Fin

```

Ejemplo 5.1. Supongamos que de los tags del documento html que esta siendo analizado se extraen las primeras 25 palabras y luego se obtienen de ellas las siguientes 5 palabras: *the, whale, and, have* y *about*. De estas palabras obtenemos 90 ngrams, algunos ejemplos de ellos son: *abou, about, bout_, h, ha, av, ve, e_, an, and, nd_, etc.*

Luego se ordenan los ngrams según su frecuencia de aparición en el texto asociado al documento html original. Los ngrams de mayor frecuencia son: *e, a, t, o, n, h, e_, l, t, u, d, th, th, he, an, t_, _w, _a, b, etc.*

Se calcula el puntaje de estos ngrams buscándolos en los 76 archivos de perfiles de idiomas, sumando el puntaje asociado a cada ngram en el archivo para obtener el puntaje total correspondiente a cada perfil.

Cada línea de los archivos de perfiles de idiomas tienen el siguiente formato: Ngram-Puntaje. Ver Anexo Ejemplo de archivo de Perfiles de Idioma. La tabla 5.4 ilustra el puntaje obtenido

Puntajes de Perfiles de Idiomas	
Catalán	55419
Francés	53002
Inglés	33021
Español	56496

Tabla 5.4: Ejemplo de Puntajes de Perfiles de Idiomas

Finalmente obtenemos el idioma con menor puntaje, que en este caso es *Inglés*.

5.5.3 Obtención de Páginas Vecinas

Esta etapa consiste en recorrer las páginas de contenido en Inglés accedidas por cada uno de los usuarios y obtener el conjunto de páginas apuntadas por cada página p con un alcance de k niveles (parámetro del sistema ingresado por el usuario) de profundidad, excluyendo aquellas páginas que corresponden a:

- links navegacionales dentro del mismo sitio y aquellos que comienzan con "mail", "#", "ftp", "javascript:", "/", "../", ".".
- links incluídos en el archivo de *StopUrls*
- links escritos en otro idioma que no sea inglés
- links que incluyen alguna palabra del archivo *StopWordUrl*

Es decir,

$$OutLink = \bigcup_{i=1}^k Links_i$$

donde $Links_1$ es el conjunto de páginas apuntadas por p , $Links_2$ es el conjunto de páginas apuntadas por las páginas de $Link_1$, y así sucesivamente. En general, el conjunto de páginas $Link_{i+1}$ es apuntado por las páginas de $Link_i$ con $1 \leq i \leq k$, donde todas estas páginas cumplen las condiciones enunciadas con anterioridad.

También se debe obtener un conjunto acotado de las páginas que apuntan a p con un alcance de k niveles de profundidad, excluyendo aquellas páginas que cumplen las condiciones anteriores. Es decir,

$$InLink = \bigcup_{i=1}^k Links_i$$

donde $Links_1$ es el conjunto de páginas que apuntan a p , $Links_2$ es el conjunto acotado de páginas que apuntan a las páginas de $Link_1$, y así sucesivamente. En general, el conjunto de páginas $Link_{i+1}$ apunta a las páginas de $Link_i$ con $1 \leq i \leq k$.

Tanto para los *InLinks* como para los *OutLinks*, antes de pasar de un nivel a otro se analiza el idioma del documento html correspondiente a cada link obtenido y si el idioma no es Inglés, se descarta el link.

Para obtener las páginas que apuntan a p , se envía la consulta *link:p* a un motor de búsqueda, por ejemplo a *Altavista* y el motor devuelve un documento html con el resultado de la búsqueda. El Preprocesador debe parsear este documento para obtener las urls que apuntan a p . Después de realizar varias pruebas con *Altavista*, llegamos a la conclusión que se deben obtener todos los tags *href* del documento Html que contengan el parámetro r (*parámetro de la url*), el cual contiene una de las urls buscadas.

Por ejemplo, parte del documento devuelto por *Altavista* al buscar las páginas que apuntan a www.dc.uba.ar es el siguiente:

```
...
<b><a href="/r?srp1" style="COLOR:black;TEXT-DECORATION:none" title="Submit web pages
with Express Inclusion">
We found 468 results:</b></a><br>
<a href="/r?ck_sm=32e43d2f&ref=20080&r=http://www.anice.net.ar/themestart/government
.html" onMouseOver="status='http://www.anice.net.ar/themestart/government.html';
return true;">Theme Start: Government</a>
<br>Government Gobierno. add item. International / Internacional. Government /
Gobierno. National / Nacional. --&gt; Government / Gobierno Presidencia de
la...<br>
<span class=x>URL: http://www.anice.net.ar/themestart/government.html
```

```

<ahref="http://jump.altavista.com/trans.go?urltext=http://www.anice.net.ar/themesta
rt/government.html&language=es" class=x>Translate</a>
<br>
<ahref="http://jump.altavista.com/more_from_site.go?q&stype=stext&Translate=on&sc=o
ff&q=link%3Awww.dc.uba.ar&si=7fe473c5301200" class=x>Additional relevant
pages from this site</a>&nbsp;
</span><br><br>
<ahref="/r?ck_sm=ecb01ca4&ref=20080&r=http://sg.dir.yahoo.com/Science/Computer_Scie
nce/Courses/Course_Information/"
onMouseOver="status='http://sg.dir.yahoo.com/Science/Computer_Science/Courses/Cours
e_Information/'; return true;">Yahoo! Science>Computer
Science>Courses>Course Information</a>
...

```

En este documento se han resaltado dos de las url's que apuntan a www.dc.uba.ar.

Si en vez de utilizar *Altavista* usamos *Google*, debemos invocarlo de la siguiente manera: <http://www.google.com/search?q=www.dc.uba.ar>, y para obtener los inlinks del documento html devuelto al realizar la búsqueda, debemos tomar todos los tags *href* y excluir los que contienen alguno de los siguientes substrings "*q=link:*", "*/help/*", "*/search?*", "*/ads/*", "*/services/*", "*/news/*", "*/language_tools?*", "*/about*"; ya que representan links internos del buscador.

Es importante aclarar que tanto con *Altavista* como con *Google*, se obtienen 10 URLs por búsqueda, por lo tanto para pasar a la siguiente página resultado es necesario enviarle un parámetro especial al motor de búsqueda.

Por ejemplo, para obtener 20 url's de páginas que apuntan a www.dc.uba.ar, el *Preprocesador* deberá invocar al motor de búsqueda dos veces, de la siguiente forma:

- Para *Altavista*:
 - ✓ <http://www.altavista.com/cgi-bin/query?q=link:www.dc.uba.ar>
 - ✓ <http://www.altavista.com/cgi-bin/query?q=link:www.dc.uba.ar&stq=10>
- Para *Google*:
 - ✓ <http://www.google.com/search?q=link:www.dc.uba.ar>
 - ✓ <http://www.google.com/search?q=link:www.dc.uba.ar&start=10>

En todos los casos, ya sea para obtener un InLink o un OutLink, el preprocesador debe verificar si el documento html obtenido de Internet corresponde o no a un error. Al obtener el archivo html utilizando el protocolo http, las primeras líneas del archivo tendrán un header, cuya primer línea tiene la siguiente forma:

HTTP/Versión Status Descripción

donde *Versión* es el número de versión del protocolo, *Status* es el código resultado de la obtención del archivo y *Descripción* es OK o una leyenda descriptiva del error ocurrido.

El preprocesador debe verificar que el campo *Status* de la primera línea del header sea 200 (ver *Anexo Códigos de Estado del Protocolo Http*). Esta verificación también debe realizarse con los requerimientos realizados por los usuarios.

Finalmente, se obtiene el conjunto $Q = InLink \cup OutLink$. Veamos el pseudo código asociado a esta etapa.

Algoritmo getVecinos

Input: aReqs Array de requerimientos de contenido en inglés,

Output: --

Comienzo

```

    Para cada requerimiento r ∈ aReqs hacer
        // obtiene las páginas que apuntan a r que tienen distinto dominio
        // el requerimiento r
        ftpInLinks(r,false)
        Si la cantidad de páginas obtenidas que apuntan al requereimiento r
        es menor que la cota de páginas por requerimiento(parámetro
        del sistema) Entonces
            // obtiene las páginas que apuntan a r que tienen el
            // mismo dominio que el requerimiento r
            ftpInLinks(r,true)
        Fin Si
        // obtiene las páginas apuntadas por r que tienen distinto dominio
        ftpOutLinks(r,false)
        Si la cantidad de páginas obtenidas que apuntan al requereimiento r
        es menor que la cota de páginas por requerimiento Entonces
            // obtiene las páginas apuntadas r que tienen el mismo dominio que r
            ftpOutLinks(r,true)
        Fin Si
    Fin Para
Fin

```

Método ftpInLinksInput: Requerimiento r,
bIgualDominio Boolean que indica si los link a obtener pueden o
no pertenecer al mismo dominio que el requerimiento r

Output: --

Comienzo

Array aLinks

Link l

Documento Html doc

```

    // Parsea el documento Html devuelto por el motor de búsqueda para
    // obtener un array con los links que apuntan a r (tags href)
    aLinks = searchLinksToURL(r)
    Para cada link l ∈ aLinks hacer
        // obtiene el documento html correspondiente a l
        doc = ftp(l)
        Si DocumentoEnIngles(docHtml) y la cantidad de inLinks en la BD es
        Menor o igual a la CotaMaxInLinks y la url de l no contiene
        palabras que estén en el archivo StopWordsUrl Y la url de l
        no está en StopUrls
            Entonces
                Si No bMismoDominio Entonces
                    Si el dominio de la url de l y r son distintos Entonces
                        Almacenar doc en la base de datos como un link
                        // Cota Niveles parámetro del sistema
                        ftpInLinksLink(l,bMismoDominio,CotaNiveles-1)
                    Fin Sin
                Sino
                    Almacenar doc en la base de datos como un link
                    // Cota Niveles parámetro del sistema
                    ftpInLinksLink(l,bMismoDominio,CotaNiveles-1)
                Fin Si
            Fin Si
        Fin Si
    Fin

```


El método *ftpInLinksLink* hace lo mismo que el método *ftpInLinks*, sólo difiere en lo siguiente:

- ✓ recibe como parámetro un link y no un requerimiento,
- ✓ se agrega el parámetro *CotaNiveles*, el cual se va decrementando en cada llamada recursiva. Al comienzo, el método verifica si la cota de niveles (parámetro del sistema) es o no cero y cuando lo es finaliza la recursividad.

La constante *CotaMaxInLinks* es un parámetro del sistema que indica la cantidad de inlinks que desea obtener por requerimiento el usuario del sistema.

```

Método ftpOutLinks
Input:   Requerimiento r
Output:  -
Comienzo
Array    aLinks, aOutLinks
Documento Html doc
//Parsea el documento html de r y obtiene los tags de links (href)
aLinks = getLinks(r)
Para cada link l ∈ aLinks hacer
    // obtiene el el documento html correspondiente a l
    doc = ftp(l)

    Si DocumentoEnIngles(doc) Y la cantidad de outLinks en la BD es menor
    o igual a la CotaMaxOutLinks Y la url de l no contiene palabras
    que estén en el archivos StopWordsUrl y la url de l no está
    en StopUrls
    Entonces
        Si No bMismoDominio Entonces
            Si el dominio de la url de l y r son distintos Entonces
                Almacenar doc en la base de datos como un link
                ftpOutLinksLink(l, bMismoDominio, CotaNiveles-1)
            Fin Si
        Sino
            Almacenar doc en la base de datos como un link
            ftpOutLinksLink(l, bMismoDominio, CotaNiveles-1)
        Fin Si
    Fin Para
Fin

```

Igual que antes el método *ftpOutLinksLink* es similar al método *ftpOutLinks* sólo difieren en los parámetros (uno trabaja con links y el otro con requerimientos) y en el chequeo de niveles (*CotaNiveles*).

5.5.4 Representación de Páginas

Las páginas que pertenecen al conjunto *Q* obtenido en la sección anterior, se representan con un conjunto de palabras relevantes, de ahora en más Ítems, que se obtienen del documento html asociado a cada una de ellas.

Antes de realizar la selección de las palabras que se analizarán, se deben eliminar todos los comentarios y scripts de diferentes lenguajes de programación que contenga el documento html. Este preprocesamiento del documento se realiza para evitar obtener palabras que no aporten información relevante para el análisis semántico.

Como mencionamos en el Capítulo 3, debemos establecer una prioridad a los tags. La figura 5.5 muestra los tags que consideramos que contienen información relevante para la categorización semántica, ordenados de mayor a menor según su prioridad.

Tags Html [W16]		
Tag	Propiedad	Descripción
Title		Título de la página. La mayoría de los buscadores utilizan las palabras que contiene para clasificar la página.
Metadata	Keywords	Palabras clave que identifican a la página. Este tag es utilizado por algunos motores de búsqueda y no todos los documentos las incluyen, depende del programador.
Metadata	Clasification	Palabras que clasifican la página. Este tag es utilizado por algunos motores de búsqueda y no todas las páginas lo incluyen, depende del programador.
Metadata	Description	Palabras que describen el contenido de la página. Este tag es utilizado por algunos motores de búsqueda y no todas las páginas lo incluyen, depende del programador.
H1		Cabecera de gran tamaño para indicar títulos en el documento
H2		Cabecera de menor tamaño que la anterior utilizada para indicar títulos en el documento
H3		Palabras contenidas en cabeceras de menor tamaño que H2 utilizadas para indicar títulos en el documento
Big		Texto de similar tamaño que H3 pero no resaltado en negrita
Strong		El texto contenido en este tag se resalta en Negrita
B		Idem anterior
I		Texto resaltado en Cursiva
Em		Texto en Cursiva, generalmente utilizado para citar un párrafo de otro autor
Cite		Texto en Cursiva, generalmente utilizado para citar un párrafo de otro autor
Blockquote		Texto de similar tamaño al anterior pero no en cursiva
H4		Palabras contenidas en cabeceras de menor tamaño que H3 utilizadas para indicar títulos en el documento
H5		Palabras contenidas en cabeceras de menor tamaño que H4 utilizadas para indicar títulos en el documento
Kbd		Texto de similar tamaño al H5 pero de trazo más fino
H6		Palabras contenidas en cabeceras de menor tamaño que H5 utilizadas para indicar títulos en el documento
A	Title	Descripción del link, aparece cuando se posiciona el mouse sobre el link
Img	Alt	Descripción de la imagen, aparece cuando se posiciona el mouse sobre la imagen
Body		Palabras en el cuerpo del documento

Tabla 5.5: Tags Html ordenados por prioridad de mayor a menor

Para un mayor detalle del resultado final de dichos tags ver el Anexo *Ejemplos Tags Html*.

Es evidente que no podemos tomar todas las palabras contenidas en un documento html, porque podríamos obtener muchísimas palabras por cada documento y por consiguiente la categorización demandaría demasiado tiempo. Por tal razón decidimos agregar un parámetro al sistema, el cual servirá de cota para la cantidad de palabras que representarán a los requerimientos y links.

Antes de obtener esta cantidad de palabras debemos analizar con algún criterio todas las palabras obtenidas para determinar cuales son las mejores para representar la página. El

criterio de selección que utilizamos depende de la prioridad del tag que contiene a la palabra y en el caso particular del tag *Body*, una vez obtenidas todas las palabras que sólo están contenidas en este tag, las ordenamos según su frecuencia de aparición.

Además las palabras seleccionadas tienen que cumplir las siguientes condiciones:

- ✓ *Los caracteres que componen las palabras deben ser Alfabéticos, es decir se descartan las palabras con caracteres especiales (",\$,%,&,#,etc.) o caracteres numéricos.*
- ✓ *Si alguna de las palabras contiene un carácter especial al comienzo o al final, se elimina dicho carácter y se procesa nuevamente.*

Decidimos invertir tiempo en este punto para no descartar palabras de tags de alta prioridad, por el simple hecho de que comiencen o finalicen con un carácter especial. Por ejemplo, si en el tag *Title* aparece la palabra "Whale," si no hiciésemos este paso las descartaríamos porque finaliza con coma.

- ✓ *Las palabras seleccionadas deben ser sustantivos (part of speech) [W31]*
- ✓ *Las palabras seleccionadas no deben pertenecer al archivo StopWords [W29] (ver Anexo StopWords).*
- ✓ *Si alguna palabra no pertenece al archivo StopWords [W29], se le aplicará la técnica Lemmatiser para obtener el lema de la palabra y se procederá a analizarlo.*
- ✓ *Las palabras seleccionadas no deben ser Nombres Propios. Contamos con un archivo importado en una tabla de la base de datos que contiene un listado de Nombres Propios.*
- ✓ *Finalmente, las palabras seleccionadas deben estar presentes en la base de datos jerárquica WordNet [W30], ya que esta se utilizará para obtener tópicos más generales en la categorización.*

Por último las palabras seleccionadas con los criterios mencionados con anterioridad se almacenan en la base de datos para su posterior análisis semántico.

Veamos el pseudo código asociado a esta etapa.

Algoritmo EtapaRepresentacionDePaginas

Input: - Output: -

Comienzo

Array aReq

//obtiene usuarios que tiene requerimientos en estado "FTP Realizado"

Array aUsuarios = getUsers("FTP Realizado")

Para cada usuario u ∈ aUsuarios hacer

//obtiene requerimientos del usuario u que con estado "FTP Realizado"

aReq = getRequerimientos(u,"FTP Realizado")

//obtiene los items relevantes de los requerimientos

getItemsRequerimientos(aReq)

Para cada requerimiento r ∈ aReq hacer

// Obtiene los items relevantes para los in y outlinks de

getItemsLinks(r)

Fin Para

Fin Para

Fin

Método getItemLinks

```

Input:  Requerimiento r
Output: -
Comienzo
DocHtml doc
//obtiene todos los in y out links del requerimiento
Array  aLinks = getLinks(r)
Archivo archTxt
Array  aItems, aSustantivos

Para cada Link l ∈ aLinks hacer
    // Obtiene el archivo html almacenado en disco
    doc = getHtml(l)
    //obtiene las palabras contenidas en los tags del archivo html
    aItems = getItem(doc)
    //obtiene el texto original del archivo html
    archTxt = partOfSpeech(doc)
    // Obtiene un array con los sustantivos del texto original
    aSustantivos = getNouns(archTxt,aItems)
    //Elimina las palabras con caracteres especiales
    aSustantivos = deleteSpecialCharacters(aSustantivos)
    Para cada Item i ∈ aSustantivos hacer
        Si i no está en el archivo StopWords y i no es un nombre propio
            Entonces
                Si existe lemmatizer(i) Entonces
                    Si lemmatizer(i) ∈ WordNet Entonces
                        Agregar lemmatizer(i) a la base de datos
                    Fin Si
                Sino
                    Si i ∈ WordNet entonces
                        Agregar i a la base de datos
                    Fin Si
            Fin Si
        Fin Si
    Fin Para
Fin Para
Fin

```

Recordemos que WordNet [W30] es una base de datos semántica jerárquica compuesta por *synsets* (*conjuntos de sinónimos*), por lo tanto verificar la existencia de un ítem en ella consiste en hallar al menos un synset que contenga al ítem que esta siendo analizado. Ejemplos de synsets pueden encontrarse más adelante en la implementación del *Categorizador*.

Ejemplo 5.2. La Figura 5.4 muestra el código Html correspondiente a la url www.whale-web.com.

```

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
  <TITLE>Whale Web</TITLE>
  <META content="text/html; charset=windows-1252"
    http-equiv=Content-Type>
  <META content="MSHTML 5.00.3315.2870" name=GENERATOR>
</HEAD>
<BODY aLink=#8fb4dd bgColor=#000072 link=#8fb4dd text=#000072 vLink=#8fb4dd>
<MAP name=mainpage><AREA coords=3,23,83,53 href="http://
  www.whale-web.com/siteinfo.html" shape=RECT>
  <AREA coords=3,53,83,80 href="http://
    forums.whale-web.com/postlist.php?Cat=&Board=news" shape=RECT>
</MAP>
<MAP name=ring>
  <AREA coords=21,27,75,44
    href="http://www.webring.org/cgi-bin/webring?ring=tursiops&id=26&prev"
    shape=RECT>
  <AREA coords=83,27,123,44
    href="http://www.webring.org/cgi-bin/webring?ring=tursiops&id=26&next5"
    shape=RECT>
  <AREA coords=132,27,161,44 href="http://www.tursiops.org/ring.html"
    shape=RECT><AREA coords=167,27,201,44
    href="http://www.webring.org/cgi-bin/webring?ring=tursiops&id=26&next"
    shape=RECT></MAP>
<CENTER>
<TABLE border=0 borderColor=#000072 cellPadding=0 cellSpacing=0 width="100%">
  <TBODY>
    <TR>
      <TD bgColor=#4c72bb width=141><IMG alt="Sumar! ="
        src="Whale Web_archivos/sumar-top.jpg"></TD>
      <TD bgColor=#4c72bb><IMG alt="Whale Web Header"
        src="Whale Web_archivos/ww-head.jpg"></TD>
      <TD align=right bgColor=#4c72bb><IMG align=right alt=Menu border=0
        src="Whale Web_archivos/main-top.jpg" useMap=#mainpage></TD></TR>
    <TR>
      <TD align=middle bgColor=#4c72bb vAlign=top>
        <CENTER><P>&nbsp;</P>
        <P><FONT color=#000072 face="comic sans, comic sans ms" size=-1><A
          href="http://www.whale-web.com/finna.html" target=top>In Memory of
          Finna</A></FONT></CENTER>
        <P><P><I>From space, the planet is blue. From space, the planet is the
          territory, not of humans, but of the whale.<BR>-Heathcote Williams</I>
        <UL></UL></TD>
      <TD bgColor=#8fb4dd colSpan=2 width="100%"><FONT
        face="comic sans, comic sans ms" size=-1>
        <P><P>
      <TABLE border=0><TBODY>
        <TR>
          <TD>&nbsp;<A href="http://www.whale-web.com/whales/index.html"><IMG
            border=0 height=86 src="Whale Web_archivos/whales-main.jpg"
            width=213></A> </TD>
          <TD><FONT face="comic sans, comic sans ms" size=-1>Many have been
            hunted to near extinction - some may never recover. Only recently
            have we learned more about the largest animal to have ever lived, or
            about the whale that composes songs year after year.</FONT></TD></TR>
        <TR>
          <TD>&nbsp;<A
            href="http://www.whale-web.com/dolphins/index.html"><IMG border=0
            height=86 src="Whale Web_archivos/dolphins-main.jpg" width=232></A>
          </TD>
          <TD><FONT face="comic sans, comic sans ms" size=-1>We've been entranced
            for years by the bottlenose's smile and stood in awe of the largest dolphin
            - orca - the top predator of the ocean that is
            <A href="http://www.whale-web.com/porpoises/index.html"><IMG border=0
            height=86 src="Whale Web_archivos/porpoises-main.jpg" width=256></A>
          </TD>
          <TD><FONT face="comic sans, comic sans ms" size=-1>The smallest
            family, these shy and retiring cetaceans are often overlooked. Dive
            into their world to learn more about them.</FONT></TD></TR>
      </TABLE>
    </TR>
  </TBODY>
</TABLE>

```

```

<TR>
  <TD>&nbsp;<A href="http://www.whale-web.com/resources/index.html"><IMG
    border=0 height=86 src="Whale Web_archivos/resources-main.jpg"
    width=268></A>
  </TD>
  <TD><FONT face="comic sans, comic sans ms" size=-1>Learn more about
    cetaceans in general with videos, frequently asked questions,
    scientific classifications, myths and legends, and careers with
    marine mammals.</FONT> </TD></TR>
<TR>
  <TD>&nbsp;<A href="http://forums.whale-web.com/"><IMG border=0
    src="Whale Web_archivos/forum-main.jpg"></A> </TD>
  <TD><FONT face="comic sans, comic sans ms" size=-1>Talk and discuss
    cetaceans with others on the Whale Web forums!</FONT></TD></TR>
<TR>
  <TD>&nbsp;<A href="http://www.whale-web.com/store.html"><IMG
    border=0 height=86 src="Whale Web_archivos/store-main.jpg"
    width=190></A> </TD>
  <TD><FONT face="comic sans, comic sans ms" size=-1>Help us keep this
    site going by buying a fantastic book, video, or CD featuring
    cetaceans! We have read or thoroughly examined every one of these
    products and own most of them. We'll give you our reviews so you
    know you're getting some of the best items available.</FONT></TD></TR>
<TR>
  <TD>&nbsp;<A href="http://www.whale-web.com/guestbook/guestbook.html"><IMG
    border=0 src="Whale Web_archivos/logbook-main.jpg"></A> </TD>
  <TD><FONT face="comic sans, comic sans ms" size=-1>Sign our logbook
    and let us know what you thought of your whale
    watch!</FONT></TD></TR></TBODY></TABLE>
<CENTER>
  <P><IMG border=0 src="Whale Web_archivos/ring.jpg" useMap=#ring>
  <P><FONT face="comic sans, comic sans ms">You are whale watcher number
  <IMG src="Whale Web_archivos/Count.gif"> since June 15th,
  2000!</FONT></CENTER><P>&nbsp;</FONT></P></TD></TR></TBODY>
</TABLE></CENTER>
<P><FONT color=#8fb4dd face="comic sans, comic sans ms" size=-2>last updated
2/5/01<BR>Whale Web copyright 1996-2000<BR><A
href="mailto:kasatka@whale-web.com">kasatka@whale-web.com</A><BR><A
href="mailto:keet@whale-web.com">keet@whale-web.com</A><BR></FONT></P>
</BODY>
</HTML>

```

Figura 5.4: Código Html correspondiente a la url www.whale-web.com

Supongamos que ejecutamos el *Preprocesador* seteando el parámetro que especifica la cantidad de palabras por página en 30 y que obtuvimos los siguientes ítems: *WHALE*, *CETACEANS*, *SPACE*, *PLANET*, *LARGE*, *YEAR*, *LEARN*, *MEMORY*, *FINNA*, *TERRITORY*, *HUMAN*, *EXTINCTION*, *ANIMAL*, *COMPOSE*, *ENTRANCE*, *SMILE*, *STAND*, *AWE*, *DOLPHIN*, *TOP*, *OCEAN*, *MAN*, *SMALL*, *SHY*, *RETIRE*, *OVERLOOK*, *VIDEO*, *QUESTION*, *CLASSIFICATION* y *MYTH*. El ítem *WHALE* lo obtuvimos del tag *Title* y los demás ítems del tag *Body*, luego de ordenar todos sus ítems según la frecuencia de aparición de los mismos en el texto.

5.6 Implementación del Categorizador

El Categorizador realiza el proceso de análisis por cada usuario y al igual que el Preprocesador, también decidimos separarlo en diferentes etapas: Generación de Itemsets Frecuentes, Clustering de Itemsets Frecuentes, Selección del Tópico de Mayor Bondad y Mapeo Tópicos-Usuarios.

En la primer etapa se obtienen todos los conjuntos de palabras frecuentes de las páginas web accedidas por el usuario y sus vecindades, utilizando el algoritmo Apriori de Association Rules [AS94]. En la etapa siguiente, se obtienen los grupos o clusters de conjuntos de palabras frecuentes empleando un algoritmo de Clustering Conceptual del tipo Jerárquico y

Aglomerativo[HM96]. Luego se identifica cada cluster con el tópico de mayor bondad entre todos los tópicos que caracterizan al cluster[LK99]. La última etapa consiste en realizar la asignación de estos tópicos al usuario con su correspondiente peso, realizando un mapeo inverso que depende de las páginas a las cuales accedió el usuario.

Cada una de estas etapas se ejecuta para cada usuario por separado. El pseudo código del algoritmo principal es el siguiente.

Algoritmo categorizacionSemantica

```

Input:  s soporte mínimo
Output: -
Comienzo
Array aRequerimientos, aItemsets, aClusters

//obtiene los usuarios que tienen requerimientos en estado "Preprocesado"
Array aUsuarios = getUsuariosDB("Preprocesado")
Para cada usuario u ∈ aUsuarios
    //obtiene los requerimientos del usuario en estado "Preprocesado"
    aRequerimientos = getRequerimientos(u,"Preprocesado")
    Para cada requerimiento r ∈ aRequerimientos
        //genera los itemsets frecuentes para el requerimiento
        aItemsets = generarItemsetsFrecuentes(s,u,r)
        //genera los clusters de itemsets frecuentes
        Agregar el resultado de Clusterer(aItemsets,r,u) a aClusters
    Fin Para
    //Agrupa los clusters de todos los requerimientos
    aClusters = ClustererOfClusters(aClusters,u)
    //Selección del mejor descriptor de cada cluster.
    aClusters = computeBondad(aClusters)
    //Calcula el peso de los clusters
    computeClustersWeight(aClusters,u)
    //Mapeo de los tópicos obtenidos al usuario u.
    mapping(aClusters,u)
Fin Para
Fin

```

5.6.1 Generación de Itemsets Frecuentes

En esta etapa se obtienen los conjuntos de palabras (ítems) que ocurren juntas con cierta frecuencia, llamados itemsets frecuentes en el conjunto de páginas web formado por el requerimiento en análisis y la vecindad definida para dicho requerimiento.

La frecuencia de ocurrencia de un itemset es variable y se determina a través de un parámetro del sistema (*Soporte de Association Rules*). Para obtener los itemsets frecuentes invocamos al algoritmo *Apriori* [AS94].

Algoritmo getItemsetsFrecuentes

```

Input:  s soporte mínimo, r requerimiento, u usuario
Output: aItemsets Array de itemsets frecuentes
Comienzo
    // Obtiene los itemsets frecuentes
    aItemsets = Apriori(s,u,r)
Fin

```

El algoritmo *Apriori* invocado en este método, genera los candidatos de itemsets frecuentes en cada paso usando los itemsets frecuentes encontrados en el paso anterior. La intuición básica es que cualquier subconjunto de un itemset frecuente debe ser frecuente. Veamos el pseudo código correspondiente a este algoritmo.

Algoritmo Apriori

```

Input:  s soporte mínimo, u usuario, r requerimiento
Output: aItemsetsFrecuentes Array de itemsets frecuentes
Comienzo
Entero cantTransacciones, pesoMinimo
Array aItemsetsCandidatos, aItemsFreq
  //obtiene la cantidad de transacciones
  cantTransacciones = getCantTransacciones(r)
  //calcula el soporte mínimo
  soporteMinimo = ( cantTransacciones * s) + 1
  Si cantTransacciones > 0 entonces
    // Obtiene los itemsets frecuentes que cumplen con el soporte mínimo
    aItemsFreq = getItemsetsFrecuentes(soporteMinimo,r)
    // Recorre los items y crea itemsets candidatos de 1-item por c/u
    Para cada i ∈ aItemsFreq hacer
      Crea un itemset t con el item i
      Agregar el itemset t a aItemsetsCandidatos
    Fin Para
    Mientras aItemsetsCandidatos tenga elementos
      // calcula el peso de cada itemset candidato
      weighCandidates(aItemsetsCandidatos);
      //obtiene los candidatos frecuentes que verifican el soporte mínimo
      aItemsets = evaluateCandidates(aItemsetsCandidatos);
      Agregar a aItemsetsFrecuentes todos los itemsets de aItemsets;
      //genera nuevos candidatos a partir de los itemsets frecuentes
      aItemsetsCandidatos = generateCandidates(aItemsetsFrecuentes);
    Fin Mientras
  Fin Si
  Devuelve aItemsetsFrecuentes
Fin

```

Inicialmente, el algoritmo genera un itemset candidato a ser itemset frecuente por cada ítem frecuente en la vecindad del requerimiento r . Luego entra en un ciclo que se ejecuta mientras existan candidatos a evaluar, en el cual se calcula el soporte de cada candidato y se definen como itemsets frecuentes aquellos que verifican el soporte mínimo s , y a partir de estos itemsets frecuentes obtenidos se generan los nuevos candidatos.

Para implementar este algoritmo utilizamos una estructura de datos compleja denominada hash-tree para tratar en memoria los candidatos y los itemsets frecuentes hallados en cada pasada del algoritmo.

5.6.2 Clustering de Itemsets Frecuentes

Una vez obtenidos los itemsets frecuentes, en esta etapa se generan los grupos o clusters de itemsets frecuentes similares según la medida de similaridad definida en términos de cantidad de palabras y relaciones semánticas entre las mismas en el capítulo 3.

Es aquí donde se utiliza la jerarquía de conceptos *Wordnet* para agrupar aquellos itemsets frecuentes que tienen una misma conceptualización, es decir que tienen conceptos en común. De esta manera, se agrupan juntos aquellos itemsets frecuentes que son más similares en términos de cantidad de palabras y relaciones semánticas entre las mismas, formando clusters de itemsets frecuentes "*cohesivos*", dejando de lado aquellos itemsets frecuentes aislados.

Cada cluster queda determinado por un conjunto de tópicos que conceptualizan a los itemsets frecuentes que pertenecen al cluster. Una vez obtenidos los clusters de itemsets frecuentes de todos los requerimientos del usuario en análisis, se aplica nuevamente el algoritmo de clustering, sobre los clusters obtenidos, para generar una abstracción aún

mayor en el nivel de conceptualización de tópicos. Veamos el pseudocódigo asociado a esta etapa.

Algoritmo Clusterer

```

Input: aItemsets array, r requerimiento, u usuario
Output: aClusters array de clusters
Comienzo
Array aItems, aNearestClusters
Lista lTopics
//Recorer los itemsets y crea clusters de 1 itemset cada uno.
Para cada i ∈ aItemsets hacer
    //Obtiene los items del itemset
    aItems = getItemsets(i)
    //Obtiene la lista de conceptos de WordNet. Rt Ratio y Ds Profundidad
    //minima son parámetros del sistema
    lTopics = getTopics(Rt,Ds,aItems)
    Crea un cluster c con el itemset i y la lista de tópicos
    Agrega el cluster c a aClusters
Fin Para
// Agrupa clusters según su similaridad
aClusters = findNearestClusters(aClusters)
Mientras existan clusters para agrupar hacer
    // Obtiene los clusters mas cercanos y los agrupa
    aClusters = findNearestClusters(aClusters)
Fin Mientras
Devuelve aClusters
Fin

```

5.6.3 Obtención de Tópicos

Para agrupar los itemsets frecuentes en clusters, utilizamos el algoritmo *getTopics*. Este algoritmo devuelve los padres en común más bajos en la jerarquía de *WordNet* [W30] de los synsets que representan a los ítems recibidos como parámetro. El algoritmo que utilizamos en este proyecto surge del algoritmo de conteo de conceptos de la tesis "*Robust Automated Topic Identification*" [LIN97]. Veamos su pseudocódigo.

Algoritmo getTopics

```

Input: Rt Proporción (RATIOThreshold), Ds Profundidad (STARTINGDepth),
      aItems Array de items, WN Base de Conocimiento Semántica WordNet
Output: lista de tópicos
Comienzo
  // Lista de synsets de interes
  Lista lIList
  // Lista final de synsets de interes
  Lista fiList
  // Obtiene un array con los synsets relacionados a los items
  Array aSynsets = getSynsets(WN,aItems)
  // Arma una jerarquia con los synsets obtenidos y sus ancestros
  propagateParents(WN,aSynsets)
  boolean bGoOn = true
  Mientras bGoOn hacer
    //Propagar los pesos de los synsets activos a sus ancestros
    propagateWeight(aSynsets)
    //Computar la proporción de los synsets activos
    computeRatio(aSynsets)
    //Obtiene los synsets más altos en la jerarquia
    Array aNodosTop = getNodosTop(WN,aSynsets)
    Para cada synset s ∈ aNodosTop hacer
      Si getWeight(s) > 0 entonces
        // Es un synset activo entonces lo procesa
        Si getDepth(s) < Ds entonces
          Agrega los hijos directos del synset s a aNodosTop
        Sino
          Si (getRatio(s) < Rt) y (s no es una hora) y (s ∉ lIList) Entonces
            Agregar s a lIList
          Sino
            Agrega los hijos directos del synset s a aNodosTop
          Fin Si
        Fin Si
      Fin Si
    Fin Para
    Si lIList tiene elementos entonces
      // Obtiene el synset de mayor peso de la lista
      Synset s = getNodeMaxWeight(lIList)
      Si (s ∉ fiList) entonces
        Agregar s a fiList
      Fin Si
      Eliminar s de lIList
      // Setea los hijos de s como nodos no activos
      setLeafsWeightInZero(s)
      Sino
        bGoOn = false
      Fin Si
    Fin Mientras
  Devolver fiList
Fin

```

Como vimos en el Capítulo 3, la mínima unidad utilizada por WordNet [W30] son los synsets o conjuntos de sinónimos. Cada synset posee un identificador unívoco (offset), el conjunto de palabras que lo componen y una descripción.

Supongamos que ejecutamos el algoritmo con los siguientes parámetros: $R_t = 0.68$ $D_s = 6$ y $\{pear, apple\}$. El primer paso del algoritmo es hallar los synsets que representan a cada uno de los ítems recibidos como parámetro, invocando al método *getSynsets*. Por ejemplo, al invocar a este método con los ítems *Apple* y *Pear* se obtienen:

- Dos synsets para *Apple*:

- ✓ Synset con offset 05775435 compuesto únicamente por la palabra *Apple* y cuya descripción es *"fruit with red or yellow or green skin and sweet to tart crisp whitish flesh"*.
- ✓ Synset con offset 08933828 compuesto por las siguientes palabras: *Apple*, *Orchard apple tree* y *Malus pumila* y cuya descripción es *"native Eurasian tree widely cultivated in many varieties for its firm rounded edible fruits"*
- Dos synsets para *Pear*:
 - ✓ Synset con offset 05803355 compuesto únicamente por la palabra *Pear* y cuya descripción es *"sweet juicy gritty-textured fruit available in many varieties"*.
 - ✓ Synset con offset 08951407 compuesto por las palabras: *Pear*, *Pear tree* y *Pirus communis* y cuya descripción es *"Old World tree having sweet gritty-textured juicy fruit; widely cultivated in many varieties"*

El segundo paso del algoritmo es obtener todos los synsets padres de los synsets obtenidos en el paso anterior. Este paso lo realiza el método `propagateParents` y si lo invocamos con los 4 synsets del ejemplo anterior obtenemos la siguiente jerarquía de synsets (Figura 5.5).

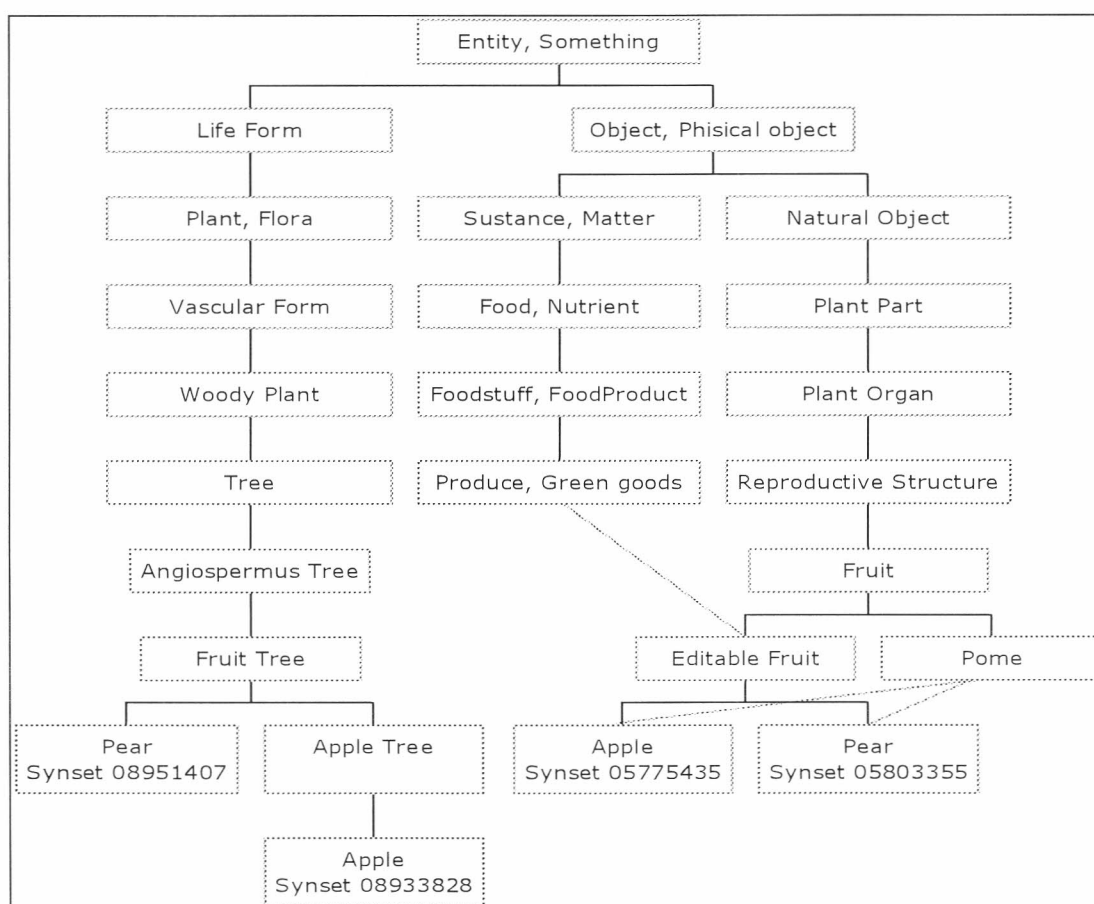


Figura 5.5: Jerarquía de Synsets para *Apple* y *Pear*

El siguiente paso del algoritmo es procesar todos los synsets obtenidos, para ello debe calcular el peso y la proporción de cada uno de los synsets. El cálculo del peso lo realiza el método `propagateWeight` y se basa en la Definición 3.6 y el cálculo de la proporción lo realiza el método `calculateRatio`, que se basa en la Definición 3.5.

Método propagateWeight

```

Input: aSynsets array de Synsets
Output: -
Comienzo
Entero sumaPesos = 0
    // Recorre los synsets recibidos como parámetros
    Para todo s ∈ aSynset hacer
        // Calcula el peso de s y lo setea
        setWeight(s, s.getPlusWeightLeafSons())
    Fin Para
    //Llamada recursiva con los padres
    propagateWeight(getParents(s))
Fin

```

Método getPlusWeightLeafSons

```

Input: synset s
Output: Entero correspondiente al peso de synset
Comienzo
Entero sumaPesos = 0
    Si s no es una hoja entonces
        // Obtiene los hijos directos del synset s
        Array aSons = getChildrens(s)
        Para todo h ∈ aSons hacer
            Si h es una hoja entonces
                sumaPesos = sumaPesos + getWeigth(h)
            Sino
                sumaPesos = sumaPesos + getPlusWeightLeafSons(h)
        Fin Si
    Fin Para
    Sino
        sumaPesos = getWeight(s)
    Fin Si
    Devolver sumaPesos
Fin

```

Veamos el código del método *computeRatio* que se aplica a todos los synsets obtenidos.

Método computeRatio

```

Input: aSynsets array de Synsets
Output: -
Comienzo
    Para todo s ∈ aSynset hacer
        // calcula el ratio y se lo setea a s invocando al método computeRatio
        // (polimorfismo) que recibe como parámetro un synset
        computeRatio(s)
    Fin Para
Fin

```

Método computeRatio

```

Input:   s Synset
Output:  Double dRatio
Comienzo
//obtiene el máximo peso de los hijos del synset s
Double dMaxWeightSon = s.getMaxWeightSons()
//obtiene la suma de todos los pesos de los hijos del synset s
Double dPlusRatioSons = s.getPlusWeightLeafSons();
Double dRatio = 0;
Si s.getWeight() > 0 entonces
    // Es un nodo activo entonces el ratio es el peso de la hoja
    Si s es una hoja entonces
        dRatio = getWeight(s)
    Sino
        Si dPlusRatioSons > 0 entonces
            dRatio = dMaxWeightSon / dPlusRatioSons
        Fin Si
    Fin Si
Fin Si
Devolver dRatio
Fin

```

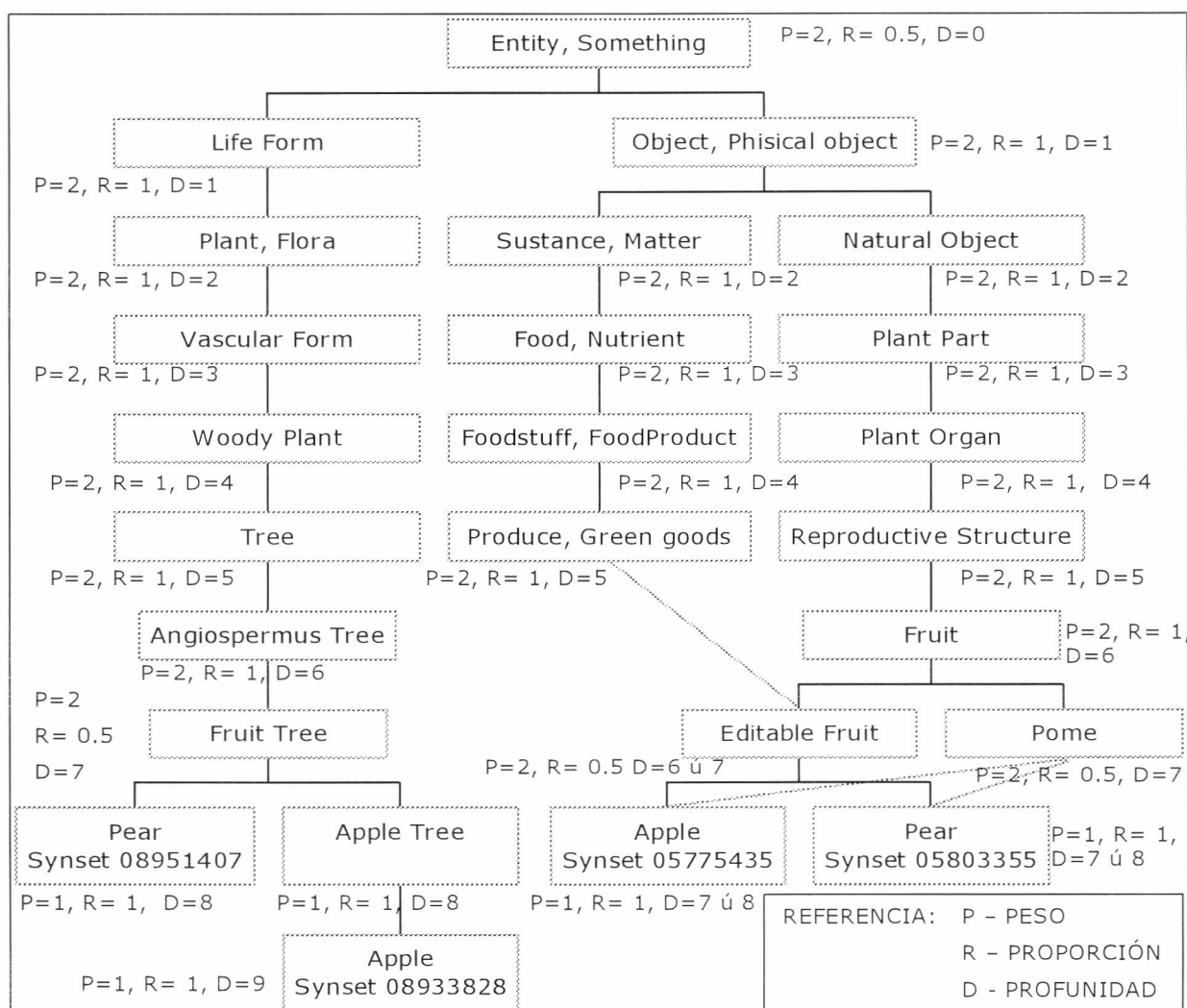


Figura 5.6: Cálculo del Peso, Proporción y Profundidad de los nodos

Veamos el valor del peso, la profundidad y la proporción de los nodos (synsets) de la jerarquía de la Figura 5.6.

El siguiente paso del algoritmo *getTopics* es invocar al método *getNodosTop* para obtener los nodos o synsets más arriba en la jerarquía actual. En el caso particular del ejemplo de la figura 5.6, se obtiene en la primera pasada el synset *Entity*.

Luego como el synset *Entity* tiene peso (2) y profundidad (0), el algoritmo obtiene sus hijos directos (*Life Form*, *Object* y *Physical Object*) y continua ejecutándose. Finalmente el resultado del algoritmo es la lista *fiList* con los siguientes synsets: *Fruit Tree* y *Fruit*.

Si observamos la Figura 5.6, vemos que estos synsets no son los padres en común más bajos de la jerarquía para *Apple* y *Pear*. La razón por la cual no se obtuvieron está relacionada con los parámetros con los cuales invocamos al algoritmo ($R_t = 0.68$ y $D_s = 6$), si volvemos a ejecutar el algoritmo con $R_t = 0.68$ y $D_s = 7$, obtendremos los siguientes synsets: *Fruit Tree* y *Pome*, que sí son los padres más bajos en la jerarquía.

En nuestro proyecto necesitamos obtener siempre los padres más bajos de la jerarquía, aca surge la primer modificación que realizamos al algoritmo. Nuestra modificación consiste en obtener la máxima profundidad (*MaxDepth*) de la jerarquía relacionada a los items actuales, en este caso sería 10 (nodo de *Apple*, Figura 5.6) e invocar recursivamente al algoritmo incrementando en cada llamada la profundidad en 1 hasta obtener los padres en común más bajos en la jerarquía, teniendo en cuenta en cada llamada que la profundidad sea menor *MaxDepth* - 1.

Después de realizar esta modificación, ejecutamos nuestro algoritmo *getTopics* con $R_t=0.68$ y $D_s=2$ y obtuvimos *Fruit Tree* y *Pome* y con el algoritmo original de la tesis "Robust Automated Topic Identification"[LIN97] obtuvimos: *Fruit* y *Fruit Tree*.

Otra de las modificaciones que realizamos al algoritmo esta relacionado con *Clustering*. El primer paso del algoritmo de *Clustering* es generar un cluster con cada uno de los itemsets obtenidos con el algoritmo de *Association Rules* y obtener los tópicos relacionados a cada uno de los itemsets con el algoritmo *getTopics*. Si se invoca al algoritmo original con un solo item, éste no devolverá nada ya que su función es devolver los padres en común más bajos de la jerarquía de un conjunto de ítems. Para nuestro propósito específico necesitamos obtener la lista de tópicos que representan a este ítem, por lo tanto modificamos nuestro algoritmo *getTopics* para que devuelva la lista de synsets que contienen al item.

Por ejemplo, si invocamos al algoritmo original con *Apple* la lista de tópicos resultante estará vacía y si invocamos a nuestro algoritmo obtendremos los synsets que contienen al item *Apple*:

- ✓ Synset con offset 05775435 compuesto únicamente por la palabra *Apple*.
- ✓ Synset con offset 08933828 compuesto por las siguientes palabras: *Apple*, *Orchad apple tree* y *Malus pumila*

Otro caso particular que detectamos en el cual el algoritmo original de la tesis "Robust Automated Topic Identification"[LIN97] obtiene una lista de tópicos vacía es cuando se lo invoca con items de una misma rama. Por ejemplo, si invocamos a este algoritmo con $R_t = 0.68$, $D_s = 7$ y $\{tree, woody plant\}$ obtendremos una lista vacía.

Para resolver este problema decidimos agregar otro algoritmo que sólo se ejecute cuando nuestro algoritmo devuelva una lista de tópicos vacía. Este nuevo algoritmo verifica si existe una relación *padre-hijo*, entre los items que está analizando y en caso de existir devuelve el hijo, ya que consideramos que es más específico que el padre (generalización de sus hijos).

Continuando con el ejemplo anterior, nuestro nuevo algoritmo devuelve una lista conteniendo al synset *Tree*. En este ejemplo invocamos a nuestro algoritmo con 2 items. Veamos que sucede cuando lo invocamos con más de 2 elementos de una misma rama, por

ejemplo con {*tree*, *vascular form*, *fruit tree*}. En este caso obtenemos una lista con un único synset, *fruit tree*, que es el mas bajo en la jerarquía.

Finalmente nuestro algoritmo *getTopics* es el siguiente:

```
Algoritmo getTopics (MODIFICADO)
Input:  Rt Proporción (RATIOThreshold), Ds Profundidad (STARTINGDepth),
        aItems Array de items, WN Base de Conocimiento Semántica WordNet
Output: lista de tópicos
Comienzo
  Lista liList      // Lista de synsets de interes
  Lista fiList      // Lista final de synsets de interes
  Array aSynsets = getSynsets(WN,aItems) //obtiene los synsets de los items
  Si (length(aItems) == 1)
    Agregar todos los synset de aSynsets a fiList
    Devolver fiList
  Fin Si
  // Arma una jerarquía con los synsets obtenidos y sus ancestros
  propagateParents(WN,aSynsets)
  boolean bGoOn = true
  Mientras bGoOn hacer
    //Propaga los pesos de los synsets activos a sus ancestros
    propagateWeight(aSynsets)
    computeRatio(aSynsets) //Computa la proporción de los synsets activos
    //Obtiene los synsets más altos en la jerarquía
    Array aNodosTop = getNodosTop(WN,aSynsets)
    Para cada synset s ∈ aNodosTop hacer
      Si getWeight(s) > 0 entonces // Es un synset activo → lo procesa
        Si getDepth(s) < Ds entonces
          Agrega los hijos directos del synset s a aNodosTop
    Sino
      Si (getRatio(s) < Rt) y s no es una hoja y s ∉ liList entonces
        Agregar s a liList
      Sino
        Agrega los hijos directos del synset s a aNodosTop
    Fin Si
  Fin Si
  Fin Para
  Si liList tiene elementos entonces
    s = getNodeMaxWeight(liList) // Obtiene el synset de mayor peso
    Si (s ∉ fiList) entonces
      Agregar s a fiList
    Fin Si
    Eliminar s de liList
    // Setea los hijos de s como nodos no activos
    setLeafsWeightInZero(s)
  Sino bGoOn = false
  Fin Si
  Fin Mientras
  // Obtiene la maxima profundidad de los synsets actuales y le resta 1
  Entero iMaxDepth = getMaxDepth(aSynsets) - 1
  Si (fiList tiene elementos) y (Ds < iMaxDepth) entonces
    // Busca tópicos en común más abajo en la jerarquía
    Lista lTopics = getTopics(Rt,DS+1,aItems)
    Si lTopics tiene elementos entonces
      fiList = lTopics
    Fin Si
  Fin Si
  Devolver fiList
Fin
```

En el algoritmo resaltamos las modificaciones que incorporamos respecto del algoritmo original de la tesis "*Robust Automated Topic Identification*"[LIN97].

Como mencionamos antes, agregamos un nuevo algoritmo al que denominamos *choiceTopics*, este algoritmo obtiene los tópicos correspondientes a varios tems ubicados en la misma rama de la jerarquía de *WordNet* [W30] y lo invocamos en el algoritmo de *Clustering* para calcular la similaridad entre los ítems de 2 itemsets.

Algoritmo choiceTopics

```

Input: aItems1 array de items, aItems2 array de items,
      WN Base de Conocimiento Semántica WordNet
Output: lList lista de tópicos
Comienzo
  //Obtiene un array con los synsets relacionados a los items de aItems1
  Array aSynsets1 = getSynsets(WN,aItems1)
  // Arma una jerarquia con los synsets obtenidos y sus ancestros
  propagateParents(WN,aSynsets1)
  // Obtiene un array con los synsets relacionados a los items de aItems2
  Array aSynsets2 = getSynsets(WN,aItems2)
  // Arma una jerarquia con los synsets obtenidos y sus ancestros
  wn.propagateParents(aSynsets2)
  Lista lTemporal
  Para cada synset sSon ∈ aSynsets1 hacer
    Para cada synset sParent ∈ aSynsets2 hacer
      Si (sSon es distinto de sParent) Entonces
        Si (sSon es hijo de sParent) Entonces
          Agregar sSon a lTemporal
        Sino
          Si (sParent es hijo de sSon) Entonces
            Agregar sParent a lTemporal
          Fin Si
        Fin Si
      Fin Si
    Fin Para
  Obtener el synset de mayor profundidad de lTemporal y agregarlo a lList
Fin Para
Devolver lList
Fin

```

Continuemos con los métodos invocados en el método *Clusterer*. Veamos primero el pseudocódigo asociado al método *findNearestClusters*.

Método findNearestClusters

```

Input: aClusters Array de clusters
Output: aNewClusters Array con los clusters más cercanos
Comienzo
Float fClusterSimilarity
  Para c1, c2 clusters ∈ aClusters hacer
    // Calcula la similaridad entre los clusters
    fClusterSimilarity = computeClustersSimilarity(c1, c2)
    // MinSimilarity es la similaridad mínima permitida.+
    // Es un parámetro del sistema
    Si fClusterSimilarity > MinSimilarity entonces
      Limpiar aNewClusters
      Agregar c1 y c2 a aNewClusters
    Fin Si
  Fin Para
  Devolver aNewClusters
Fin

```

Veamos como se calcula la similaridad semántica entre dos clusters.

Método computeClusterSimilarity

```

Input:   $c_1$  y  $c_2$  Clusters
Output: fOut float especificando la similaridad entre 2 clusters
Comienzo
Float fSimilarity = 0
  Para cada itemset  $i_1 \in c_1$  hacer
    Para cada itemset  $i_2 \in c_2$  hacer
      fSimilarity = fSimilarity + computeItemsetsSimilarity( $i_1$ ,  $i_2$ )
    Fin para
  Fin para
Si fSimilarity > 0 Entonces
  fOut =  $1 - ((\text{cantItemsets}(c_1) + \text{cantItemsets}(c_2)) / \text{fSimilarity})$ 
Fin Si
Devolver fOut
Fin

```

Este método invoca al método *computeItemsetsSimilarity* que calcula la similaridad semántica entre los itemsets que componen los clusters basándose en la similaridad semántica de los ítems que componen los itemsets y generalizando estos ítems con el algoritmo *getTopics*.

Las fórmulas implementadas para calcular las similaridades semánticas entre clusters, itemsets e ítems corresponden a las Definiciones 3.7, 3.8 y 3.9.

Método computeItemsetsSimilarity

```

Input   $i_1$ ,  $i_2$  Itemsets
Output: fOut float especificando la similaridad semántica entre 2 itemsets
Comienzo
Array aItems
float
  //obtiene los items de los itemsets
  aItems = getItemsets( $i_1$ )  $\cup$  getItemsets( $i_2$ )
  //obtiene los tópicos que generalizan los items
  Lista lTopics = getTopics(Rt, DS, aItems)
  Si lTopics está vacía Entonces
    // Obtiene los tópicos de los items que están en las mismas ramas
    lTopics = choiceTopics(getItemsets( $i_1$ ), getItemsets( $i_2$ ))
  Fin Si
  Si lTopics no está vacía Entonces
    //calcula la similaridad semántica
    fOut = size(lTopics) + depth(lTopics)
  Fin Si
  //calcula la similaridad de palabras y le suma la similaridad semántica
  fOut = fOut + wordsSimilarity( $i_1$ ,  $i_2$ )
  Devolver fOut
Fin

```

La similaridad de palabras entre dos itemsets se calcula de la siguiente manera:

Método wordsSimilarity

```

Input:  i1, i2 Itemsets
Output: fOut float especificando la similaridad de palabras entre 2 itemsets
Comienzo
Entero iIntersect, iTotalItems
Float fOut
    //calcula la cantidad de items en común
    iIntersect = Size(getItems(i1) ∩ getItems(i2))
    iTotalItems = size(i1) + size(i2)
    //Calcula la similaridad de palabras entre los itemsets
    fOut = 2 * iIntersect / iTotalItems
Devolver fOut
Fin

```

El método *computeClustersWeight* que calcula el peso de todos los clusters obtenidos para el usuario utiliza la Definición 3.10 del capítulo 3.

Método computeClustersWeight

```

Input:  aClusters Array de Clusters , u Usuario
Output: -
Comienzo
int iTotalPages, iClusterPages
Cluster c
float dWeight = 0
    //Obtiene la cantidad de páginas a las que accedió el usuario u
    iTotalPages = getCantRequerimientos(u, Preprocesado)
    //Recorre los clusters calculando su peso
    Para cada c ∈ aClusters hacer
        // Obtiene la cantidad de requerimientos correspondientes al cluster
        iClusterPages = getPagesCount(c)
        //calcula el peso del cluster
        dWeight = iClusterPages / iTotalPages
        //Setea el peso en el cluster
        setWeight(c, dWeight)
    Fin Para
Fin

```

El peso de los clusters se emplea para asignar pesos a las distintas categorías resultantes para el usuario en análisis. Un cluster con peso alto significa que una gran cantidad de las páginas accedidas por el usuario caen en él, lo cual nos da la pauta de que la categoría resultante de dicho cluster tiene mucho peso en el perfil del usuario.

5.6.4 Selección del tópico de mayor Bondad

Por cada cluster resultante de la etapa anterior se establece cuál es el tópico, entre todos los tópicos que caracterizan al cluster, que mejor lo describe. Para ello implementamos un algoritmo que calcula la bondad de un tópico para un cluster y seleccionamos aquel tópico con mayor bondad.

El algoritmo *computeBondad* utiliza las definiciones de frecuencia de un tópico en un cluster (Definición 3.11), frecuencia relativa de un tópico en un cluster (Definición 3.12) y bondad de un tópico en un cluster (Definición 3.13).

Algoritmo computeBondad

Input: aClusters Array de Clusters

Output:

Comienzo

float dBondad, dMaxBondad = 0

// Recorre los clusters calculando el topico de mayor bondad

Para cada c ∈ aClusters hacer

//obtiene los tópicos del cluster

Lista lTopics = getTopicsCluster(c)

// Recorre los tópicos del cluster

Para cada t ∈ lTopics hacer

//calcula la bondad del tópico t en el cluster c

dBondad = frecuenciaRelativa(c,tópico) *

(frecuenciaRelativa(c,tópico)/

sumatoriaFrecuenciasRelativas(c,tópico,aClusters))

Si dBondad > dMaxBondad entonces

tópicoMaxBondad = tópico

Fin Si

Fin Para

//setea la bondad en el cluster

setTopicoMayorBondad(c,tópicoMaxBondad)

Fin Para

Fin

El método *computeBondad* hace uso de la función *sumatoriaFrecuenciasRelativas* que acumula la frecuencia relativa del tópico recibido por parámetro en el resto de los clusters. Veamos el pseudocódigo del cálculo de la frecuencia relativa de un tópico en un cluster.

Método frecuenciaRelativa

Input: c Cluster y sTopico Tópico

Output: float indicando la frecuencia relativa del tópico sTopico en el cluster

Comienzo

float dFreqRelativa = 0

int iFreq = 0

//obtiene todos los tópicos del cluster

Lista lTopics = getTopicsCluster(c)

Para cada t ∈ lTopics hacer

// Acumula la frecuencia de los demás topicos del cluster

Si t <> sTopico entonces

iFreq = iFreq + frecuencia(c,t)

Fin Si

Fin Para

Si iFreq > 0 entonces

dFreqRelativa = frecuencia(c,sTopico) / iFreq

Fin Si

Devolver dFreqRelativa

Fin

Método frecuenciaInput: c Cluster, $sTopico$ TópicoOutput: entero indicando la frecuencia del tópico $sTopico$ en el cluster c

Comienzo

int $iFreq = 0$ Array $aItemsets = getItemsets(c)$

// Recorre todos los itemsets del cluster

Para cada itemset e $aItemsets$ hacer

//obtiene los items del itemset

Array $aItems = getItemset(itemset)$

// Recorre todos items del itemset

Para cada item e $aItems$ hacer//verifica si $sTopics$ es pariente del item en la jerarquia de WordNetSi $isParent(item, sTopico)$ $iFreq = iFreq + 1$

Fin Si

Fin Para

Fin Para

Devolver $iFreq$

Fin

Ejemplo 5.3. Supongamos que los tópicos del cluster c compuesto por los itemsets $I_1=\{Apple\}$ y $I_2=\{Pear\}$ son $t_1=Fruit Tree$ y $t_2=Pome$ (ver figura 5.6).

$$computeBondad(c) = \max\{ (fR(c, t_1) * (fR(t_1) / sFR(c, t_1))), (fR(c, t_2) * (fR(c, t_2) / sFR(c, t_2))) \} = \\ = \max\{1, 1\} = 1$$

con fR = frecuenciaRelativa y sFR = sumatoriaFrecuenciasRelativas.

Tanto t_1 como t_2 son buenos descriptores del cluster c . Esto se debe a que los tópicos del cluster verifican ser padres de todos los ítems que componen al cluster.

5.6.5 Mapeo Tópicos-Usuarios

En función de los clusters y tópicos obtenidos en la etapa anterior, cada uno de los tópicos que caracteriza un cluster define una categoría del usuario en análisis. El peso de dicha categoría se establece en función del peso del tópico correspondiente. En esta etapa se actualiza la base de datos almacenando las categorías resultantes para cada usuario en análisis, cada una con su respectivo peso.

El mapeo de usuarios consiste en asignar a los usuarios los tópicos que describen a los clusters obtenidos a partir de los requerimientos de cada uno de ellos.

5.7 Resumen

En este capítulo detallamos como fue implementada cada una de las diferentes etapas de *SWMiner*, presentamos el pseudo código asociado a cada una de ellas y describimos nuestro algoritmo de obtención de tópicos (*getTopic - Modificado*) especificando las diferencias de éste con el algoritmo *getTopic* de la tesis *Robust Automated Topic Identification* [LIN97].

CAPITULO 6: Experimentación

6.1 Introducción

En el presente capítulo detallamos los experimentos realizados con *SWMiner*, mostramos los resultados obtenidos en cada una de las etapas y los comparamos con los resultados obtenidos con *TOPIC* [W36], con el Directorio *Google* [W37, W38] y con una categorización realizada a mano.

6.2 Experimentación

Para cada uno de los logs con los cuales realizamos los experimentos, se efectuó un seguimiento de todos los requerimientos obtenidos en la etapa de depuración, así como también de todos los *IN* y *OUT* links obtenidos en las vecindades de los requerimientos. Luego se evaluó la performance y eficacia de las siguientes etapas del sistema:

- Identificación de Idiomas
- Part of Speech
- Lemmatiser
- Clustering e Identificación de Tópicos

Se compararon los resultados obtenidos con *SWMiner* contra los resultados obtenidos con *TOPIC* [W36], con el Directorio *Google* [W37, W38] y con una categorización realizada a mano.

6.2.1 Identificación de Idiomas

Para medir la performance del algoritmo de identificación de idiomas de Cavnar y Trenkle [W21, CT94, CAV94, W22], utilizamos la Definición 3.18 [*Hits, Misses y Mistakes para Identificación de Idiomas*].

Comparamos el idioma obtenido por el algoritmo para cada una de las páginas analizadas contra el idioma real de las mismas. El conjunto de páginas analizadas en cada experimento corresponde al conjunto formado por todos los requerimientos junto con todos los *IN* y *OUT* links en la vecindad de los mismos.

6.2.2 Part Of Speech

Para medir la performance del algoritmo de *Part Of Speech* [W31, W32], utilizamos la Definición 3.20 [*Hits, Misses y Mistakes para Part Of Speech*].

Tomamos un requerimiento en forma aleatoria de entre todos los requerimientos de cada log con su vecindad de *IN* y *OUT* links completa; y por cada página correspondiente, analizamos las primeras 100 palabras extraídas en la etapa de preprocesamiento de documentos web, comparando los sustantivos identificados por el algoritmo contra los sustantivos identificados manualmente.

6.2.3 Lemmatiser

Para verificar la performance del algoritmo de *Lemmatiser* [W26, W27, W28], utilizamos la definición 3.19 [*Hits, Misses y Mistakes para Lemmatiser*].

Analizamos todos los sustantivos extraídos de cada una de las páginas involucradas en la evaluación de Part of Speech, y comparamos los lemas obtenidos por el algoritmo de *Lemmatiser* contra los lemas identificados manualmente.

6.2.4 Clustering e Identificación de Tópicos

Para cada una de las páginas involucradas en el análisis de un log se extrae una cantidad X de palabras (entre todos los sustantivos identificados anteriormente) de los tags *Html* considerados con mayor peso, las cuales representan a la página. X es configurable, ya que es uno de los parámetros del *SWMiner*.

Cabe aclarar que no siempre se obtiene la cantidad especificada X de palabras por páginas ya que muchas de las palabras son descartadas por no ser consideradas sustantivos por el algoritmo de *Part Of Speech* o por ser palabras de la lista de *Stopwords* o por no existir en WordNet. Es muy importante la existencia en WordNet de las palabras con las cuales se trabaja, ya que *SWMiner* utiliza esta jerarquía semántica para obtener los tópicos que conceptualizan a las palabras contenidas en los itemsets frecuentes.

Los conjuntos de palabras extraídas representan las transacciones sobre las cuales se aplica el algoritmo de *Association Rules* [GON96, ASI93, AS94] para obtener los itemsets de palabras frecuentes que aparecen en el conjunto de páginas en análisis de un usuario formado por los requerimientos y sus vecindades.

El algoritmo de *Clustering* aplica las Definiciones 3.7 [Similaridad de palabras entre Itemsets], 3.8 [Similaridad Semántica entre Itemsets] y 3.9 [Similaridad entre Clusters] y utiliza la *cota de Mínima Similaridad entre Clusters TOP*(parámetro del sistema) para determinar cuando debe agrupar los clusters.

Para obtener los tópicos que finalmente describen a cada cluster resultante utilizamos las Definiciones 3.11 [Frecuencia de un Tópico en un Cluster], 3.12 [Frecuencia Relativa de un Tópico en un Cluster] y 3.13 [Bondad de un Tópico en un Cluster].

6.2.5 SWMiner vs. Google

El Directorio *Google* [W37] integra la sofisticada tecnología de búsqueda de *Google* con las páginas del *Open Directory* (Directorio Abierto) [W38]. El *Open Directory* es un amplio directorio público dirigido por *Netscape*, creado y mantenido por un grupo mundial de redactores voluntarios que evalúan sitios para su inclusión en el directorio. Las páginas web seleccionadas por estos redactores son organizadas dentro de una serie de amplias categorías debajo de las que existen muchas sub-categorías más específicas. El *Open Directory* es la más amplia base de datos de contenido web distribuida y clasificada por humanos. *Google*, así como también *Lycos* y *HotBot* entre otros, utiliza esta jerarquía como la base de su propio directorio.

Para medir la eficacia del algoritmo de identificación de tópicos, utilizamos la Definición 3.21 [Hits, Misses y Mistakes para Identificación de Tópicos]. Comparamos las categorías obtenidas por *SWMiner* para un usuario determinado del log en análisis, contra las categorías asignadas al conjunto de páginas web accedidas por dicho usuario en el Directorio *Google*. *Google* establece para cada página web una jerarquía de categorías.

Del mismo modo que las categorías que define *SWMiner* para un usuario se corresponden con los tópicos más bajos en la jerarquía de *WordNet* que cumplen con las condiciones necesarias, por cada página correspondiente a cada requerimiento del usuario decidimos tener en cuenta solamente las 4 últimas categorías de la jerarquía de categorías asignada por *Google* a cada página, ya que las categorías superiores son demasiado genéricas y no aportan información para discriminar el significado semántico de las páginas.

6.2.6 SWMiner vs. TOPIC

Comparamos las categorías asignadas por *SWMiner* al usuario del log en análisis contra los tópicos que se obtienen con la herramienta *TOPIC* (*TO*ronto *P*age *I*nfluence *C*omputation) [W36] para cada uno de los requerimientos del log.

6.2.7 SWMiner vs. Categorías asignadas manualmente

Por último, aplicamos la Definición 3.21 nuevamente para comparar las categorías asignadas por *SWMiner* con las categorías asignadas en forma manual según el contenido de los documentos *Html* correspondientes a los requerimientos accedidos por el usuario del log en análisis.

6.2.8 Experimentos

Con la finalidad de validar la necesidad de algunas etapas definidas en *SWMiner*, sometimos los logs a diferentes experimentos. A saber:

- *SWMiner* con Normalización Morfológica de palabras y con Clustering
- *SWMiner* con Normalización Morfológica de palabras y sin Clustering
- *SWMiner* sin Normalización Morfológica de palabras y con Clustering
- *SWMiner* sin Normalización Morfológica de palabras y sin Clustering

Entendiendo por Normalización Morfológica de palabras a las etapas de aplicación de *Part Of Speech* y *Lemmatiser*.

Los logs utilizados en los distintos experimentos fueron generados en forma manual, conteniendo en su mayoría requerimientos de páginas específicas de animales, dado que la jerarquía de animales se encuentra bien definida en la jerarquía *WordNet*. Los requerimientos que componen el log genérico utilizado en el último experimento, fueron extraídos del sitio web de *TOPIC* [W36] de la página <http://www.cs.toronto.edu/db/topic/examples/known-auth.html> con la finalidad de comparar los resultados obtenidos con *SWMiner* contra los obtenidos con *TOPIC*.

6.3 Experimentos con Log de Mamíferos Marinos

6.3.1 SWMiner con Normalización Morfológica y con Clustering

6.3.1.1 Parámetros de SWMiner

En la tabla 6.3.1 detallamos el valor de los parámetros y/o variables del sistema utilizados en en los siguientes experimentos.

Parámetro	Valor
Cantidad de In Links por Requerimiento	5
Cantidad de Out Links por Requerimiento	5
Cantidad de In Links por Link	2
Cantidad de Out Links por Link	2
Cantidad de Niveles	2
Cantidad de Palabras por Página	50
Soporte Association Rules	0.3
Similaridad Mínima entre Clusters	0.7
WordNet – Ratio	0.68
WordNet – Profundidad	2

Tabla 6.3.1: Parámetros del SWMiner

6.3.1.2 Requerimientos y su Vecindad

En los siguientes experimentos el *Depurador* del *SWMiner* analizó un log del cual obtuvo los 3 requerimientos que se describen en la Tabla 6.3.2.

Requerimientos del Log

Id	Requerimiento
1	http://dkd.net/whales/
2	http://www.whale-web.com/
3	http://www.crru.org.uk/

Tabla 6.3.2: Requerimientos del Log

Uno de los primeros pasos del preprocesador del *SWMiner* es identificar el idioma correspondiente al texto de los requerimientos. Observando el texto de estos 3 requerimientos concluimos que los 3 corresponden al idioma inglés y al ser analizados por el *SWMiner*, éste identificó el idioma correctamente.

Vecindad de los requerimientos

Las Tablas 6.3.3, 6.3.4 y 6.3.5 y las figuras 6.3.1, 6.3.2 y 6.3.3 muestran los links y out links obtenidos para cada uno de los requerimientos anteriores.

Id Link	Id Padre	Tipo	URL
4		IN	http://www.clearlight.com/%7Ekid/whales/hbwhales.html
5	4	IN	http://dkd.net/services/index.html
6	4	IN	http://dkd.net/music/tune35.html
9		IN	http://techcenter.davidson.k12.nc.us/spring027/resourcesHP.htm
10		IN	http://www.omanforum.com/index.cfm/fuseaction/thread/CFB/1/TID/53857/StartRow/31.cfm
11		IN	http://www.boatingoz.com.au/boatinglinks.htm
12	11	IN	http://www.boatingoz.com.au/newschar.htm
13	11	IN	http://www.boatingoz.com.au/newsgen.htm
15		IN	http://www.infooperak.com/index.htm
16	15	IN	http://www.infooperak.com/assomain.htm
17	15	IN	http://www.infooperak.com/rts.htm
18		OUT	http://dkd.net/whales/hbwdolph.html
19	18	OUT	http://dkd.net/whales/hbwhb.html
20	18	OUT	http://dkd.net/whales/index.html
21		OUT	http://dkd.net/whales/stranded.html
22	21	OUT	http://dkd.net/whales/hbhback.html
23	21	OUT	http://dkd.net/whales/hbwbehav.html
24		OUT	http://dkd.net/whales/wsounds.html
25	24	OUT	http://dkd.net/whales/hbminke.html

26	24	OUT	http://dkd.net/whales/oceania.html
27		OUT	http://dkd.net/whales/hbwboats.html
28	27	OUT	http://www.oceania.org.au/expedition/expedition.html
29	27	OUT	http://whaleplanet.net/herveybay
30		OUT	http://dkd.net/whales/postcard/index.html

Tabla 6.3.3: In y Out Links del Requerimiento 1

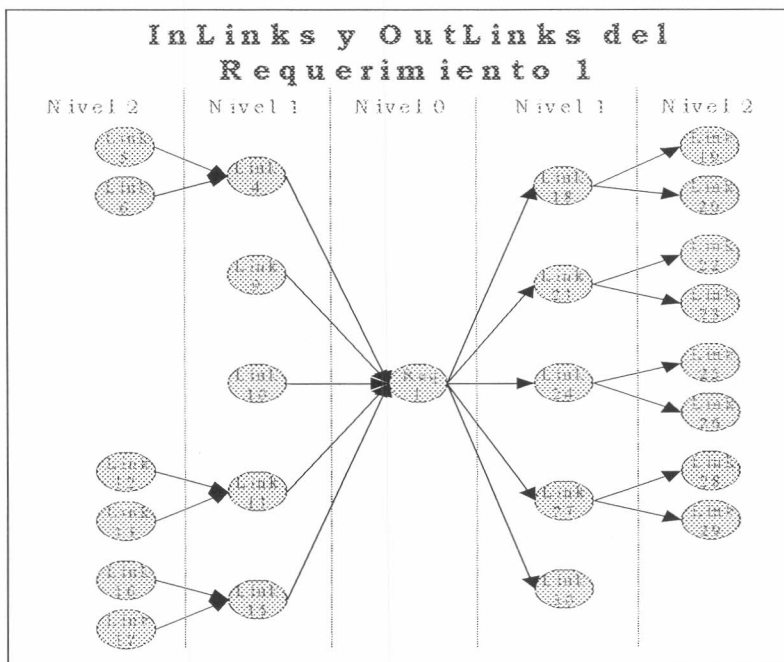


Figura 6.3.1: In y Out Links del Requerimiento 1

Id Link	Id Padre	Tipo	URL
31		IN	http://members.aol.com/spot645/KillerWhalesinCaptivity.html
32		IN	http://www.andrea-schroeder.com/Orcae.html
33		IN	http://www.physics.helsinki.fi/whale/whalcons.html
34	33	IN	http://www.physics.helsinki.fi/whale/reflect.html
35	33	IN	http://www.physics.helsinki.fi/whale/comics/seti/archseti.html
36		IN	http://www.orcahome.de/links.htm
37	36	IN	http://www.more.net/%7Edenise/whales/explore.htm
38	36	IN	http://www.birgitmeyer.com/Delphine/linksdelphine.html
40		IN	http://www.tomigaya.shibuya.tokyo.jp/lilly/linklibrary.html
41	40	IN	http://www.tomigaya.shibuya.tokyo.jp/%7Ebigtwin/BTX/linkpage.html
43		OUT	http://www.whale-web.com/finna.html
44		OUT	http://www.whale-web.com/whales/index.html
45	44	OUT	http://www.whale-web.com/whales/blue.html
46	44	OUT	http://www.whale-web.com/whales/fin.html

47		OUT	http://www.whale-web.com/dolphins/index.html
48	47	OUT	http://www.whale-web.com/dolphins/ahd.html
49	47	OUT	http://www.whale-web.com/dolphins/bottlenose.html
50		OUT	http://www.whale-web.com/porpoises/index.html
51	50	OUT	http://www.whale-web.com/porpoises/vaquita.html
52	50	OUT	http://www.whale-web.com/porpoises/harbor.html
53		OUT	http://www.whale-web.com/resources/index.html
54	53	OUT	http://www.whale-web.com/resources/scinames.html
55	53	OUT	http://www.whale-web.com/resources/classification.html

Tabla 6.3.4: In y Out Links del Requerimiento 2

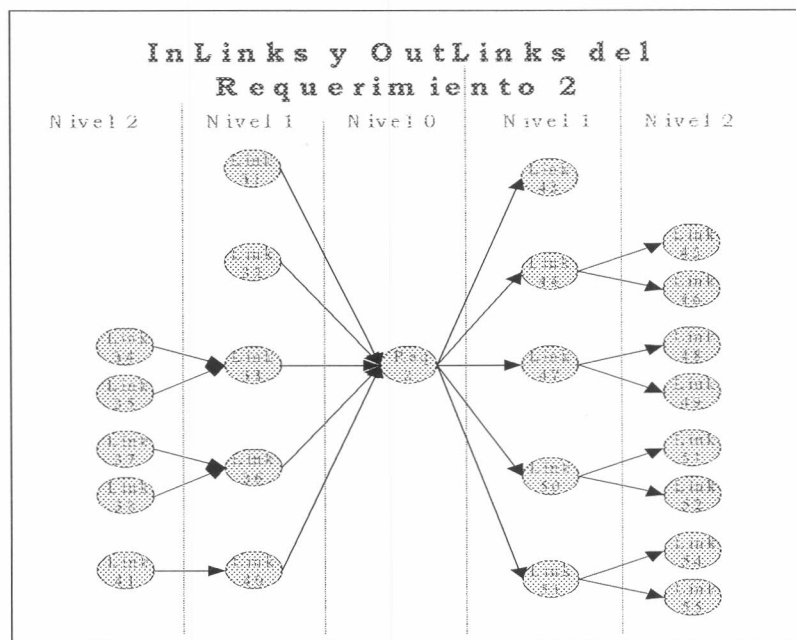


Figura 6.3.2: In y Out Links del Requerimiento 2

Id Link	Id Padre	Tipo	URL
56		IN	http://www.crru.org.uk/humpback.htm
57	56	IN	http://ourworld.compuserve.com/homepages/BMLSS/news2001.htm
59	56	IN	http://ourworld.compuserve.com/homepages/BMLSS/Cetacea.htm
61		OUT	http://www.crru.org.uk/who.htm
62	61	OUT	http://www.bdmir.org.uk
63	61	OUT	http://search.freefind.com/find.html?id=210238&m=0&p=0
64		OUT	http://www.crru.org.uk/moray.htm
65	64	OUT	http://www.crru.org.uk/sunset.htm
66	64	OUT	http://www.crru.org.uk/sponsors.htm
67		OUT	http://www.crru.org.uk/training.htm
68	67	OUT	http://www.bdmir.org.uk/pages/handbook.html
69	67	OUT	http://www.bdmir.org.uk/pages/member.html

70		OUT	http://www.crru.org.uk/research.htm
74	70	OUT	http://cgi.europiukes.net/html_pages/index.shtml
75	70	OUT	http://www.crru.org.uk/prestudies.htm
76		OUT	http://www.crru.org.uk/studies.htm
77	76	OUT	http://www.crru.org.uk/help.htm
78	76	OUT	http://www.crru.org.uk/research.htm#studies

Tabla 6.3.5: In y Out Links del Requerimiento 3

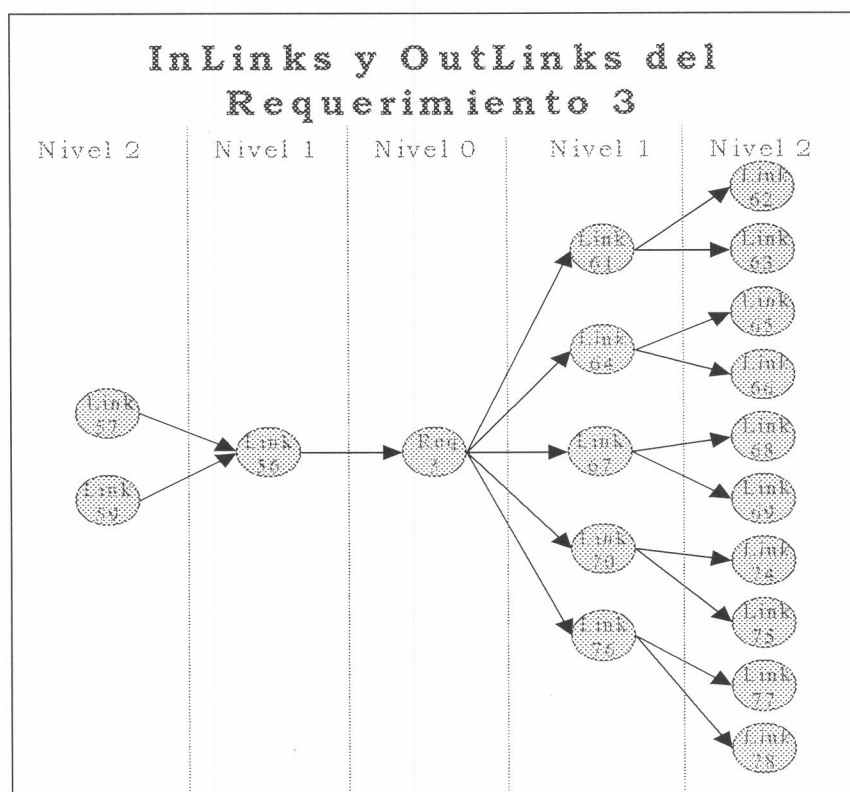


Figura 6.3.3: In y Out Links del Requerimiento 3

Como puede observarse en las Tablas 6.3.3, 6.3.4 y 6.3.5, *SWMiner* no obtuvo la cantidad de in y out links por requerimiento y por link especificados como parámetros del sistema. Analicemos algunos de estos links en detalle para entender porqué sucedió esto.

Lo primero que podemos observar en la tabla 6.3.3 es la ausencia de los links 7, 8 y 14, esto se debe a que, para cada uno de los links, el sistema obtuvo vía ftp archivos html de 0Kb. Al hacer ftp se produjo el siguiente error: "*Socket closed*". En segundo lugar, en la Tabla 6.3.3, podemos observar la ausencia de inlinks para el link con id 9, esto se debe a que Altavista no encontró ninguna página que apunte a dicho link.

Observando la Tabla 6.3.4 detectamos la ausencia de los links 39 y 4. Con estos links sucedió lo mismo que con los links 7, 8 y 14 del requerimiento 1, se obtuvieron archivos html de 0Kb. También notamos que para el link 40 sólo se obtuvo un inlink (link con id 41), esto se debe a que Altavista obtuvo 2 páginas que apuntan a este link y que sólo una de ellas fue considerada por el sistema como escrita en idioma inglés. También notamos que para el link 43 no se obtuvieron outlinks.

El requerimiento 3 es un caso especial, ya que el código html correspondiente a la página <http://www.crru.org.uk/> contiene frames, y *SWMiner* reemplaza la url original por el link indicado en este código como el correspondiente a la página pero sin frames, es decir por <http://www.crru.org.uk/sitemap.htm>. Analizando el contenido de la Tabla 6.3.4 detectamos que para este requerimiento sólo se obtuvo un inlink. El motivo por el cual sucedió esto se debe a que Altavista sólo encontró una página que apunte al requerimiento.

En la Tabla 6.3.5 puede observarse la ausencia de los links 60, 71, 72 y 73, el motivo por el cual sucedió esto es que el sistema los consideró no escritos en inglés, aunque al observar el texto correspondiente a cada uno de ellos podemos ver que están escritos en inglés. Este error lo tuvimos en cuenta al calcular los porcentajes *Precision* y *Recall* correspondientes al algoritmo de identificación de idiomas.

En la Figura 6.3.4 se puede observar que la página correspondiente al link 71 fue escrita en idioma inglés, aunque para el sistema no lo está.



Figura 6.3.4: Página correspondiente al link 71

6.3.1.3 Evaluación del algoritmo de Identificación de Idiomas

Para medir la performance de la metodología de Cavnar y Trenkle [W21, CT94, CAV94, W22] analizamos todos los in y out links correspondientes a los requerimientos 1, 2 y 3. Para completar la Tabla 6.3.6 utilizamos la Definición 3.18 [Hits, Misses y Mistakes para Identificación de idiomas]

Requerimiento	Hits	Misses	Mistakes	Precision	Recall
http://dkd.net/whales/	24	0	0	1,00	1,00
http://www.whale-web.com/	23	1	0	1,00	0,96
http://www.crru.org.uk/	18	4	0	1,00	0,82

Tabla 6.3.6: Precision y Recall para Identificación de Idiomas

Precision - Valor Promedio: 1

Recall - Valor Promedio: 0,93

De estos valores podemos concluir que el 100% de las páginas reconocidas por el algoritmo de identificación de idiomas como escritas en idioma inglés efectivamente lo son y que del total de las páginas analizadas, el algoritmo reconoció correctamente el idioma del 93%. El no haber detectado el 7% de las páginas restantes como escritas en idioma inglés no incrementa el porcentaje de error del sistema, ya que el mismo sólo continúa analizando las páginas que reconoció como escritas en idioma inglés y el idioma del 100 % de éstas fue correctamente reconocido.

6.3.1.4 Evaluación del algoritmo de Part Of Speech

Para analizar la performance del algoritmo de *Part Of Speech* [W31, W32] analizamos 100 de palabras del requerimiento 1 (<http://dkd.net/whales/>) y de cada uno de sus *IN* y *OUT* links. Para completar esta Tabla 6.3.7 utilizamos la Definición 3.20 [*Hits*, *Misses* y *Mistakes* para *Part Of Speech*]

Requerimiento o Link	Hits	Misses	Mistakes	Precision	Recall
1	61	22	17	0,78	0,73
4	60	22	18	0,77	0,73
5	56	26	18	0,76	0,68
6	44	38	18	0,71	0,54
9	53	32	15	0,78	0,62
10	40	42	18	0,69	0,49
11	47	41	12	0,80	0,53
12	57	28	15	0,79	0,67
13	58	28	14	0,81	0,67
15	47	48	5	0,90	0,49
16	52	45	3	0,95	0,54
17	46	51	3	0,94	0,47
18	53	28	19	0,74	0,65
19	60	28	12	0,83	0,68
20	60	26	14	0,81	0,70
21	55	30	15	0,79	0,65
22	52	34	14	0,79	0,60
23	56	22	22	0,72	0,72
24	61	21	18	0,77	0,74
25	54	26	20	0,73	0,68
26	51	28	21	0,71	0,65
27	57	30	13	0,81	0,66
28	65	23	12	0,84	0,74
29	48	27	25	0,66	0,64
30	37	30	33	0,53	0,55

Tabla 6.3.7: Precision y Recall para Part Of Speech

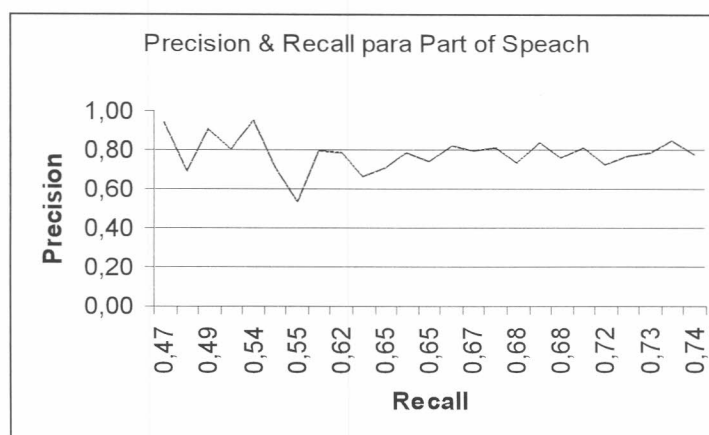


Figura 6.3.5: Precision y Recall para Part Of Speech

Precision - Valor Promedio 0.78

Recall - Valor Promedio 0.63

Estos valores indican que 78% de los sustantivos obtenidos por el algoritmo de *Part of Speech* son correctos y que del total de sustantivos existentes el algoritmo reconoció el 63%.

6.3.1.5 Evaluación del algoritmo de Lemmatiser

Para verificar la performance del algoritmo de Lemmatización [W26, W27, W28] analizamos todos los sustantivos obtenidos del requerimiento 1 (<http://dkd.net/whales/>) y de cada uno de sus *IN* y *OUT* links. Para completar la Tabla 6.3.8 utilizamos la Definición 3.19 [*Hits*, *Misses* y *Mistakes* para Lemmatiser]

Requerimiento o Link	Hits	Misses	Mistakes	Precision	Recall
1	37	18	2	0,95	0,67
4	35	18	2	0,95	0,66
5	40	10	1	0,98	0,80
6	17	3	0	1,00	0,85
9	7	8	0	1,00	0,47
10	371	128	6	0,98	0,74
11	46	29	0	1,00	0,61
12	190	92	5	0,97	0,67
13	184	74	4	0,98	0,71
15	48	24	0	1,00	0,67
16	11	24	0	1,00	0,31
17	9	23	0	1,00	0,28
18	24	9	0	1,00	0,73
19	31	13	0	1,00	0,70
20	36	18	3	0,92	0,67
21	44	9	2	0,96	0,83
22	74	20	1	0,99	0,79

23	71	20	0	1,00	0,78
24	40	11	4	0,91	0,78
25	33	15	0	1,00	0,69
26	63	21	0	1,00	0,75
27	52	19	1	0,98	0,73
28	18	16	1	0,95	0,53
29	21	4	1	0,95	0,84
30	3	2	0	1,00	0,60

Tabla 6.3.8: Precision y Recall para Lemmatiser

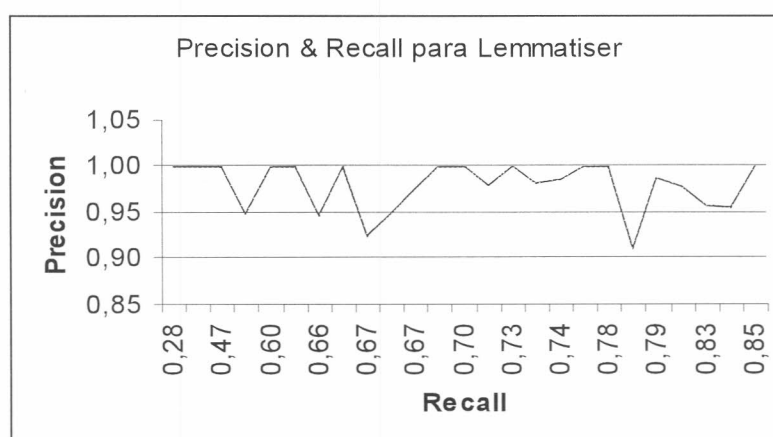


Figura 6.3.6: Precision y Recall para Lemmatiser

Precision - Valor Promedio: 0.98

Recall - Valor Promedio: 0.68

Estos valores indican que el 98% de los lemas obtenidos por el algoritmo de *Lematización* son correctos y del total de palabras analizadas el algoritmo obtuvo los lemas del 68%.

6.3.1.6 Itemsets Frecuentes

En esta sección presentamos los ítemsets frecuentes obtenidos al aplicar el algoritmo de *Association Rules* [GON96, ASI93, AS94] sobre todas los sustantivos que representan a cada uno de las páginas involucradas en este análisis. Las Tablas 6.3.9, 6.3.10 y 6.3.11 describen los itemsets frecuentes obtenidos para cada uno de los requerimientos.

Id Itemset	Soporte	Item
8	0.36000001430511475	RIGHT
22	0.31999999284744263	BOAT QUEENSLAND
42	0.31999999284744263	HUMPBACK QUEENSLAND WHALE WORLD

43	0.31999999284744263	DOLPHIN QUEENSLAND WHALE WORLD
44	0.31999999284744263	DOLPHIN HUMPBACK QUEENSLAND WHALE
45	0.31999999284744263	CETACEAN HUMPBACK QUEENSLAND WHALE
46	0.31999999284744263	HUMPBACK LOGO QUEENSLAND WHALE

Tabla 6.3.9 : Itemsets Frecuentes del Requerimiento 1

Id Itemset	Soporte	Item
53	0.375	WHALE CETACEANS
57	0.375	WHALE QUESTION CORRECTION

Tabla 6.3.10 : Itemsets Frecuentes del Requerimiento 2

En la Tabla 6.3.10 podemos observar que aparece la palabra *Cetaceans*. Esta palabra pertenece al 32 % de las palabras a las cuales el algoritmo *Lemmatiser* no pudo hallarles su lema; en el caso particular de esta palabra el algoritmo debió haber hallado *Cetacean*.

Id Itemset	Soporte	Item
92	0.73684209585189819	DOLPHIN PORPOISE CETACEAN
124	0.73684209585189819	WHALE DOLPHIN CETACEAN RESCUE KILLER UNIT

Tabla 6.3.11 : Itemsets Frecuentes del Requerimiento 3

6.3.1.7 Clustering e Identificación de Tópicos

El algoritmo de *Clustering* se ejecuta una vez para cada requerimiento y una vez más para intentar agrupar los clusters de todos los requerimientos del usuario. Para obtener los clusters el algoritmo aplica las Definiciones 3.7 [Similaridad de palabras entre Itemsets], 3.8 [Similaridad Semántica Inter-Itemsets], 3.9 [Similaridad Intra-Cluster] y 3.10 [Similaridad Inter-Clusters] y el parámetro del sistema *Similaridad Mínima entre Clusters* utilizado por el algoritmo de *Clustering* para determinar si dos clusters deben ser agrupados o no.

En la Tabla 6.3.12 detallamos los diferentes valores de similaridad obtenidos en cada pasada del algoritmo de *Clustering* entre todos los pares de itemsets correspondientes al requerimiento 1 y resaltamos en negrita los itemsets que se agrupan en cada pasada.

Iteración	Itemset 1	Itemset 2	Similaridad
1	8	22	0,833333
		42	0,75
		43	0,6
		44	0,8
		45	0,5
		46	0,75
	22	42	0,538462
		43	0,625
		44	0,806452
		45	0,538462
		46	0,538462
	42	43	0,967078
		44	0,813953
		45	0,46667
		46	0,703704
	43	44	0,813953
		45	0,636364
		46	0,958763
	44	45	0,652174
		46	0,975401
	45	46	0,466667
2	8	22	0,833333
		42	0,75
		43	0,6
		45	0,5
		44-46	0,833333
	22	42	0,538462
		43	0,625
		45	0,538452
		44-46	0,795455
	42	43	0,967078

		45	0,46667
		44-46	0,828571
	43	45	0,636364
		44-46	0,949367
3	45	44-46	0,793103
	8	22	0,833333
		45	0,5
		44-46	0,833333
		42-43	0,769231
	22	45	0,538462
		44-46	0,795455
		42-43	0,689655
	45	44-46	0,793103
		42-43	0,960656
	44-46	42-43	0,954023
4	8	22	0,833333
		44-46	0,833333
		42-43-45	0,764706
	22	44-46	0,795453
		42-43-45	0,714286
	44-46	42-43-45	0,949749
5	8	22	0,833333
		42-43-44-45-46	0,828571
	22	42-43-44-45-46	0,790698
6	42-43-44-45-46	8-22	0,58

Tabla 6.3.12: Pasadas del Algoritmo de Clustering para el Requerimiento 1

La Figura 6.3.7 describe gráficamente como se fueron agrupando los clusters.

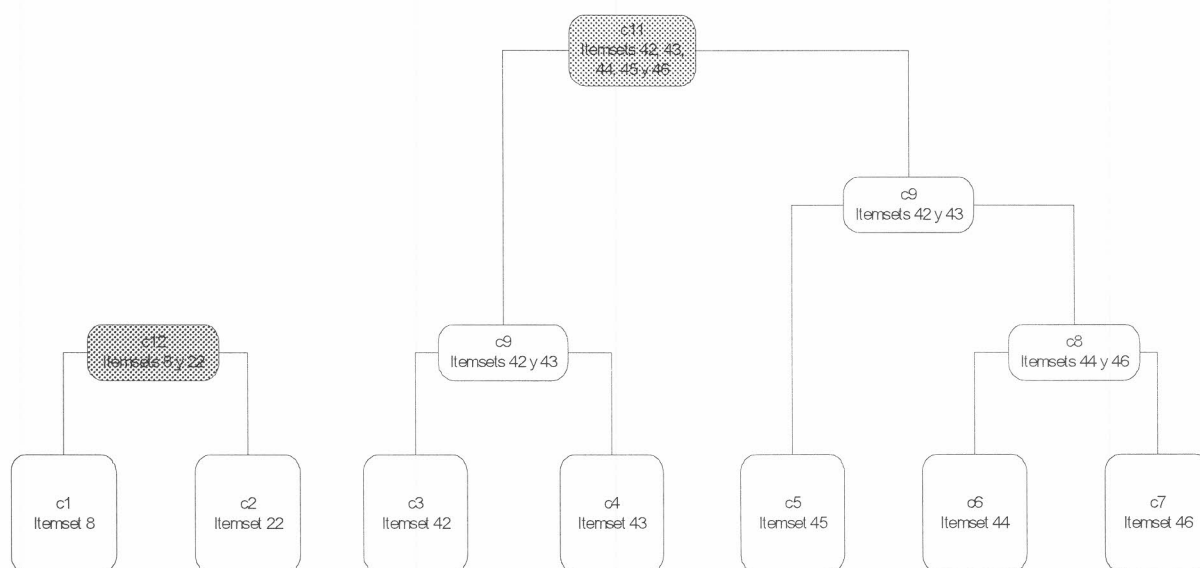


Figura 6.3.7: Clusters del Requerimiento 1

Las Tablas 6.3.13 y 6.3.14 detallan las pasadas del algoritmo de *Clustering* correspondientes a los requerimientos 2 y 3 respectivamente.

Iteración	Itemset 1	Itemset 2	Similaridad
1	53	57	0,62963

Tabla 6.3.13: Pasada del Algoritmo de Clustering para el Requerimiento 2

Iteración	Itemset 1	Itemset 2	Similaridad
1	92	124	0,949296

Tabla 6.3.14: Pasada del Algoritmo de Clustering para el Requerimiento 3

No realizamos las figuras correspondientes a la ejecución del algoritmo de *Clustering* para los requerimientos 2 y 3 porque son triviales, para el requerimiento 2 se obtuvieron 2 clusters, uno para el itemset 53 y otro para el itemset 57 y para el requerimiento 3 se obtuvo un solo cluster formado por los itemsets 92 y 124.

La Tabla 6.3.15 describe los clusters obtenidos con sus respectivos tópicos y el requerimiento al que corresponden. Estos clusters son los que se obtienen antes de aplicar el algoritmo de *Clustering* a todos los clusters del usuario con el fin de agrupar los clusters similares.

Requerimiento	Cluster	Itemsets	Tópicos
1	1	42-43-44-45-46	person, individual, someone, somebody, mortal, human, soul natural object vertebrate, craniate Whale humpback, humpback whale, Megaptera novaeangliae
	2	8-22	Right

			instrumentality, instrumentation rightfield, right right, rightfulness right, right hand right, right wing
2	3	53	Whale
	4	57	message, content, subject matter, substance
3	5	92-124	vertebrate, craniate Whale Dolphin killer whale, killer, orca, grampus, sea wolf, Orcinus orca

Tabla 6.3.15: Tópicos de los clusters

Una vez obtenidos todos los clusters correspondientes a cada requerimiento del usuario, el sistema aplica una vez más el algoritmo de *Clustering* sobre todos ellos con el fin de agrupar los clusters con tópicos similares y reducir así la cantidad de tópicos que le asignara al usuario.

La Tabla 6.3.16 detalla las distintas pasadas del algoritmo de *Clustering* para agrupar todos los clusters de los requerimientos y la Figura 6.3.8 muestra dichas pasadas gráficamente.

Iteracion	Cluster 1	Cluster 2	Similaridad
1	42-43-44-45-46	8-22	0,58
		53	0,419355
		57	0,416667
		92-124	0,973168
	8-22	53	0,25
		57	0,4
		92-124	0,2
	53	57	0,62963
		92-124	0,835616
	2	8-22	53
57			0,4
42-43-44-45-46-92-124			0,857895
53		57	0,62963
		44-46-92-124	0,848819
57		44-46-92-124	0,845114
3	53	57	0,62963
		42-43-44-45-46-92-124	0,835841
	57	44-46-92-124	0,837796
4	57	42-43-44-45-46-53-92-124	0,837796

Tablas 6.3.16: Pasadas del Algoritmo de Clustering para todos los Requerimientos

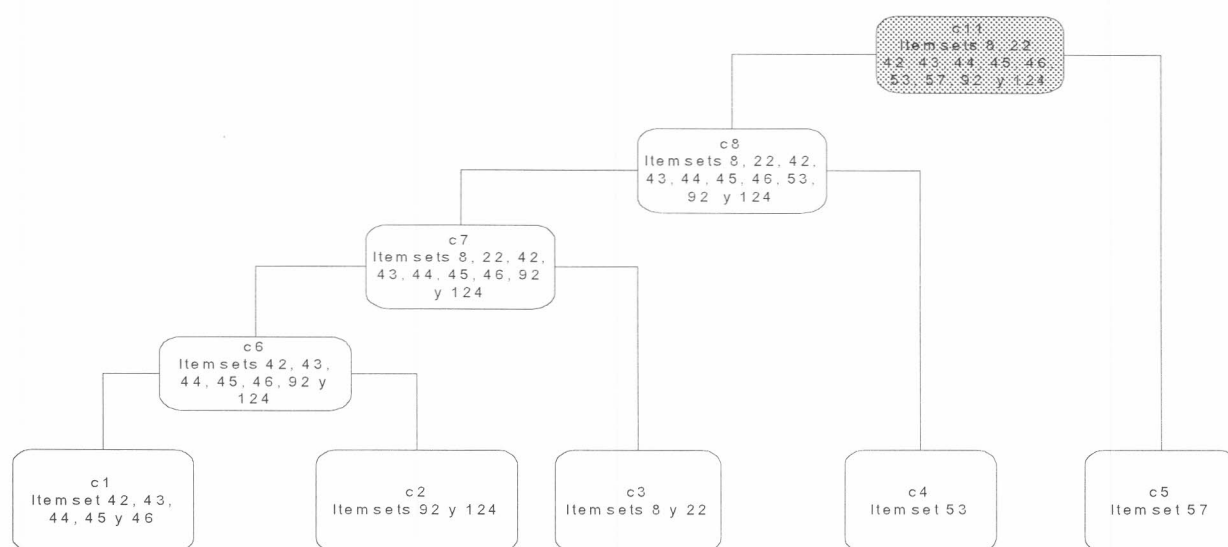


Figura 6.3.8: Clustering de Clusters

La Tabla 6.3.17 describe los tópicos correspondientes al cluster final.

Cluster	Itemsets	Tópicos
11	8	person, individual, someone, somebody, mortal, human, soul
	22	natural object
	42	right
	43	vertebrate, craniate
	44	whale
	45	humpback, humpback whale, Megaptera novaeangliae
	46	dolphin
	53	killer whale, killer, orca, grampus, sea wolf, Orcinus orca
	57	instrumentality, instrumentation
	92	rightfield, right
	124	right, rightfulness
		right
		right, right hand
		message, content, subject matter, substance
		right, right wing

Tabla 6.3.17: Tópicos del Cluster Final

Al observar los tópicos obtenidos en las Tablas 6.3.15 y 6.3.17 notamos que al aplicar el algoritmo de *Clustering* a los clusters del requerimiento sólo se evitan tener tópicos repetidos, ya que el algoritmo obtuvo los mismos tópicos.

6.3.1.8 Categorías

Finalmente las categorías asignadas al usuario del log son los tópicos de mayor bondad de cada uno de los clusters generados para el usuario a partir de los requerimientos del mismo.

A continuación presentamos los tópicos de mayor bondad correspondientes a los clusters obtenidos para cada uno de los requerimientos (Tabla 6.3.18) y el tópico de mayor bondad correspondiente al cluster final (Tabla 6.3.19). Para obtener dichos tópicos *SWMiner* utiliza las definiciones 3.11 [Frecuencia de un Tópico en un Cluster], 3.12 [Frecuencia Relativa de un Tópico en un Cluster] y 3.13 [Bondad de un Tópico en un Cluster].

Requerimiento	Cluster	Itemsets	Tópico de Mayor Bondad
1	1	42-43-44-45-46	Whale
	2	8-22	Instrumentality, instrumentation
2	3	53	Whale
	4	57	message, content, subject matter, substance
3	5	192-124	killer whale, killer, orca, grampus, sea wolf, Orcinus orca

Tabla 6.3.18: Tópicos de Mayor Bondad de los clusters de los Requerimientos

Por lo tanto si *SWMiner* solo aplica el algoritmo de *Clustering* a los requerimientos del usuario los tópicos de la Tabla 6.3.18 serían las categorías correspondientes al usuario que realizó los requerimientos pero como aplica una vez más el algoritmo de *Clustering* sobre todos los clusters obtenidos termina asignándole al usuario, las categorías descritas en la Tabla 6.3.22.

Cluster	Itemsets	Categorías
1	8-22-42-43-44-45	killer whale, killer, orca, grampus
	46-53-57-92-124-192	sea wolf, Orcinus orca

Tabla 6.3.19: Tópico de Mayor Bondad del Cluster Final

Si comparamos las categorías de ambas tablas, vemos que las categorías de la Tabla 6.3.18 son mas específicas que las de la Tabla 6.3.19, sin embargo en la primera aparecen categorías que poco tienen que ver con los mamíferos marinos, como ser las correspondientes a los clusters 2 y 4.

6.3.1.9 SWMiner vs. Google

En esta sección, comparamos los tópicos obtenidos por *SWMiner* para cada requerimiento contra el directorio de categorías de *Google*. La Tabla 6.3.20 describe las categorías asignadas por *Google* a cada requerimiento.

Id	Requerimiento	Categorías
1	http://dkd.net/whales/	Science > Biology > Flora and Fauna > Animalia > Chordata > Mammalia > Cetacea
2	http://www.whale-web.com/	Science > Biology > Flora and Fauna > Animalia > Chordata > Mammalia > Cetacea
3	http://www.crru.org.uk/	Society > Issues > Animal Welfare > Specific Animals > Whales

Tabla 6.3.20: Categorías de los requerimientos en el Directorio *Google*

La Tabla 6.3.21 muestra el subconjunto de categorías asignadas por *Google* para los 3 requerimientos, que se consideran en las comparaciones con *SWMiner*. Cabe recordar, que

decidimos tomar las últimas 4 categorías de cada jerarquía definida en el Directorio *Google* para cada requerimiento.

Categorías
Animalia, Chordata, Mammalia, Cetacea, Issues, Animal Welfare, Specific Animals, Whales

Tabla 6.3.21: Categorías de Google

Para completar la Tabla 6.3.23 utilizamos la Definición 3.21 [*Hits*, *Misses* y *Mistakes* para Identificación de Tópicos], la Tabla 6.3.22 detalla qué categorías consideramos *Hits*, *Mistakes* y *Misses* al comparar las categorías de la Tabla 6.3.21 con las obtenidas por *SWMiner* (Tabla 6.3.19).

Categorías	
Hits	killer whale, orca, grampus, sea wolf, Orcinus orca
Misses	Animalia, Chordata, Mammalia, Issues, Animal Welfare, Specific Animals
Mistakes	Killer

Tabla 6.3.22: Hits, Misses y Mistakes

Hits	Misses	Mistakes	Precision	Recall
5	6	1	0.83333333	0.41666667

Tabla 6.3.23: Precision y Recall

Precision - Valor Promedio: 0.83333333

Recall - Valor Promedio: 0.41666667

De estos valores podemos concluir que el 83 % de los tópicos obtenidos coinciden o están relacionados con las categorías obtenidas del directorio de *Google* y que de todas las categorías de *Google* el algoritmo obtuvo el 41%.

6.3.2 SWMiner con Normalización Morfológica y sin Clustering

En esta sección presentamos el resultado obtenido al ejecutar *SWMiner con Normalización Morfológica* de las palabras correspondientes a los requerimientos y sus vecindades y sin aplicar *Clustering*. Es decir con los itemsets frecuentes de cada uno de los requerimientos y sus vecindades accedemos a *WordNet* para obtener los tópicos que generalizan los ítems que los componen y luego asignamos dichos tópicos a los requerimientos. La Tabla 6.3.24 describe los tópicos asignados a cada itemset.

Req	Id Itemset	Items	Tópicos
1	8	RIGHT	RIGHT
			RIGHTFIELD
			RIGHTFULNESS
			RIGHT HAND
			RIGHT WING
	22	BOAT	INSTRUMENTALITY
			INSTRUMENTATION
	42	HUMPBACK	NATURAL OBJECT
		QUEENSLAND	
		WHALE	

		WORLD	
	43	DOLPHIN	VERTEBRATE, CRANIATE
		QUEENSLAND	
		WHALE	
		WORLD	
	44	DOLPHIN	WHALE
		HUMPBAC	
		QUEENSLAND	
		WHALE	
	45	CETACEAN	WHALE
		HUMPBAC	HUMPBAC
		QUEENSLAND	HUMPBAC WHALE
		WHALE	MEGAPTERA NOVAEANGLIAE
	46	HUMPBAC	PERSON
		LOGO	INDIVIDUAL
		QUEENSLAND	SOMEBODY
		WHALE	SOMEONE
			MORTAL
			HUMAN
			SOUL
2	53	WHALE	WHALE
		CETACEANS	
	57	WHALE	MESSAGE
		QUESTION	CONTENT
		CORRECTION	SUBJECT MATTER
			SUBSTANCE
3	92	DOLPHIN	VERTEBRATE
		PORPOISE	CRANIATE
		CETACEAN	
	124	WHALE	WHALE
		DOLPHIN	DOLPHIN
		CETACEAN	KILLER WHALE
		RESCUE	KILLER
		KILLER	ORCA
		UNIT	GRAMPUS
			SEA WOLF
			ORCINUS ORCA

Tabla 6.3.24: Tópicos de los Itemsets Frecuentes de los Requerimientos

Finalmente el sistema le asigna al usuario los siguientes tópicos como categorías.

Categorías
instrumentality, instrumentation
message, content, subject matter, substance
natural object
person, individual, someone, somebody, mortal, human, soul
Right
vertebrate, craniate
Whale

Tabla 6.3.25: Categorías asignadas al Usuario

Si comparamos las categorías que obtuvimos con la ejecución completa del *SWMiner* (Tabla 6.3.19) y las de la Tabla 6.3.25 notamos que se incrementa la cantidad de categorías pero sólo *vertebrate*, *craniate* y *whale* son tópicos de interés del usuario por lo tanto podemos concluir que al ejecutar *SWMiner Completo* obtenemos mejores categorías para el usuario.

6.3.2.1 SWMiner vs. Google

Supongamos que cada uno de los itemsets obtenidos es un cluster y comparemos las categorías obtenidas para cada una de los requerimientos con las categorías obtenidas en el Directorio *Google* (Tabla 6.3.21).

La Tabla 6.3.26 detalla las categorías que consideramos que corresponden a *Hits*, *Mistakes* y *Misses* al comparar las Tablas 6.3.19 y 6.3.25.

	Categorías
Hits	Whale
Misses	Animalia, Chordata, Mammalia, Issues, Animal Welfare, Specific Animals
Mistakes	instrumentality, instrumentation, , message, content, subject matter, substance, natural object, person, individual, someone, somebody, mortal, human, soul, Right, vertebrate, craniate

Tabla 6.3.26: Hits, Misses y Mistakes

Hits	Misses	Mistakes	Precision	Recall
1	7	17	0.05555556	0.125

Tabla 6.3.27: Precision y Recall

Precision - Valor Promedio: 0.05555556

Recall - Valor Promedio: 0.125

De estos valores podemos concluir que el 5 % de los tópicos obtenidos coinciden o están relacionados con las categorías obtenidas del directorio de *Google* y que de todas las categorías de *Google* el algoritmo obtuvo al 12%. Por lo tanto los valores de Precision y Recall obtenidos al comparar la ejecución de *SWMiner Completo* (Tabla 6.3.19) son mejores que los obtenidos al ejecutar *SWMiner sin Clustering*.

6.3.3 SWMiner sin Normalización Morfológica y con Clustering

En esta sección detallamos los resultados obtenidos al ejecutar *SWMiner* sin normalización morfológica de las palabras que representan a cada requerimiento y su vecindad y

aplicando el algoritmo de clustering y comparamos los resultados obtenidos con la ejecución completa del *SWMiner*.

Lo primero que podemos destacar es que al no aplicar normalización lexicográfica se incrementó bastante el tiempo de ejecución del *Preprocesador*, ya que en cada una de las subetapas de normalización disminuía la cantidad de palabras a analizar y al no pasar por estas se aplican las subetapas siguientes (verificación de la existencia de las palabras en WordNet y en StopWords, eliminación de caracteres especiales, etc.) sobre todas las palabras extraídas de los documentos html.

En segundo lugar, al no hacer normalización morfológica de las palabras *SWMiner* obtiene en la mayoría de los casos la cantidad de palabras deseadas por el usuario (parámetro del sistema), al aplicar la normalización no siempre se obtiene esta cantidad de palabras porque se descartan aquellas que no son sustantivos. Debido a esto fue necesario modificar el soporte del algoritmo de *Association Rules* ya que al ejecutar el *Categorizador* con soporte 0.30 la cantidad de itemsets obtenida se incrementó notablemente. El nuevo valor especificado en este parámetro es 0.50.

La Tabla 6.3.28 describe los clusters obtenidos con los itemsets que los componen, los items que componen a estos últimos y el requerimiento al que corresponden. Estos clusters son los que se obtienen antes de aplicar el algoritmo de *Clustering* a todos los clusters del usuario con el fin de agrupar los clusters similares.

Requerimiento	Cluster	Itemsets	Items	Tópicos
1	1	9	WHALE	
			WATCHING	
	2	11	SOUNDS	BODY OF WATER, WATER
			QUEENSLAND	
	3	13	RIGHTS	MORALITY
			RESERVED	
	4	14	WHALE	
			QUEENSLAND	
			LOGO	
2	5	17	WHALE	VERTEBRATE, CRANIATE
			DOLPHIN	
3	6	23	WHALES	WHALE
			CETACEAN	
		34	WILDLIFE	
			CETACEAN	
			RESCUE	
			UNIT	

Tabla 6.3.28: Clusters SWMiner sin Normalización y con Clustering

La tabla 6.3.29 describe los clusters obtenidos al aplicar el algoritmo de *Clusters* a todos los clusters de los requerimientos.

Cluster	Itemsets	Items
1	11	SOUNDS
		QUEENSLAND

2	13	RIGHTS
		RESERVED
3	9	WHALE
		WATCHING
	14	WHALE
		QUEENSLAND
		LOGO
	17	WHALE
		DOLPHIN
	23	WHALES
		CETACEAN
	34	WILDLIFE
		CETACEAN
		RESCUE
		UNIT

Tabla 6.3.29: Clusters Finales de SWMiner sin Normalización y con Clustering

La Tabla 6.3.30 describe los tópicos correspondientes a cada uno de los clusters y la Tabla 6.3.31 describe los tópicos de mayor bondad de cada cluster que son las categorías que se le asignan al usuario.

Cluster	Tópico
1	BODY OF WATER, WATER
2	MORALITY
3	VERTEBRATE, CRANIATE WHALE

Tabla 6.3.30: Tópicos de los Clusters

Categorías
Body of water, Water, Morality, Whale

Tabla 6.3.31: Categorías asignadas al Usuario

6.3.3.1 SWMiner vs. Google

La Tabla 6.3.33 describe los valores de precision y recall obtenidos al comparar las categorías asignadas al usuario por *SWMiner* (Tabla 6.3.31) con las categorías obtenidas del directorio de *Google* (Tabla 6.3.21).

Categorías	
Hits	Whale
Misses	Animalia, Chordata, Mammalia, Issues, Animal Welfare, Specific Animals
Mistakes	Body of water, Water, Morality

Tabla 6.3.32: Hits, Misses y Mistakes

Hits	Misses	Mistakes	Precision	Recall
1	7	3	0.25	0.125

Tabla 6.3.33: Precision y Recall

Precision - Valor Promedio: 0.25

Recall - Valor Promedio: 0.125

De estos valores podemos concluir que el 25% de los tópicos obtenidos coinciden o están relacionados con las categorías obtenidas del directorio de *Google* y que de todas las categorías de *Google* el algoritmo obtuvo el 12%.

6.3.4 SWMiner sin Normalización Morfológica y sin Clustering

En la Tabla 6.3.34 presentamos los resultados obtenidos al ejecutar *SWMiner* sin aplicar normalización morfológica ni clustering.

Requerimiento	Itemsets	Items	Tópicos
1	9	WHALE	
		WATCHING	
	11	SOUNDS	BODY OF WATER, WATER
		QUEENSLAND	
	13	RIGHTS	MORALITY
		RESERVED	
	14	WHALE	
		QUEENSLAND	
		LOGO	
2	17	WHALE	VERTEBRATE, CRANIATE
		DOLPHIN	
3	23	WHALES	WHALE
		CETACEAN	
	34	WILDLIFE	
		CETACEAN	
		RESCUE	
		UNIT	

Tabla 6.3.34: SWMiner sin Normalización y sin Clustering

La Tabla 6.3.35 describe las categorías que se le asignan la usuario

Categorías
Body of water, Water, Morality, Vertebrate, Craniate, Whale

Tabla 6.3.35: Categorías asignadas al usuario

6.3.4.1 SWMiner vs. Google

Supongamos que cada uno de los itemsets obtenidos es un cluster y comparemos las categorías asignadas al usuario por *SWMiner* (Tabla 6.3.35) con las categorías obtenidas en el Directorio *Google* (Tabla 6.3.21).

Categorías	
Hits	Whale
Misses	Animalia, Chordata, Mammalia, Issues, Animal Welfare, Specific Animals
Mistakes	Body of water, Water, Morality, Vertebrate, Craniate

Tabla 6.3.36: Hits, Misses y Mistakes

Hits	Misses	Mistakes	Precision	Recall
1	7	5	0.16666667	0.125

Tabla 6.3.37: Precision y Recall

Precision - Valor Promedio: 0.16666667

Recall - Valor Promedio: 0.125

De estos valores podemos concluir que el 16% de los tópicos obtenidos coinciden o están relacionados con las categorías obtenidas del directorio de *Google* y que de todas las categorías de *Google* el algoritmo obtuvo el 12%.

6.3.5 SWMiner vs. TOPIC

En esta sección comparamos los resultados obtenidos con la ejecución completa de *SWMiner* con los tópicos obtenidos por *TOPIC* [W36]. En la Tabla 6.3.38 se describen los tópicos obtenidos al ejecutar *TOPIC* con *Lycos*, para cada uno de los requerimientos del log de mamíferos marinos.

Requerimiento	Rank	Tópico
1 (http://dkd.net/whales/)	9.11645e-05	Hervey Bay
	1.07326e-05	Queensland
	5.40152e-06	Whales
	8.93443e-07	Australia
2 (http://www.whale-web.com/)		No se encontraron tópicos
3 (http://www.crru.org.uk/)	0.196078	http://www.cs.toronto.edu/db/topic/topic.cgi?url=www.crru.org.uk/&engine=Lycos&ino=300&terms=www.crru.org.uk
	7.52887e-05	Cetacean
	1.19351e-06	Rescue
	6.07677e-07	Research

Tabla 6.3.38: Tópicos de los Requerimientos obtenidos con TOPIC

Como no obtuvimos ningún tópico correspondiente al requerimiento 2, ejecutamos *TOPIC* indicándole que devuelva las páginas de mayor reputación sobre el tópico *whale* (ver Tabla 6.3.19).

Rank	Página
0.00259041	Go to WhaleNet

0,00034791	The World-Wide Web Virtual Library: Whale-Watching Web
0,000268348	WDCS homepage - The Whale and Dolphin Conservation Society - ...
6,60595	WhaleTimes SeaBed ~ Whales, dolphins, sharks, penguins and more
4,22781	Welcome to the Whale Museum.
3,81677	Dolphins, Whales, & Manatees - Education Made Fun - The WhaleClub
3,58189	Welcome to WhaleNet - Boston, Massachusetts
2,05518	Whale Songs
1,90838	Charlotte, The Vermont Whale
0,557836	www.whale-web.com/

Tabla 6.3.39: Tópicos de los requerimientos obtenidos con TOPIC

En la tabla 6.3.39 podemos ver que la página correspondiente al requerimiento 2 (ver fila sombreada) posee una buena reputación sobre el tópico *whale* que fue el obtenido por *SWMiner*.

Al comparar los tópicos obtenidos con la ejecución de *SWMiner completa* (ver Tabla 6.3.19) con los de la Tabla 6.3.38 notamos que los tópicos *killer whale*, *orca*, *grampus*, *sea wolf* y *Orcinus orca* están relacionados a los tópicos *Whales* y *Cetacean* obtenidos por *TOPIC* y que *SWMiner* no obtuvo los tópicos *Hervey Bay*, *Quensland*, *Australia*, *Rescue*, *Research* y la url. *Hervey Bay*, *Rescue* y *Research* no fueron obtenidos porque no cumplen el soporte mínimo utilizado al obtener los itemsets frecuentes; *Queensland* y *Australia* fueron descartados por *SWMiner* porque no tiene en cuenta los nombres propios al realizar el preprocesamiento de los documentos web. El tópico correspondiente a la url no fue identificado por *SWMiner* porque sólo reconoce sustantivos como tópicos.

6.3.6 SWMiner vs. Categorías asignadas manualmente

En esta sección detallamos los resultados obtenidos al comparar las categorías asignadas al usuario por la ejecución de *SWMiner completo* (Tabla 6.3.19) contra las categorías asignadas en forma manual al observar el contenido de los documentos *Html* correspondientes a los requerimientos accedidos por el usuario (Tabla 6.3.41).

La Tabla 6.3.40 describe las categorías asignadas manualmente por requerimiento y la Tabla 6.3.41 muestra el conjunto total de categorías.

Requerimiento	Categorías
1	Whales, Hervey Bay, Queensland, Australia, Cetacean, Humpbacks, Orcas, Minkes, Dolphins
2	Cetancean, Whale watch, Whales
3	Cetacean, Whales, Whale and Rescue Work, Dolphins

Tabla 6.3.40: Categorías asignadas manualmente a cada requerimiento

Categorías
Whales, Hervey Bay, Queensland, Australia, Cetacean, Humpbacks, orcas, Minkes, Dolphins, Whale watch, Whale and Dolphins Rescue Work

Tabla 6.3.41: Conjunto total de categorías asignadas a mano

Categorías	
Hits	Killer whale, orca, grampus, sea wolf, Orcinus orca
Misses	Hervey Bay, Queensland, Australia, Humpbacks, Minkes, Dolphins, Whale Watch, Whale and Dolphins Rescue Work
Mistakes	Killer

Tabla 6.3.42: Hits, Misses y Mistakes

Hits	Misses	Mistakes	Precision	Recall
5	8	1	0.83333333	0.38461538

Tabla 6.3.43: Precision y Recall

Precision - Valor Promedio: 0.83333333

Recall - Valor Promedio: 0.38461538

De estos valores podemos concluir que el 83% de los tópicos obtenidos coinciden o están relacionados con las categorías identificadas a mano y que de todas las categorías definidas manualmente el algoritmo obtuvo el 38%.

6.4 Experimentos con Log de Animales Salvajes

En esta sección se muestran los resultados obtenidos con los distintos experimentos realizados con un log de Animales Salvajes. Para ver el detalle completo de la experimentación realizada remitirse al Anexo de Experimentación.

6.4.1 SWMiner con Normalización Morfológica y con Clustering

6.4.1.1 Requerimientos del Log

En este experimento se analizó un log compuesto por 3 requerimientos de un mismo usuario, llegando a analizar en total un conjunto de 100 páginas incluyendo las vecindades involucradas.

Id	Requerimiento
1	http://www.nature-wildlife.com/girtxt.htm
2	http://www.cathouse-fcc.org/catsinfo.html
3	http://www.geocities.com/RainForest/4076/index1.html

Tabla 6.4.1: Requerimientos del Log

6.4.1.2 Evaluación del algoritmo de Identificación de Idiomas

Para completar la Tabla 6.4.2 utilizamos la Definición 3.17 [*Hits, Misses y Mistakes para Identificación de Idiomas*]

Requerimiento	Hits	Misses	Mistakes	Precision	Recall
http://www.nature-wildlife.com/girtxt.htm	11	4	2	0,84615385	0,73333333
http://www.cathouse-fcc.org/catsinfo.html	24	3	0	1	0,84615385
http://www.geocities.com/RainForest/4076/index1.html	23	0	0	1	1

Tabla 6.4.2: Precision y Recall para Identificación de Idiomas

Precision - Valor Promedio 0,94871795

Recall - Valor Promedio 0,87407407

Como puede observarse, el algoritmo de identificación de idiomas tiene en este caso, una precisión del 94%.

6.4.1.3 Evaluación del algoritmo de Part Of Speech

Para completar la Tabla 6.4.3 se utilizó la definición 3.19 [*Hits, Misses y Mistakes para Part Of Speech*]

Link	Hits	Misses	Mistakes	Precision	Recall
2	9	16	11	0,45	0,36
46	13	8	9	0,59090909	0,61904762
33	19	16	8	0,7037037	0,54285714
40	19	18	8	0,7037037	0,51351351
24	21	24	8	0,72413793	0,46666667
41	13	36	4	0,76470588	0,26530612
37	20	13	6	0,76923077	0,60606061
29	24	21	7	0,77419355	0,53333333
44	31	16	9	0,775	0,65957447
35	21	13	6	0,77777778	0,61764706
43	15	6	4	0,78947368	0,71428571
27	24	30	6	0,8	0,44444444
45	32	16	8	0,8	0,66666667
47	32	16	8	0,8	0,66666667
32	29	11	5	0,85294118	0,725
48	19	29	2	0,9047619	0,39583333
28	30	15	3	0,90909091	0,66666667
31	37	8	3	0,925	0,82222222
36	30	3	1	0,96774194	0,90909091
30	32	2	0	1	0,94117647
39	15	69	2	0,88235294	0,17857143
25	26	21	4	0,86666667	0,55319149
26	26	34	3	0,89655172	0,43333333

Tabla 6.4.3: Precision y Recall para Part Of Speech

Precision - Valor Promedio 0,80121493

Recall - Valor Promedio 0,57831113

De estos valores se desprende que el algoritmo de *Part Of Speech* ha detectado correctamente el 57 % de los sustantivos existentes y tiene una precisión promedio del 80%; es decir el 80% de los sustantivos obtenidos por el algoritmo son correctos.

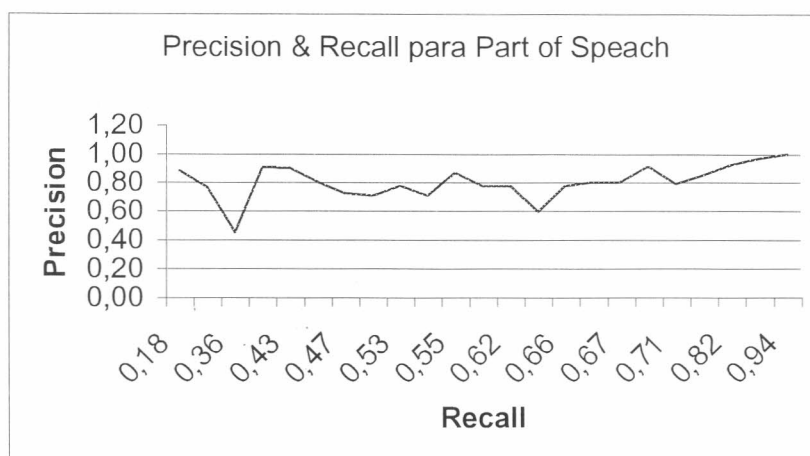


Figura 6.4.1: Precision y Recall para Part Of Speech

6.4.1.4 Evaluación del algoritmo de Lemmatiser

Para completar la Tabla 6.4.4 utilizamos la definición 3.18 [*Hits*, *Misses* y *Mistakes* para Lemmatiser]

Link	Hits	Misses	Mistakes	Precision	Recall
2	44	86	0	1	0,33846154
24	94	36	1	0,98947368	0,72307692
27	19	9	0	1	0,67857143
28	68	26	1	0,98550725	0,72340426
29	16	12	0	1	0,57142857
31	25	13	0	1	0,65789474
32	59	35	4	0,93650794	0,62765957
33	34	20	4	0,89473684	0,62962963
35	108	50	1	0,99082569	0,6835443
36	180	211	4	0,97826087	0,46035806
37	204	121	9	0,95774648	0,62769231
40	146	49	5	0,96688742	0,74871795
41	21	22	0	1	0,48837209
43	29	8	2	0,93548387	0,78378378
44	37	31	0	1	0,54411765
45	44	30	0	1	0,59459459
46	17	9	0	1	0,65384615
47	38	30	0	1	0,55882353
48	39	18	3	0,92857143	0,68421053
30	3	1	0	1	0,75
39	15	6	0	1	0,71428571
25	18	10	0	1	0,64285714
26	20	4	0	1	0,83333333

Tabla 6.4.4: Precision y Recall para Lemmatiser

Precision - Valor Promedio 0,98104354

Recall - Valor Promedio 0,6399419

El 98% de los lemas obtenidos por el algoritmo de Lemmatiser son correctos, y sobre el total de palabras analizadas el algoritmo obtuvo los lemas del 63%.

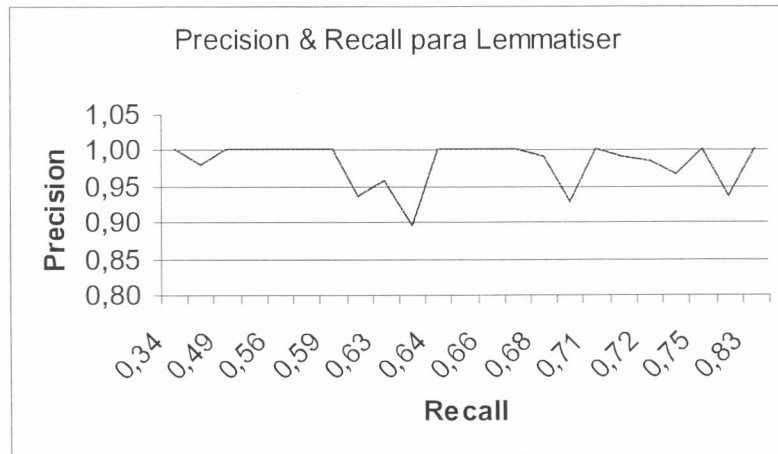


Figura 6.4.2: Precision y Recall para Lemmatiser

6.4.1.5 Categorías

La Tabla 6.4.5 muestra los tópicos identificados como las categorías del usuario del log.

Categorías
Copy
class, year
natural object
feline, felid
internet, cyberspace
country, rural area
Sign
heart, spirit
giraffe, camelopard, Giraffa camelopardalis

Tabla 6.4.5: Categorías asignadas al usuario

6.4.1.6 SWMiner vs. Google

Para tener una medida de la performance del algoritmo de identificación de tópicos, utilizamos la Definición 3.21 [Hits, Misses y Mistakes para Identificación de Tópicos], comparando las categorías asignadas al usuario por *SWMiner* (Tabla 6.4.5) contra la categorización de la respectiva página en el directorio de categorías del motor de búsqueda *Google*.

En la Tabla 6.4.6 se muestra el subconjunto de categorías asignadas por *Google* a cada uno de los requerimientos, que se consideran en las comparaciones con *SWMiner*. Cabe

recordar, que decidimos tomar las últimas 4 categorías de cada jerarquía definida en el Directorio *Google* para cada requerimiento.

Id	Requerimiento	Categoría
1	http://www.nature-wildlife.com/girtxt.htm	Kids and Teens > School Time > Science > Living Things > Animals > Mammals > Giraffes
2	http://www.cathouse-fcc.org/catsinfo.html	Science > Biology > Flora and Fauna > Animalia > Chordata > Mammalia > Carnivora > Felidae
3	http://www.geocities.com/RainForest/4076/index1.html	Society > Religion and Spirituality > Shamanism

Tabla 6.4.6: Categoría de cada requerimiento en el directorio de Google

La tabla 6.4.7 muestra el subconjunto las categorías finales asignadas por *Google* a los 3 requerimientos.

Categorías
Living Things , Animals, Mammals, Giraffes, Chordata , Mammalia, Carnivora, Felidae, Society, Religion and Spirituality, Shamanism

Tabla 6.4.7: Categorías de Google

Para completar la Tabla 6.4.9 utilizamos la Definición 3.21 [*Hits, Misses y Mistakes para Identificación de Tópicos*]. La Tabla 6.4.8 describe las categorías que consideramos que caen en *Hits, Misses y Mistakes*.

Categorías	
Hits	feline, felid, spirit, giraffe, camelopard, Giraffa camelopardalis
Misses	Living Things , Animals, Mammals, Chordata , Mammalia, Carnivora, , Society, Shamanism
Mistakes	copy, class, year, natural object, internet, cyberspace, country, rural area, sign, heart

Tabla 6.4.8: Hits, Misses y Mistakes

Hits	Misses	Mistakes	Precision	Recall
6	8	10	0.375	0.42857143

Tabla 6.4.9: Precision y Recall para Identificación de Tópicos

Precision - Valor Promedio 0,375

Recall - Valor Promedio 0,42857143

De estos valores podemos concluir que el 37% de los tópicos obtenidos coinciden o están relacionados con las categorías asignadas por *Google* y que de todas las categorías de *Google* el algoritmo detectó el 42%.

6.4.2 SWMiner con Normalización Morfológica y sin Clustering

En esta sección presentamos el resultado obtenido al ejecutar *SWMiner* sin aplicar la etapa de *Clustering*. Es decir, a partir de los itemsets frecuentes obtenidos del conjunto de

requerimientos y sus vecindades accedemos a *WordNet* para obtener los tópicos que generalizan los ítems que los componen y luego asignamos dichos tópicos a los usuarios.

El sistema asigna como categorías al usuario los tópicos que se muestran en la Tabla 6.4.10.

Categorías
copy
giraffe, camelopard, Giraffa camelopardalis
person, individual, someone, somebody, mortal, human, soul
representation
cougar, puma, catamount, mountain lion, painter, panther, Felis concolor
bobcat, bay lynx, Lynx rufus
internet, cyberspace
big cat, cat
natural object
feline, felid
instrumentality, instrumentation
country, rural area
sign
heart, spirit
class, year

Tabla 6.4.10: Categorías asignadas al usuario

Podemos observar que en este caso, se obtienen todas las categorías que se obtienen con la ejecución de *SWMiner Completo* junto con algunas categorías más.

6.4.2.1 SWMiner vs. Google

Supongamos que cada uno de los itemsets obtenidos es un cluster y comparemos las categorías asignadas al usuario por *SWMiner* (Tabla 6.4.10) con las categorías asignadas por *Google* (Tabla 6.4.7). La Tabla 6.4.11 describe qué categorías consideramos *Hits*, *Misses* y *Mistakes*.

	Categorías
Hits	giraffe, camelopard, Giraffa camelopardalis, cougar, puma, catamount, mountain lion, painter, panther, Felis concolor, bobcat, bay lynx, Lynx rufus, big cat, cat, feline, felid, spirit, soul
Misses	Living Things , Animals, Mammals, Chordata , Mammalia, Society, Shamanism
Mistakes	copy, person, individual, someone, somebody, mortal, human, representation, internet, cyberspace, natural object, instrumentality, instrumentation, country, rural area, sign, heart, class, year

Tabla 6.4.11: Hits, Misses y Mistakes

Hits	Misses	Mistakes	Precision	Recall
19	7	19	0.5	0.73076923

Tabla 6.4.12: Precision y Recall para Identificación de Tópicos sin Clustering

Precision - Valor Promedio 0.50

Recall - Valor Promedio 0,73076923

El 50% de los tópicos obtenidos están relacionados con las categorías asignadas por *Google* y de todas las categorías de *Google* el algoritmo detectó el 73%.

6.4.3 SWMiner sin Normalización Morfológica y con Clustering

A continuación detallamos los resultados obtenidos al ejecutar *SWMiner sin Normalización Morfológica* de las palabras que representan a cada requerimiento y su vecindad, y comparamos los resultados obtenidos con la ejecución completa de *SWMiner*.

Categorías
animal, animate being, beast, brute, creature, fauna

Tabla 6.4.13: Categorías asignadas al usuario

6.4.3.1 SWMiner vs. Google

Aplicamos la Definición 3.21 [*Hits, Misses y Mistakes para Identificación de Tópicos*] para analizar los resultados obtenidos con *SWMiner*, comparando las categorías identificadas por el sistema sin aplicar *Normalización Morfológica* (Tabla 6.4.13) contra la categorización que *Google* asigna a cada requerimiento (Tabla 6.4.7). La Tabla 6.4.14 detalla qué categorías consideramos *Hits, Misses y Mistakes*.

	Categorías
Hits	animal, animate being, beast, creature, fauna
Misses	Living Things, Mammals, Giraffes, Chordata, Mammalia, Carnivora, Felidae, Society, Religion and Spirituality, Shamanism
Mistakes	brute

Tabla 6.4.14: Hits, Misses y Mistakes

Hits	Misses	Mistakes	Precision	Recall
5	9	1	0.83333333	0.33333333

Tabla 6.4.15: Precision y Recall para Identificación de Tópicos sin Clustering

Precision - Valor Promedio 0,83333333

Recall - Valor Promedio 0,33333333

Podemos concluir que el 83% de los tópicos obtenidos coinciden o están relacionados con las categorías asignadas por *Google* y que de todas las categorías de *Google* el algoritmo detectó el 33%. Estos valores son mucho menores que los obtenidos anteriormente.

6.4.4 SWMiner sin Normalización Morfológica y sin Clustering

A continuación detallamos los resultados obtenidos al ejecutar *SWMiner sin Normalización Morfológica* de las palabras ni *Clustering*, y comparamos los resultados obtenidos con la ejecución completa de *SWMiner*.

Categorías
copy, representation, mammal, populace, public, world, animal, animate being, beast, brute, creature, fauna, internet, cyberspace, coinage, mintage, specie, metal money, feline, felid, wildcat

Tabla 6.4.16: Categorías asignadas al usuario

6.4.4.1 SWMiner vs. Google

Aplicamos la Definición 3.21 [*Hits, Misses y Mistakes para Identificación de Tópicos*] para analizar los resultados obtenidos con *SWMiner*, comparando las categorías identificadas por el sistema sin aplicar las etapas de *Normalización Morfológica* y *Clustering* (ver Tabla 6.4.16) contra la categorización que *Google* asigna a cada requerimiento (ver Tabla 6.4.7). La Tabla 6.4.17 detalla qué categorías consideramos *Hits, Misses y Mistakes*.

Categorías	
Hits	Mammal, animal, animate being, beast, creature, fauna, specie, feline, felid, wildcat
Misses	Living Things, Giraffes, Chordata, Society, Religion and Spirituality, Shamanism
Mistakes	copy, representation, populace, public, world, brute, internet, cyberspace, coinage, mintage, metal money

Tabla 6.4.17: Hits, Misses y Mistakes

Hits	Misses	Mistakes	Precision	Recall
10	6	11	0.47619047	0.625

Tabla 6.4.18: Precision y Recall

Precision - Valor Promedio 0.47619047

Recall - Valor Promedio 0.625

De los resultados obtenidos podemos concluir que el 47% de los tópicos están relacionados con las categorías asignadas por *Google* y de todas las categorías de *Google* el *SWMiner* detectó el 62%.

6.4.5 SWMiner vs. TOPIC

En la Tabla 6.4.19 se listan los tópicos detectados por *TOPIC* utilizando el motor de búsqueda *Lycos*, para cada uno de los requerimientos del log en análisis.

Requerimiento	Rank	Tópico
1		No se encontraron tópicos.
2		No se encontraron tópicos.
3	0.454996	index1.html Shamanism FAQs A collection of FAQs
	0.0834098	FAQs A collection of FAQs regarding shamanism
	0.0770749	animal helpers with graphics and text
	0.0748728	healed by their spirits
	0.0734711	Learn about 200 animal helpers
	0.000529901	www.faqs.org
	3.58407e-05	odp
	2.09301e-05	RainForest
	9.22356e-06	www.geocities.com
	4.0855e-07	Search

Tabla 6.4.19: TOPIC(Lycos)

TOPIC no encontró ningún tópico asociado a los requerimientos 1 y 2. Con respecto al requerimiento 3, se puede observar que entre los tópicos obtenidos con la ejecución de *SWMiner Completo* (ver Tabla 6.4.5) el tópico *heart, spirit* está relacionado con alguno de los tópicos devueltos por *TOPIC* como por ejemplo *healed by their spirits*. El caso del tópico correspondiente a la url no fue obtenido por *SWMiner* por que el sistema sólo obtiene sustantivos como tópicos.

6.4.6 SWMiner vs. Categorías asignadas manualmente

En esta sección detallamos los resultados obtenidos al comparar las categorías asignadas al usuario con la ejecución de *SWMiner Completo* (ver Tabla 6.4.5) contra las categorías asignadas en forma manual al observar el contenido de los documentos *Html* correspondientes a los requerimientos accedidos por el usuario (Tabla 6.4.21).

La Tabla 6.4.20 describe las categorías asignadas manualmente por requerimiento y la Tabla 6.4.21 contiene la unión de las categorías de todos los requerimientos accedidos por el usuario.

Requerimiento	Categorías
1	Giraffes, Giraffe Behavior, Giraffe subspecies, Giraffe features
2	Feline species, Wild cat species
3	Working with animal spirits, Jaguars, Orcas, Black Jaguars, Crow Rooks, Cougars, Mountain Lions, Pumas, Animals, Dinosaurs, Mythical animals, Helping injured orphaned and sick wildlife, Books about animals

Tabla 6.4.20: Categorías asignadas a mano

Categorías
Giraffes, Giraffe Behavior, Giraffe subspecies, Giraffe features, Feline species, Wild cat species, Working with animal spirits, Jaguars, Orcas, Black Jaguars, Crow Rooks, Cougars, Mountain Lions, Pumas, Animals, Dinosaurs, Mythical animals, Helping injured orphaned and sick wildlife, Books about animals

Tabla 6.4.21: Unión de Categorías asignadas a mano

	Categorías
Hits	Feline, felid, giraffe, camelopard, giraffa camelopardis
Misses	Working with animals spirits, orcas, crown rooks, dinosaurs, mythical animals, helping injured orphaned and sick wildlife, books about animals
Mistakes	Copy, class, year, natural object, internet, cyberspace, country, rural area, sing, heart

Tabla 6.4.22: Hits, Misses y Mistakes

Hits	Misses	Mistakes	Precision	Recall
5	7	10	0.33333333	0.41666666

Tabla 6.4.23: Precision y Recall

Precision - Valor Promedio: 0.33333333

Recall - Valor Promedio: 0.41666666

De estos valores podemos concluir que el 33% de los tópicos obtenidos coinciden o están relacionados con las categorías identificadas a mano y que de todas las categorías definidas manualmente el algoritmo obtuvo el 41%.

6.5 Experimentos con Log General

En los siguientes experimentos evaluamos los resultados obtenidos al analizar un log cuyos requerimientos no están relacionados ni fueron acotados a un tema específico como en los experimentos anteriores. Las urls contenidas en el log fueron extraídas de la página *Reputation of Know Authorities de TOPIC* [W36] con el fin de comparar nuestros resultados con los de la herramienta TOPIC.

6.5.1 SWMiner con Normalización Morfológica y con Clustering

6.5.1.1 Requerimientos del Log

En este experimento se analizó un log con los 3 requerimientos que se describen en la Tabla 6.5.1.

Id	Requerimiento
1	http://java.sun.com
2	http://www.eff.org
3	http://www.dw-institute.com

Tabla 6.5.1: Requerimientos del Log

6.5.1.2 Categorías

La Tabla 6.5.2 muestra los tópicos identificados como las categorías del usuario del log.

Categorías
privacy, privateness, secrecy, concealment
internet, cyberspace
exemption, freedom
electronic mail, e-mail, email
application, diligence

Tabla 6.5.2: Categorías asignadas al usuario

6.5.1.3 SWMiner vs. Google

La Tabla 6.5.3 describe las categorías asignadas en el directorio de *Google* a cada requerimiento, la Tabla 6.5.4 detalla las categorías finales de *Google* y la Tabla 6.5.6 muestra los valores de Precision y Recall obtenidos que permiten evaluar la performance del algoritmo de *Identificación de Tópicos* al comparar las categorías de las Tablas 6.5.2 y 6.5.4.

Id	Requerimiento	Categorías
1	http://java.sun.com	Computers > Programming > Language > Java
2	http://www.eff.org	Computers > Internet > History > People > Barlow, John Perry
3	http://www.dw-institute.com	Computers > Software > Databases > Data Warehousing

Tabla 6.5.3: Categorías de los requerimientos en el directorio de *Google*

Categorías
Computers, Programming, Language, Java, Internet , History, People, Barlow John Perry, Software, Databases , Data Warehousing

Tabla 6.5.4: Categorías de Google

Categorías	
Hits	internet, cyberspace, application
Misses	Computers, Programming, Language, Java, History, People, Barlow John Perry, Databases , Data Warehousing
Mistakes	privacy, privateness, secrecy, concealment , exemption, freedom, electronic mail, e-mail, email, , diligence

Tabla 6.5.5: Hits, Misses y Mistakes

Hits	Misses	Mistakes	Precision	Recall
3	9	10	0.23076923	0.25

Tabla 6.5.6: Precision y Recall para *SWMiner* Completo

Precision - Valor Promedio: 0.23076923

Recall - Valor Promedio: 0.25

El 23% de los tópicos obtenidos están relacionados con las categorías asignadas por *Google* y de todas las categorías de *Google* el algoritmo detectó el 25%.

6.5.2 SWMiner con Normalización Morfológica y sin Clustering

En esta sección describimos los resultados obtenidos al ejecutar *SWMiner* con normalización morfológica, es decir con *Lemmatiser* y *Part Of Speech* y sin aplicar el algoritmo de *Clustering*.

La Tabla 6.5.7 describe las categorías asignadas al usuario del log.

Categorías
thulium, Tm, atomic number 69
documentation, support
privacy, privateness, secrecy, concealment
public
internet, cyberspace
exemption, freedom
Indus, Indus River
residential district, residential area, community
application, diligence
database
electronic mail, e-mail, email

Tabla 6.5.7: Categorías asignadas al usuario

6.5.2.1 SWMiner vs. Google

La Tabla 6.5.8 describe qué categorías son *Hits*, *Misses* y *Mistakes* al comparar las Tablas 6.5.7 y 6.5.2.

Categorías	
Hits	internet, cyberspace, application, database
Misses	Computers, Programming, Language, Java, History, People, Barlow John Perry, Data Warehousing
Mistakes	thulium, Tm, atomic number 69, documentation, support, privacy, privateness, secrecy, concealment, public, exemption, freedom, Indus, Indus River, residential district, residential area, community, diligence, electronic mail, e-mail, email

Tabla 6.5.8: Hits, Misses y Mistakes

Hits	Misses	Mistakes	Precision	Recall
4	8	21	0.16	0.33333333

Tabla 6.5.9 Precision y Recall para SWMiner sin Clustering

Precision - Valor Promedio: 0.16

Recall - Valor Promedio: 0.33333333

De estos valores podemos concluir que el 16% de los tópicos obtenidos están relacionados con las categorías asignadas por *Google* y que de todas las categorías de *Google* el algoritmo detectó el 33%.

6.5.3 SWMiner sin Normalización Morfológica y con Clustering

A continuación detallamos los resultados obtenidos al ejecutar *SWMiner sin Normalización Morfológica* de las palabras que representan a cada requerimiento y su vecindad, y comparamos los resultados obtenidos con la ejecución completa de *SWMiner*.

Categorías
software, software system, internet, cyberspace, public, Indus, Indus River, contact, touch, capacity, content

Tabla 6.5.10: Categorías asignadas al usuario

6.5.3.1 SWMiner vs. Google

La Tabla 6.5.11 describe qué categorías consideramos *Hits*, *Misses* y *Mistakes* al comparar las Tablas 6.5.4 y 6.5.10.

Categorías	
Hits	software, software system, internet, cyberspace,
Misses	Computers, Programming, Language, Java, History, People, Barlow John Perry, Databases, Data Warehousing
Mistakes	public, Indus, Indus River, contact, touch, capacity, content

Tabla 6.5.11: Hits, Misses y Mistakes

Hits	Misses	Mistakes	Precision	Recall
4	9	7	0.36363636	0.30769231

Tabla 6.5.12: Precision y Recall para SWMiner sin NM y con Clustering

Precision - Valor Promedio: 0.36363636

Recall - Valor Promedio: 0.30769231

De estos valores podemos concluir que el 36% de los tópicos obtenidos están relacionados con las categorías asignadas por *Google* y que de todas las categorías de *Google* el algoritmo detectó el 30%.

6.5.4 SWMiner sin Normalización Morfológica y sin Clustering

A continuación detallamos los resultados obtenidos al ejecutar *SWMiner* sin Normalización Morfológica de las palabras ni Clustering, y comparamos los resultados obtenidos con la ejecución completa de *SWMiner*.

Categorías
ship's company, company, communication, software, software system, internet, cyberspace, Public, Indus, Indus River, contact, touch, subject, content, depicted object

Tabla 6.5.13: Categorías asignadas al usuario

6.5.4.1 SWMiner vs. Google

La Tabla 6.5.14 describe qué categorías consideramos *Hits*, *Misses* y *Mistakes* al comparar las Tablas 6.5.4 y 6.5.13.

	Categorías
Hits	software, software system, internet, cyberspace
Misses	Computers, Programming, Language, Java, History, People, Barlow John Perry, Databases, Data Warehousing
Mistakes	ship's company, company, communication, Public, Indus, Indus River, contact, touch, subject, content, depicted object

Tabla 6.4.14: Hits, Misses y Mistakes

Hits	Misses	Mistakes	Precision	Recall
4	9	11	0.26666667	0.30769231

Tabla 6.5.15: Precision y Recall para SWMiner sin NM y sin Clustering

Precision - Valor Promedio: 0.26666667

Recall - Valor Promedio: 0.30769231

Podemos observar que solamente el 26% de los tópicos obtenidos están relacionados con las categorías asignadas por *Google* y que de todas las categorías de *Google* el algoritmo detectó el 30%.

6.5.5 SWMiner vs. TOPIC

La Tabla 6.5.16 describe los tópicos hallados por *TOPIC* [W36] para los requerimientos del log en análisis y los tópicos sobre los cuales estos requerimientos son una autoridad.

url	Autoridad	Tópicos
http:// java.sun.com - The Source for Java Technology	Java	Developers, JavaSoft, Applets, JDK, Java applets, Sun Microsystems, API, Programming, Solaris, tutorial
http://www.eff.org/	Censorship, Net	Anti-Censorship, Join the Blue Ribbon, Blue Ribbon Campaign, Electronic Frontier Foundation, Free Speech

http://www.dw-institute.com/	Data Warehousing	TDWI, Data Warehousing Information Center, www.dw-institute.com , Data Warehousing Institute, data warehouse
---	------------------	--

Tabla 6.5.16: TOPIC(Lycos)

Si comparamos los resultados obtenidos con la ejecución completa de *SWMiner*, es decir con la aplicación de *Normalización Morfológica* de palabras y con *Clustering* (Tabla 6.5.2), con los tópicos obtenidos por TOPIC (Tabla 6.5.16) vemos que solamente hay unos pocos tópicos que coinciden o están relacionados, como ser *cyberspace*, *internet* y *application*. Muchos de los tópicos hallados por TOPIC son palabras que no existen en WordNet (JDK, API) o son nombres propios (*Sun Microsystems*, *Solaris*), por lo tanto fueron descartadas por *SWMiner*.

6.5.5.1 SWMiner vs. Categorías asignadas manualmente

En esta sección detallamos los resultados obtenidos al comparar las categorías asignadas al usuario con la ejecución completa de *SWMiner* (Tabla 6.5.2) contra las categorías asignadas en forma manual al observar el contenido de los documentos *Html* correspondientes a los requerimientos accedidos por el usuario (Tabla 6.5.18).

La Tabla 6.5.17 describe las categorías asignadas manualmente por requerimiento y la Tabla 6.5.18 contiene la unión de las categorías de todos los requerimientos accedidos por el usuario.

Requerimiento	Categorías
1	Java, Java Technologies, Java Developers, Java Documentation, Programming
2	Technologies, Software, Internet
3	Data Warehouse, Data Warehouse Research, Data warehouse applications and programs

Tabla 6.5.17: Categorías asignadas a mano

Categorías
Java, Java Technologies, Java Developers, Java Documentation, Programming, Technologies, Software, Internet, Data Warehouse, Data Warehouse Research, Data warehouses applications and programs

Tabla 6.5.18: Unión de Categorías asignadas a mano

	Categorías
Hits	Internet, cyberspace, application
Misses	Java, Java Technologies, Java Developers, Java Documentation, Programming, Technologies, Data Warehouse, Data Warehouse Research, Data warehouses applications and programs
Mistakes	Privacy, privateness, secrecy, concealment, exemption, freedom, electronic mail, e-mail, mail

Tabla 6.5.19: Hits, Misses y Mistakes

Hits	Misses	Mistakes	Precision	Recall
3	9	9	0.25	0.25

Tabla 6.5.20: Precision y Recall

Precision - Valor Promedio: 0.25

Recall - Valor Promedio: 0.25

El 25% de los tópicos obtenidos están relacionados con las categorías definidas manualmente y de todas las categorías especificadas a mano *SWMiner* identificó el 25%.

6.6 Evaluación de Resultados

Todos los experimentos fueron realizados en máquina COMPAQ AMD Duron 946 MHz con 256 Mb de memoria RAM. La Tabla 6.6.1 muestra los tiempos promedios de ejecución alcanzados por *SWMiner* con la aplicación de *Normalización Morfológica* y *Clustering*, para procesar un log con las características definidas en los experimentos anteriores.

Etapa	Tiempo		
	hr	min	Seg
SWMiner Completo	2	36	49
Depurador			1
Preprocesador	2	15	38
Ftp e Identificación del idioma de una página			29
Obtención de la vecindad de un requerimiento		10	35
Extracción de ítems de una página		14	33
Categorizador		21	10
Itemsets frecuentes por requerimiento y su vecindad			6
Clustering de itemsets frecuentes por requerimiento		4	15
Clustering de clusters		7	2

Tabla 6.6.1: Tiempos promedios de ejecución de *SWMiner*

El valor de *Ratio* utilizado ($R=0.68$) se definió a partir de los resultados alcanzados por *Chin-Yew Lin* en su trabajo de tesis *Robust Automated Topic Identification* [LIN97], en el cual concluye que con dicho valor se obtienen mejores tópicos de WordNet.

El valor de *Cantidad de Niveles* utilizado para acotar la vecindad de un requerimiento ($N=2$) se estableció experimentalmente luego de realizar varios ensayos con distintos valores y observar que al incrementar este valor se obtienen tópicos que se alejan de los tópicos reales del requerimiento en análisis. Por ejemplo, si tenemos un requerimiento que trata sobre ballenas, podemos obtener páginas relacionadas con la matanza ilegal de ballenas y de estas páginas acceder a otras relacionadas con la comercialización de productos provenientes de la carne de ballenas, como ser aceites. Con lo cual terminamos analizando páginas relacionadas a aceites y no a ballenas.

En la Tabla 6.6.2 se pueden observar los valores de *Precision* y *Recall* obtenidos para cada una de las etapas analizadas de *SWMiner*, en cada uno de los experimentos realizados y los valores promedios alcanzados. *E1*, *E2* y *E3* hacen referencia al experimento con el log de Mamíferos Marinos, al experimento con el log de Animales Salvajes y al experimento con el log General respectivamente.

Etapa	Precision				Recall			
	E1	E2	E3	Promedio	E1	E2	E3	Promedio
Identificación de Idiomas	1	0.94		0.97	0.93	0.87		0.9
Part of Speech	0.78	0.80		0.79	0.63	0.57		0.6
Lemmatiser	0.98	0.98		0.98	0.68	0.63		0.655
Identificación de Tópicos								
SWMiner con Normalización Morfológica y con Clustering	0.83	0.37	0.23	0.47	0.41	0.42	0.25	0.36

SWMiner con Normalización Morfológica y sin Clustering vs. Google	0.05	0.50	0.16	0.23	0.12	0.73	0.33	0.39
SWMiner sin Normalización Morfológica y con Clustering vs. Google	0.25	0.83	0.36	0.48	0.12	0.33	0.30	0.25
SWMiner sin Normalización Morfológica y sin Clustering vs. Google	0.16	0.47	0.26	0.29	0.12	0.62	0.30	0.34
SWMiner con Normalización Morfológica y con Clustering vs. Asignación manual	0.83	0.33	0.25	0.47	0.38	0.41	0.25	0.34

Tabla 6.6.2: Valores promedios de Precision y Recall para cada una de las etapas

Las investigaciones referentes a las medidas de *Precision* y *Recall* [EIK99, BG94] revelan que cuanto más cercanas a la unidad se encuentran ambas medidas, mejor es la performance del algoritmo que se está midiendo.

En la tabla 6.6.2 se puede observar que la eficacia de los algoritmos de *Identificación de Idiomas*, *Part of Speech* y *Lemmatiser* es muy buena. Por otro lado, tanto en las comparaciones de *SWMiner* vs. *Google* como así también en las comparaciones de *SWMiner* con la categorización manual de requerimientos, los valores promedio obtenidos de *Precision* y *Recall* en la medición de la eficacia del algoritmo de Identificación de Tópicos no son resultados fantásticos, pero combinándolos con los tópicos y resultados parciales obtenidos en cada experimento en particular, se puede concluir que con la aplicación de técnicas de *Normalización Morfológica de palabras* tales como *Part of Speech* y *Lemmatiser*, y con la aplicación del algoritmo de *Clustering* se obtienen los mejores resultados.

Analizando en detalle los tópicos y valores de *Precision* y *Recall* obtenidos en cada experimento particular, puede observarse que los tópicos identificados por *SWMiner* son siempre más específicos en comparación con las categorías definidas en el Directorio *Google*. Esto puede verse en el experimento con el log de animales mamíferos en donde los tópicos identificados por *SWMiner* son *killer whale*, *killer*, *orca*, *grampus*, *sea wolf* y *Orcinus orca* (ver Tabla 6.3.22), y las categorías definidas en el Directorio *Google* son *Animalia*, *Chordata*, *Mammalia*, *Cetacea*, *Issues*, *Animal Welfare*, *Specific Animals*, *Whales* (ver Tabla 6.3.24). Debido a esta diferencia de granularidad de categorización se producen algunos desvíos en el cálculo de los valores de *Precision* y *Recall*, principalmente en el cálculo de la cantidad de *Misses*, ya que los resultados obtenidos dependen mucho de la subjetividad de la persona que los analiza, siendo que en muchos casos sucede que los tópicos identificados por *SWMiner* son una especificación o especialización de las categorías de *Google*. En la Tabla 6.3.25, por ejemplo, se consideraron como *Misses* las palabras *Animalia*, *Chordata* y *Mammalia* siendo que *SWMiner* obtuvo como categorías a *killer whale*, *orca* y *sea wolf* entre otras (ver Tabla 6.3.22), que son una especialización de *Animalia*, *Chordata* y *Mammalia*.

Con el cálculo de la cantidad de *Mistakes* sucede algo similar; existen categorías que claramente pueden considerarse erróneas pero también existen otras, como por ejemplo *Privacy* (ver Tabla 6.5.36) que por su generalidad pueden ser consideradas un *Hit* o un *Mistake*, ya que la misma puede referirse a la privacidad de los datos en *Internet* o a la privacidad en otro contexto no relacionado con *Internet*.

6.7 Resumen

En el presente capítulo presentamos los resultados obtenidos al ejecutar *SWMiner* en todas sus variantes sobre 3 diferentes logs y comparamos dichos resultados con el Directorio *Google*, *TOPIC* y con las categorías asignadas manualmente al observar el contenido de los documentos *Html* correspondientes a los requerimientos accedidos por el usuario en análisis.

De todos los experimentos y comparaciones concluimos que los mejores resultados se obtienen al ejecutar *SWMiner* con *Normalización Morfológica* y *Clustering*.

CAPÍTULO 7: Conclusiones

7.1 Introducción

En este trabajo hemos presentado una arquitectura e implementación de un sistema para categorizar semánticamente los usuarios de una organización a partir de las páginas web a las que acceden, al que hemos dado por nombre *SWMiner* proveniente de *Semantic Web Miner*.

En las siguientes secciones, exponemos las conclusiones finales obtenidas, los aportes que consideramos realizados con *SWMiner*, y por último hacemos referencia a los posibles futuros trabajos que pueden realizarse a partir de la implementación de *SWMiner*.

7.2 Conclusiones

Con el trabajo de experimentación realizado en el Capítulo 6, hemos llegado a la conclusión que con la ejecución de *SWMiner* aplicando técnicas de *Normalización Morfológica de palabras* y aplicando el algoritmo de *Clustering* se obtienen los mejores resultados en la identificación de tópicos.

Es importante destacar que La aplicación de las técnicas *Part of Speech* y *Lemmatiser* juegan un papel importante en los resultados obtenidos por *SWMiner*, pues sin la correcta identificación de la categoría sintáctica de cada palabra pueden obtenerse significados semánticos erróneos de la jerarquía semántica de *WordNet*. Por ejemplo, si se tiene una oración en la cual aparece la palabra *figure* como verbo y se la reconoce como un sustantivo, se obtendrá un significado erróneo: *ilustration, body, important person* en lugar de *understand*.

A esta altura de la investigación y experimentación podemos sugerir que cuando los logs que se analizan contienen requerimientos a páginas web que tratan de un mismo tema y con un vocabulario específico, como vimos en el caso de mamíferos marinos o animales salvajes, *SWMiner* reconoce tópicos, más bien específicos, con un alto grado de similitud con los tópicos de interés del usuario en cuestión. Esto no siempre sucede cuando los logs que se analizan contienen requerimientos a páginas web de contenido semántico más amplio o genérico. Si bien todas las palabras que representan a los páginas web están relacionadas y/o hacen referencia a los tópicos de las páginas, las mismas pueden ser aplicadas en diversos contextos teniendo poco peso semántico para discriminar los tópicos de las páginas. Al acceder a *WordNet* con palabras muy genéricas, *SWMiner* obtiene tópicos más genéricos aún, con un significado semántico muy amplio aplicable a cualquier contexto, de lo que se desprende que el peso semántico de las palabras originales es muy pobre para distinguir un tópico de otro. Otra punto a tener en cuenta, es que en la gran mayoría de los casos los creadores de los documentos Html tienden a colocar palabras bien genéricas en los *Meta* tags con el fin de que sus documentos sean obtenidos en gran cantidad de las búsquedas que se realizan en los motores de búsqueda. También puede suceder que al aplicar el algoritmo de *Association Rules* [AS94] para obtener los itemsets frecuentes, las palabras más genéricas sean las que tienen mayor frecuencia de aparición en los documentos Html, con lo cual serán seleccionadas por el algoritmo y quizás se descarten palabras con mayor peso semántico pero que tienen menor frecuencia de aparición.

Comparando las categorías identificadas por *SWMiner* contra los tópicos obtenidos por *TOPIC* podemos asegurar que si bien existen coincidencias entre los resultados obtenidos con ambos sistemas, la calidad semántica de los tópicos identificados por *SWMiner* es superior. *TOPIC* obtiene en muchos casos urls como tópicos de una página; por ejemplo, para el requerimiento <http://www.geocities.com/RainForest/4076/index1.html> del log de

animales salvajes identificó www.geocities.com (ver Tabla 6.4.40) como tópico de la página, el cual tiene pobre contenido semántico para conceptualizar el requerimiento. Cabe remarcar también, que en muchos casos *TOPIC* no detecta tópico alguno para determinadas páginas.

7.3 Aportes

En este trabajo hemos presentado una arquitectura e implementación de un sistema para categorizar semánticamente los usuarios de una organización a partir de las páginas web a las que acceden.

Si bien, aún es necesario evaluar y revisar aspectos relacionados con la efectividad, performance, optimización y escalabilidad de nuestra implementación en términos de un sistema comercial que traspase los límites de la investigación y experimentación, sentimos que nuestra implementación contribuye significativamente a los sistemas de *Information Extraction (IR)* utilizando en combinación algoritmos tales como *Association Rules*, *Clustering* y la generalización de conceptos a partir de la jerarquía semántica de *WordNet* para alcanzar una categorización de usuarios a partir de una categorización semántica de los documentos web a los que acceden.

Modificamos el algoritmo de identificación de tópicos de Chin-Yew Lin[LIN97] para obtener como conceptualización de las palabras involucradas los padres en común más bajos o específicos en la jerarquía de *WordNet*. Mejoramos la eficacia de dicho algoritmo, para un caso no contemplado por Chin-Yew Lin; modificamos el algoritmo para contemplar aquellos casos de generalización de palabras en los cuales las palabras involucradas en la generalización pertenecen a una misma rama dentro de la jerarquía de *WordNet*; es decir una de las palabras es padre de la otra o viceversa, ya sea de forma inmediata o en una determinada cantidad de niveles.

Definimos medidas propias de similaridad de palabras y similaridad semántica entre itemsets frecuentes y clusters utilizadas por el algoritmo de clustering para agrupar aquellos clusters de itemsets frecuentes más similares, introduciendo una cota de mínima similaridad para producir clusters más cohesivos semánticamente.

7.4 Trabajos Futuros

Actualmente, el sistema trabaja solamente con sustantivos extraídos de todos los tags considerados relevantes de los documentos Html. La razón que motiva esta decisión es que consideramos que los sustantivos son la categoría sintáctica más expresiva semánticamente y son más factibles de ser tópicos candidatos. La implementación de *TOPIC* [W36] se basó en una suposición similar, trabajando solo con los abstracts devueltos por los motores de búsqueda. Aunque en un principio la utilización solo de sustantivos sea una decisión buena y económica para la identificación de tópicos, es interesante considerar en el futuro el poder de la información extra provista por verbos, adjetivos y adverbios.

Otras posibles extensiones son trabajar con otras jerarquías de *WordNet* [W30] además de *hypernym*; no descartar las palabras que son nombres propios; acceder a una jerarquía semántica más rica como puede ser *SENSUS*; analizar páginas web escritas en distintos idiomas, para lo cual es necesario acceder a alguna jerarquía multilenguaje como *EuroWordNet*; utilizar un algoritmo diferente de *Clustering*, que no utilice una cota de similaridad mínima para agrupar los clusters, si no que automáticamente determine la cantidad de clusters existentes como por ejemplo el algoritmo *EM*.

La performance del sistema es un punto clave a mejorar en el futuro, tanto en cuestiones de tiempos de ejecución como en términos de resultados obtenidos. Algunos puntos relacionados son: limitar la extracción de palabras a unos pocos tags nada más, aquellos que resulten ser los más significativos para el análisis; extraer de los documentos Html

palabras que existan a partir de un determinado nivel en la jerarquía de *WordNet* para evitar extraer palabras muy genéricas que no sirven para distinguir tópicos entre sí.

Referencias

- [AS94] Rakesh Agrawal, Ramakrishnan Srikant – Fast Algorithms for Mining Association Rules – VLDB 1994: 487-499
- [ASI93] Rakesh Agrawal, Arun N. Swami, Tomasz Imielinski– Mining association rules between sets of items in large databases – SIGMOD Conference 1993: 207-216
- [BG94] Michael K. Buckland, Fredric Gey – The relationship between Recall and Precision – JASIS 45(1): 12-19 (1994)
- [BH98] Krishna Bharat and Monika R. Henzinger – Improved Algorithms for Topic Distillation in a Hyperlinked Enviroment - . SIGIR 1998: 104-111
- [CAV94] William B. Cavnar – Using An N-Gram- Based Document Representation With A Processing Retrieval Model - TREC 1994: 0
- [CCR00] Chris Clifton, Robert Cooley, Jason Rennie – TopCat: Data Minig for Topic Identification in a Text Corpus – PKDD 1999: 174-183
- [CMS97] R. Cooley, B. Mobasher and J. Srivastava – Web Mining: Information and Pattern Discovery on the World Wide Web - Department of Computer Science and Engineering, University of Minnesota, Minneapolis, USA.
- [CT94] William B. Cavnar and John M. Trenkle – N-Gram-Based Text Categorization - Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval
- [DAHKRV97] David Day, John Aberdeen, Lynette Hirschman, Robyn Kozierok, Patricia Robinson y Marc Vilain – “Mixed initiative development of language processing systems” – *Proceedings of the Fifth Conference on Applied Natural Language Processing*, Washington, D.C., March 31- April 3, 1997.
- [DH99] Jeffrey Dean and Monika Henzinger – Finding Related Pages in the World Wide Web – WWW8 / Computer Networks 31(11-16): 1467-1479 (1999)
- [EIK99] Like Eikvil – Information Extraction from World Wide Web – A Survey -Norwegian Computing Center, Oslo – Norweigan – July 1999.
- [FP98] Paula Cecilia Fritzsche, César Dario Polakoff - Clustering de Datos- Departamento de Computación, Facultad de Ciencias Exactas y Naturales, UBA.
- [GGV96] Julio Gilarranz, Julio Gonzalo, Felisa Verdejo - An Approach to Conceptual Text Retrieval Using the EuroWordNet Multilingual Semantic Database - Working Notes of AAAI Spring Symposium on CrossLanguage Text and Speech Retrieval, Stanford, CA, 1997
- [GON96] Osvaldo González – Fundamentos de Data Mining - Cátedra de Data Mining – UBA
- [HKK97] Eui-Hong Han, George Karypis and Vipin Kumar – Clustering based on Association Rules Hipergraphs – Department of Computer Science, University of Minnesota.
- [HM96] Oskari Heinonen, Heikki Mannila – Attribute-Oriented Induction and Conceptual Clustering - DMKD 1997: 0
- [HSU98] C. H. Hsu - Initial Results on Wrapping Semistructured Web Pages with Finite State Transducers and Contextual Rules - Workshop on AI and Information Integration, in conjunction with the 15'th National Conference on Artificial Intelligence (AAAI-98), Madison, Wisconsin, July 1998.
- [JD98] A.K Jaim and R. C. Dubes – Algorithms for Clustering Data – Prentice Hall, 1998
- [KF00] Claus-Peter Klas , Norbert FUT – A new Effective Approch for Categorizing Web Documents – Proceedings of the 22th BCS-IRSG Colloquium on IR Research 2000, 22nd Annual Colloquium on IR Research 5th-7th April 2000, Sidney Sussex College, Cambridge, England
- [KK98] George Karypis and Vipin Kumar – Multilevel k-way Hypergraph Partitioning –

- Department of Computer Science & Engineering, Army HPC Research Center and University of Minnesota. *DAC* 1999: 343-348
- [KLE98] Jon M. Kleinberg – Authoritative Source in Hyperlinked Enviroment – IBM Almaden Research Center. *JACM* 46(5): 604-632 (1999)
- [KR98] Balacharder Krishnamurthy y Jennifer Rexford – Software issues in characterizing web server logs- AT&T Labs-Research.
- [LIN97] Chin-Yew Lin - Robust Automated Topic Identification – *Ph.D. Thesis*. University of Southern California, August 1997.
- [LK99] Krista Lagus and Samuel Kaski – Keyword Seleccion Method for Characterizing Text Document Maps - Proceedings of ICANN99, Ninth International Conference on Artificial Neural Networks, IEE, London, vol. 1, pp. 371-376, 1999.
- [MCS97] B. Mobasher, R. Cooley y J. Srivastava – Grouping Web Page References into Transactions for Mining World Wide Web Browsing Patterns - Knowledge and Data Engineering Workshop, pp. 2-9, Newport Beach, CA, 1997
- [MR00] Alberto O. Mendelzon and Davood Rafiei – What do the Neighbours Think? Computing Web Page Reputations – *IEEE Data Engineering Bulletin* 23(3): 9-16 (2000)
- [MS96] Dharmendra S. Modha, W. Scott Spangler – Clustering Hypertext with Application to Web Searching – *Hypertext 2000*: 143-152
- [RH00] Vijayshankar Raman, Joseph M. Hellersteing – An Interactive Framework for Data Cleaning - *WWW9 / Computer Networks* 33(1-6): 823-835 (2000)
- [RM00] Davood Rafiei, Alberto O. Mendelzon – What is this Page Known for? Computing Web Page Reputations - Department of Computer Science, University of Toronto, Canada.
- [SAB94] Gerard Salton, James Allan y Chris Buckley – Automatic structuring and retrieval of large text files – *CACM* 37(2): 97-108 (1994)
- [TUSVMN97] Dick Tsur, Jeffrey D. Ullman, Serge Abiteboul, Chris Clifton, Rajeev Motwani, Svetlozar Nestorov – Query Flocks: A Generalization of Assocation Rules Mining – ACM SIGMOD Intl. Conf. Management of Data, June 1998.

Referencias Web

- [W1] www.h2mweb.com/logfiles.htm *The importance of Logfiles*
- [W2] www.docs.rinet.ru/ch20.html - *Charter 20: Usage Statistics and Log Análisis*
- [W3] vtopus.cs.vt.edu/~chitra/docs/96w3j/96w3j.html
Complementing Surveying and Demographics with Automated Network Monitoring Marc Abrams y Stephen Williams – Department of Computer Science Virginia Tech
- [W4] www.baculabs.com/WsvlCLF.html - *Common Log Format – HTTP Web Server Log Format*
- [W5] www.isp-planet.com/equipment/clog_intro.html - *Analyzing Cache Log* – Lisa Phifer
- [W6] www.ljudtechnik-sporre.se/syshelp/logfmt.htm - *Sambar server – Log File Formats*
- [W7] www.w3.org/pub/WWW/TR/WD-logfile.html - *Extended Log File Format*
- [W8] www.w3.org/Daemon/User/Config/Logging.html - *Loggin Control In W3C*
- [W9] www.w3.org/1999/05/WCA-terms/01 - *"Web Characterization Terminology & Definitions Sheet*
- [W10] www.w3.org/pub/WWW/TR/WD-proxy-960221.html - *Notification for Proxy Caches*
- [W11] [www. accru e.com/hitlist/techdb/t db-logs.htm](http://www accru e.com/hitlist/techdb/t db-logs.htm) - *Log Files Definitions and examples*
- [W12] httpd.apache.org/docs/mod/mod_log_common.html - *Apache HTTP Server Version 1.3 – Module mod_log_common*
- [W13] www.microsoft.com/technet/Sitesrv/Manuals/SSExpGde/SSExpUa/uan4.asp - *Usage Import*

- [W14] www.research.ibm.com/journal/sj/371/wu.html - *SpeedTracer: A Web usage mining and analysis tool* - K.-L. Wu, P. S. Yu y A. Ballaman - *IBM System Journal*
- [W15] www.w3.org/WCA - Word Wide Web Comité Web Usage Characterization Activity
- [W16] www.w3.org/TR/html4 - Html 4.0 Specification - World Wide Web Consortium (W3C)
- [W17] babel.altavista.com/translate.dyn - Altavista - World / Translate
- [W18] www.freeTranslation.com
- [W19] www.lib.fnal.gov/library/transdics.html - Translation Directories
- [W20] www.word2word.com/dictionary.html - Language Directories & Translators
- [W21] users.info.unicaen.fr/~giquet/classif/part0.html - Categorization accodint to Language
- [W22] www.let.rug.nl/~vannoord/TextCat/ - TextCat
- [W23] www.comp.lancs.ac.uk/computing/research/stemming/general/index.htm - What is Stemming?
- [W24] www.onsee.com/developer/docs/porterstem.html - An Algorithm for suffix stripping - M. F. Porter
- [W25] clir.cs.umass.edu/whatsnew/stemming.html - Stemming and Cooccurrence on a Larger Corpus - Jeremy Pickens
- [W26] www.ifi.unizh.ch/CL/extrans/secondy/node24.html - The Lemattiser
- [W27] ixa.si.ehu.es/ingeles/dokument/EUSLEnag.html - EUSLEM: A Lemmatiser/Tagger for Basque
- [W28] www.phrasys.com/software/Lemmatiser.html - Phrasys NLP Software: Lemmatiser
- [W29] www.askeric.org/Eric/Help/stop.shtml, local.dcs.gla.ac.uk/ldom/ir_resources/linguistic_utils, www.indiana.edu/cgi-bin-local/doIsearch.pl?Stopwords, [ftp://ftp.cs.cornell.edu/pub/smart/english.stop](http://ftp.cs.cornell.edu/pub/smart/english.stop) - Stopword for English
- [W30] www.cogsci.princeton.edu/~wn/ - WordNet
- [W31] web.bham.ac.uk/O.Mason/software/tagger/ - Part of Speech Tagger Oliver Mason
- [W32] www.dlg.bham.ac.uk/QTAG/description - Oliver Mason - QTAG A Portable Probabilistic Tagger - Corpus Research, The University of Birmingham, United Kindong.
- [W33] www.support.microsoft.com/default.aspx?scid=kb;en-us;Q234147 - Web Proxy Server 2.0 Log File Format.
- [W34] www.lib.uconn.edu/lib-assignments/webglossary.htm - *Web Glosary*
- [W35] www.h2mweb.com/glossary.htm - *Glosary*
- [W36] www.cs.toronto.edu/db/topic - TOPIC
- [W37] <http://directory.google.com/> - Directorio de Google
- [W38] <http://dmoz.org/about.html> - Open Directory Project

Anexo A

WordNet

Por cada una de las 4 categorías sintácticas que maneja WordNet (NOUN, ADVERB, ADJECTIVE, VERB) se necesitan 2 archivos para representar el contenido de la base de datos WordNet, *index.pos* y *data.pos* donde *pos* es **noun**, **adv**, **adj**, **verb**.

Cada uno de los archivos **index** es una lista alfabética de todas las palabras que se encuentran en WordNet en su correspondiente *part of speech*. En cada línea, a continuación de la palabra, hay una lista de desplazamientos en bytes (*synset_offset* s) sobre el respectivo archivo **data**, un desplazamiento por cada synset que contiene la palabra.

El archivo **data** para cada categoría sintáctica contiene información correspondiente a los synsets. Cada línea corresponde a un synset con punteros relacionales a otros synsets a través del campo *synset_offset* s. Siguiendo estos punteros se navega la jerarquía conceptual pasando de un synset a otro.

Todos los archivos de la base de datos están en formato ASCII, lo cual permite que sean fácilmente legibles tanto por humanos como por máquinas. Todos los campos están separados por un espacio en blanco, y todas las líneas terminan con un carácter de nueva línea. Los campos encerrados entre corchetes([]) pueden no estar presentes.

Formato de los archivos index de WordNet[W32]

Cada archivo **index** comienza con varias líneas conteniendo *copyright*, *número de versión* y *acuerdo de licencia*. Todas estas líneas comienzan con 2 espacios en blanco y el número de línea, así no interfieren con el algoritmo de búsqueda binaria que es usado para buscar entradas en los archivos **index**. Todas las demás líneas tienen el siguiente formato.

lemma pos sense_cnt p_cnt [ptr_symbol...] sense_cnt tagsense_cnt synset_offset [synset_offset...]

Campos de la entrada del Archivo Index	
lemma	Texto ASCII de la palabra o colocación. Una colocación está formada por la unión de las palabras individuales con un caracter de guión bajo (_).
pos	Categoría sintáctica: n para archivos de sustantivos(noun), v para archivos de verbos(verb), a para archivos de adjetivos(adjective), r para archivos de adverbios(adverb).
poly_cnt	Cantidad de significados que tiene el lemma en WordNet.
sense_cnt	Cantidad de synsets a los cuales pertenece lemma. Esta es la cantidad de significados de la palabra en WordNet.
ptr_symbol	Una lista de los distintos tipos de punteros que tiene el lemma en todos los synsets a los que pertenece.
sense_cnt	Idem sense_cnt. Es redundante, pero el campo se conserva por razones de compatibilidad.
tagsense_cnt	Cantidad de significados del lemma que son rankeados de acuerdo a su frecuencia de aparición en la semántica de textos.
synset_offset	Desplazamiento en bytes del synset que contiene al lemma dentro del archivo data.pos. Cada synset_offset en la lista corresponde a un significado diferente del lemma en WordNet.

Formato de los archivos data de WordNet [W32]

Cada línea de los archivos **data** tiene el siguiente formato:

synset_offset lex_filenum ss_type w_cnt word lex_id [word lex_id...] p_cnt [ptr...] [frames...] | gloss

Campos de la entrada del archivo Data	
synset_offset	8 dígitos enteros que representan el desplazamiento actual en bytes dentro del archivo.
lex_filenum	2 dígitos enteros que corresponden al nombre del archivo lexicográfico que contiene el synset.
ss_type	1 carácter que indica el tipo del synset: n NOUN, v VERB, a ADJECTIVE, s ADJECTIVE SATELLITE y r ADVERB
w_cnt	2 dígitos hexadecimales que indican la cantidad de palabras del synset.
Word	Formato ASCII de la palabra en el synset con los espacios reemplazados por caracteres de guión bajo (_). El texto de la palabra es sensitivo al contexto(case sensitive).
lex_id	1 dígito hexadecimal que anexado a una palabra identifica unívocamente un sentido de dicha palabra. lex_id generalmente comienza con 0, y se va incrementando a medida que se agregan sentidos adicionales de la misma palabra al archivo, aunque no es un requisito que los números sean consecutivos o comiencen con 0.
p_cnt	3 dígitos enteros que indican la cantidad de punteros existentes desde el synset hacia otros synsets. Si p_cnt es 000 el synset no posee punteros.
frames	Solamente aplica al archivo de verbos, es una lista de números que corresponden a los distintos frames de tipos de sentencias simples généricas en los cuales los verbos del synset pueden ser usados. frames es de la forma: $f_cnt + f_num\ w_num\ [+ f_num\ w_num...]$ donde <i>f_cnt</i> es un número de 2 dígitos enteros que indica la cantidad de frames genéricos listados, <i>f_num</i> es un número de 2 dígitos enteros que especifica el número de frame, y <i>w_num</i> es un entero de 2 dígitos hexadecimales que indica la palabra en el synset sobre la cual el frame aplica. Como con los punteros, si este número es 00, <i>f_num</i> aplica sobre todas las palabras <i>s</i> en el synset. Si es no nulo, es aplicable solamente a la palabra indicada. Cada par <i>f_num w_num</i> es precedido por el signo +.
gloss	Un <i>gloss</i> se representa por una barra vertical (), seguida por un texto que continúa hasta el final de la línea. Puede contener una definición, uno o más ejemplos, o ambos.
Ptr	Un puntero desde el synset hacia otro es de la forma: $pointer_symbol\ synset_offset\ pos\ source/target$ donde <i>synset_offset</i> es el desplazamiento en bytes del synset en el archivo de datos correspondiente a pos. El campo <i>source/target</i> distingue punteros léxicos y semánticos. Es un campo de 4 bytes, conteniendo 2 números de 2 dígitos hexadecimales. Los primeros 2 dígitos indican el número de la palabra en el synset actual, los 2 últimos dígitos indican el número de la palabra en el synset destino. Un valor de 0000 significa que <i>pointer_symbol</i> representa una relación semántica entre el synset actual y el synset destino indicado por <i>synset_offset</i>. Una relación léxica entre 2 palabras de diferentes synsets se representa por valores no nulos en los números de palabra fuente y destino. <i>Lista de pointer_symbol s para sustantivos:</i>

!	Antonym
@	Hypernym
~	Hyponym
#m	Member holonym
#s	Substance holonym
#p	Part holonym
%m	Member meronym
%s	Substance meronym
%p	Part meronym
=	Attribute
<i>Lista de pointer_symbol s para verbos:</i>	
!	Antonym
@	Hypernym
~	Hyponym
*	Entailment
>	Cause
^	Also see
\$	Verb Group
<i>Lista de pointer_symbol s para adjetivos:</i>	
!	Antonym
&	Similar to
<	Participle of verb
\	Pertainym (pertains to noun)
=	Attribute
^	Also see
<i>Lista de pointer_symbol s para adverbios:</i>	
!	Antonym
\	Derived from adjective

Ejemplos de Synsets

Synsets de sustantivos(noun):

Si buscamos en WordNet los synsets correspondientes a *Apple* en el archivo *noun.dat* encontramos:

Synset 1

05775435 13 n 01 apple 0 006 @ 05745250 n 0000 @ 09426892 n 0000 #p 08933828 n 0000 ~ 05775654 n 0000 ~ 05775771 n 0000 ~ 05778237 n 0000 | fruit with red or yellow or green skin and sweet to tart crisp whitish flesh

Synset 2

08933828 20 n 03 apple 0 orchard_apple_tree 0 Malus_pumila 0 003 @ 08933472 n 0000 #m 08933220 n 0000 %p 05775435 n 0000 | native Eurasian tree widely cultivated in many varieties for its firm rounded edible fruits

Veamos en detalle el parseo del synset 1:

Campos de la entrada del archivo Data	
synset_offset	05775435
lex_filenum	13
ss_type	n (noun) , es decir sustantivo
w_cnt	01, es decir el synset contiene 1 palabra
word	Apple
lex_id	0, index de la palabra en el synsets
p_cnt	006, es decir este synset apunta a 6 synsets
ptr	@ 05745250 n 0000 significa que mediante la relación <i>Hypernym</i> (Es un) el synset apunta al synset cuya offset es 05745250, de tipo <i>n</i> (noun) y 0000 indica que la palabra con index 0 del synset actual se relaciona con la palabra de index 0 del synset con offset 05745250. Los otros 5 punteros de este synsets son: @ 09426892 n 0000 #p 08933828 n 0000 ~ 05775654 n 0000 ~ 05775771 n 0000 ~ 05778237 n 0000
frames	No se aplica al archivo noun.dat
Gloss	<i>fruit with red or yellow or green skin and sweet to tart crisp whitish flesh</i> Definición del synset

Anexo B

Códigos de Estado del Protocolo HTTP

A continuación se detallan los códigos de estado (*status*) más comunes correspondientes al Protocolo HTTP.

- 200 – Requerimiento devuelto con éxito.
- 201 – Requerimiento creado.
- 202 – Requerimiento aceptado.
- 206 – Requerimiento incompleto.
- 301 – La URL ha sido reemplazada.
- 302 – La URL ha sido redireccionada a otro documento.
- 400 – El requerimiento realizado por el cliente está mal.
- 401 – -Se requiere autorización para acceder al documento.
- 403 – El acceso a este documento está prohibido.
- 404 – No se encontró el documento.
- 408 – Requerimiento perdido por timeout.
- 500 – Error interno del servidor.
- 501 – El método GET o POST está incompleto
- 503 – El Servidor no está activo.

Anexo C

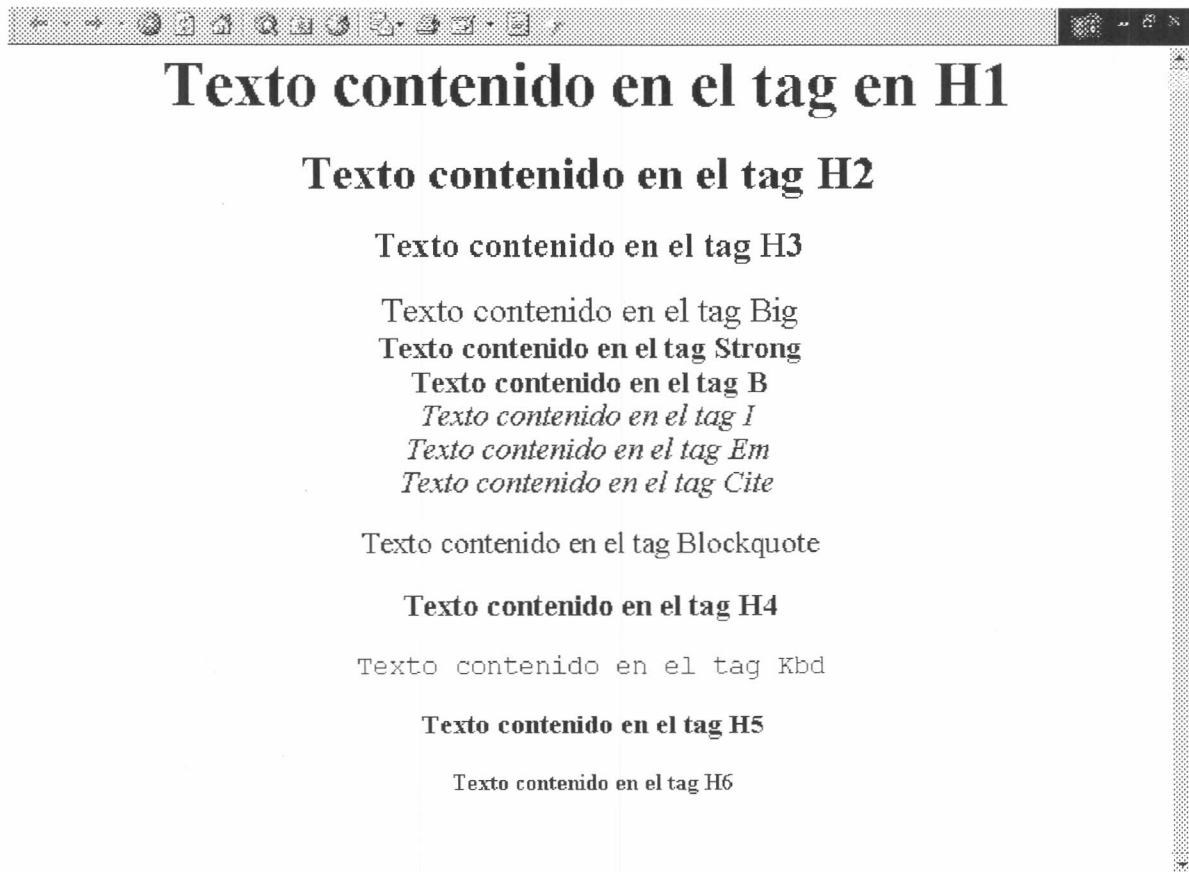
Ejemplo de Archivo de Perfiles de Idioma

En el ejemplo podemos observar los primeros ngrams correspondiente al archivo asociado al idioma inglés. Como puede observarse, los ngrams están ordenados de mayor a menor según su puntaje. En este ejemplo mostramos los ngrams en 4 columnas pero recordemos que cada línea del archivo contiene sólo un ngram y su puntaje.

_	20326	y_	595	.	329	ts	196
e	6617	of	594	_to_	320	di	196
t	4843	_of	592	co_	317	es_	195
o	3834	es	589	ic	316	si_	194
n	3653	ti	587	ns	308	re_	193
i	3602	v	580	al	307	at_	192
a	3433	_of_	575	le	304	nc	192
s	2945	of_	575	ou	304	ie	190
r	2921	nd_	568	ce	293	_a_	188
h	2507	at	549	ent	279	_in_	185
e_	2000	r_	540	l_	278	ing	184
d_	1816	_w	534	_co	277	us	182
_t	1785	it	522	tio	275	_re	182
c	1639	ed	496	on	274	g_	179
l	1635	_p	494	_d	274	ng_	178
th	1535	nt	485	tion	268	op	178
he	1351	_c	462	ri	266	con	177
th	1333	o	457	_e	264	tha	175
u	1309	io	450	ng	253	_l	174
f	1253	_an	439	hi	251	_tha	174
m	1175	te	432	er_	249	ver	173
p	1151	or	425	ea	246	ma	173
_a	1145	_b	418	as	245	ion_	171
the	1142	nd_	407	_be	242	_con	171
_the	1060	to	406	pe	242	ci	170
s_	978	st	402	h_	234	ons	170
er	968	is	401	_r	232	_it	170
_o	967	_s	396	ec	227	po	169
he_	928	_in	389	ch	223	ere	168
d_	888	ion	385	ro	222	is_	167
t_	885	and	385	ct	220		
the_	844	de	384	_h	219		
the	843	ve	382	pr	217		
on	842	ha	375	in_	217		
in	817	ar	366	ne_	214		
y	783	_m	361	ll	214		
n_	773	and_	360	rt	213		
b_	761	_and	360	s,_	210		
re	754	_and_	358	s,	210		
,	734	se	353	li	209		
,_	732	_to	347	ra	208		
an	732	me	346	T	207		
g	728	to_	344	wh	204		
w	718	ed_	339	a_	203		
_i	707	.	330	ac	201		
en	676	be	329	_wh	199		
f_	599	_f	329	_n	196		

Anexo D

Ejemplo de Tags Html



Anexo E

Stop Words

Los archivos *StopWord* [41], *StopsUrls* y *StopWordsUrls* contienen aquellas palabras o *url's* que no se deseamos analizar porque contienen información poco relevante y dependen del idioma del texto que se está siendo analizado. En el caso particular del *Preprocesador* se restringe el idioma al *Inglés*. A continuación se detallan algunas de palabras y *url's* contenidas en dichos archivos.

StopWords

a	cause	formerly	get	some
a's	causes	forth	gets	somehow
able	certain	forty	getting	someone
about	certainly	found	give	something
above	changes	four	given	sometime
according	clearly	from	gives	sometimes
accordingly	cc	front	go	somewhat

StopUrls

windowsupdate.microsoft.com
www.uol.com.ar
www.yahoo.com
ar.yahoo.com
www.altavista.com
www.lycos.com
www.google.com
ads7.speedbit.com

StopWordsUrls

Este archivo contiene palabras involucradas con información confidencial de los usuarios que realizaron los requerimientos. Estas palabras fue deducidas del analisis detallado de los logs del proxy. Por ejemplo, si el usuario accede a una página web para pagar algún servicio, se agregó a este archivo el nombre de la página para evitar categorizarla semánticamente.

Anexo F

Experimentación

F.1 Experimentos con Log de Animales Salvajes

F.1.1 Parámetros de SWMiner

En la tabla F.1.1 se detalla el valor de los parámetros y/o variables del sistema utilizados en este experimento.

Parámetro	Valor
Motor de Búsqueda	Altavista
Cantidad de In Links por Requerimiento	5
Cantidad de Out Links por Requerimiento	5
Cantidad de In Links por Link	2
Cantidad de Out Links por Link	2
Cantidad de Niveles	2
Cantidad de Palabras por Página	50
Soporte Association Rules	0,25
Similaridad Mínima entre Clusters	0,65
WordNet – Ratio	0,68
WordNet – Profundidad Mínima	2

Tabla F.1.1: Parámetros de SWMiner

F.1.2 Requerimientos y su Vecindad

Requerimientos del Log

Id	Requerimiento
1	http://www.nature-wildlife.com/girtxt.htm
2	http://www.cathouse-fcc.org/catsinfo.html
3	http://www.geocities.com/RainForest/4076/index1.html

Tabla F.1.2: Requerimientos del Log

Vecindad de los requerimientos

Las Tablas F.1.3, F.1.4 y F.1.5 y las Figuras F.1.1, F.1.2 y F.1.3 muestran los in y out links obtenidos para cada uno de los requerimientos.

Id Link	Id Link Padre	Tipo	URL
1	NULL	I	http://www.nature-wildlife.com/girtxt.htm
10	NULL	I	http://www.mrspainter.com/mammalsdk.htm
12	NULL	I	http://schoolweb.missouri.edu/ashland.k12.mo.us/amber/HOMEPAGE.htm

13	12	I	http://schoolweb.missouri.edu/ashland.k12.mo.us/student2.html
14	NULL	I	http://evreka.passagen.se/g/Library/Sciences/Animals_and_Wildlife/Mammals/Mammals_A-Z/Mammals_G/Giraffes
15	NULL	I	http://www.looksmart.co.nz/eus1/eus302562/eus317834/eus163430/eus163792/eus217721/eus226678/eus21772
16	NULL	I	http://soeg.sol.dk/g/Personal/Home/Pets_and_Animals/Wildlife/Mammals/Mammals_A-Z/Mammals_G/Giraffes
17	NULL	O	http://www.nature-wildlife.com/gir0.html
18	17	O	http://www.nature-wildlife.com/gir61s.html
19	17	O	http://www.nature-wildlife.com/gir62s.html
20	NULL	O	http://www.nature-wildlife.com/mammals.htm
21	20	O	http://www.nature-wildlife.com/copy.html
22	20	O	http://www.nature-wildlife.com/welcome.htm

Tabla F.1.3: In y Out Links del Requerimiento 1

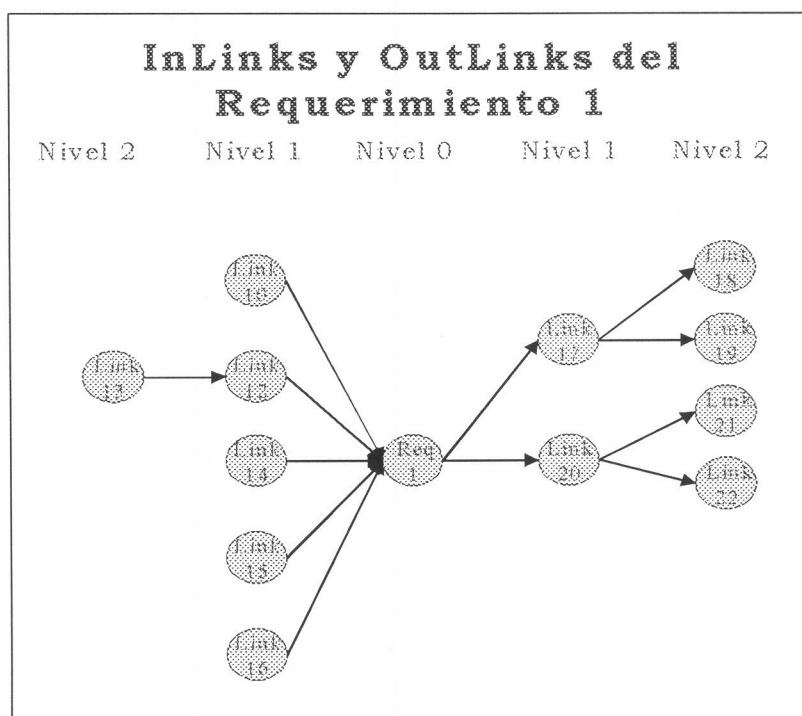


Figura F.1.1: In y Out Links del Requerimiento 1

Id Link	Id Link Padre	Tipo	URL
2	NULL	I	http://www.cathouse-fcc.org/catsinfo.html
24	NULL	I	http://www.kidinfo.com/Science/Endangered_animals.html
25	24	I	http://www.kidinfo.com/
26	24	I	http://www.kidinfo.com/SchoolSubjects.html
27	NULL	I	http://www.nemeton.demon.co.uk/Links.html
28	NULL	I	http://www.pbs.org/wgbh/nova/leopards/resources.html
29	28	I	http://www.wtv-zone.com/BigCats/links/otherlist.html
30	28	I	http://www.pbs.org/wgbh/nova/leopards/images/leopardshomeimage.map
31	NULL	I	http://www.datastatisticsonline.com/lynx.htm
32	NULL	I	http://dmoz.org/Science/Biology/Flora_and_Fauna/Animalia/Chordata/Mammalia/Carnivora/Felidae/
33	32	I	http://www.clouded-leopards.com/clouded-leopards-consv.htm
35	NULL	O	http://www.primenet.com/~brendel/
36	35	O	http://www.primenet.com/~brendel/carniv.html
37	35	O	http://www.primenet.com/~brendel/felidae.html
38	NULL	O	http://dialspace.dial.pipex.com/agarman/
39	NULL	O	http://www.qnet.com/
40	39	O	http://home.qnet.com/qnetnews/

41	39	O	http://weather.qnet.com/
43	NULL	O	http://www.cathouse-fcc.org/bobcat.html
44	43	O	http://www.cathouse-fcc.org/births.html
45	43	O	http://www.cathouse-fcc.org/births_2000.html
46	NULL	O	http://www.cathouse-fcc.org/canlynx.html
47	46	O	http://www.cathouse-fcc.org/births01.html
48	46	O	http://www.cathouse-fcc.org/sounds.html

Tabla F.1.4: In y Out Links del Requerimiento 2

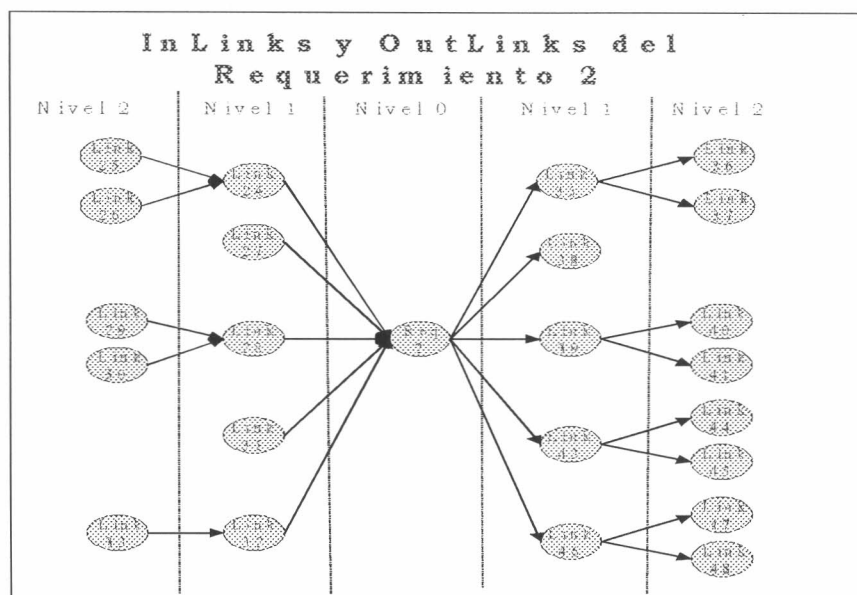


Figura F.1.2: In y Out Links del Requerimiento 2

Id Link	Id Link Padre	Tipo	Link
3	NULL	I	http://www.geocities.com/RainForest/4076/index1.html
53	NULL	I	http://www.ancientskillsschool.com/links.htm
54	NULL	I	http://www.thoughtandmemory.com/links.htm
55	54	I	http://www.thoughtandmemory.com/index1.htm
57	NULL	I	http://www.k12.nf.ca/stgeorgeshigh/Website4.htm
58	57	I	http://www.k12.nf.ca/stgeorgeshigh/default.htm
59	NULL	I	http://www.atlantareiki.com/links.html
61	NULL	I	http://www.nightingail.com/nglinks.html
62	61	I	http://www.nationsonline.org/oneworld/linkpage.htm
63	NULL	O	http://forums.delphiforums.com/animalspirits/start/
64	63	O	http://www.delphiforums.com/index.htm
65	63	O	http://www.prospero.com/about.htm
66	NULL	O	http://www.amazon.com/exec/obidos/ASIN/1580172318/animalspiritguid
67	66	O	http://s1.amazon.com/exec/varzea/ts/announcement-list-zshops/slp/
68	66	O	http://www.amazon.com/exec/obidos/subst/home/all-stores.html/ref%3Dtab_gw_storesdirectory/
69	NULL	O	http://www.animalvoices.net/default.asp
70	69	O	http://www.gotoit.com/titles/anivoi.html
71	69	O	http://www.amazon.com/exec/obidos/ASIN/1879181916/qid=1019597272/sr=1-3/animalvoices-20
72	NULL	O	http://www.webring.org/cgi-bin/webring?ring=shamanring;list
73	72	O	http://dir.webring.com/rw
74	72	O	http://I.webring.com/wrman?ring=shamanring&addsite
75	NULL	O	http://proforma.real.com/real/player/player.html?src=001222realhome_1_010110r_pchoice_h1&dc=119118117
76	75	O	http://forms.real.com/rnforms/promos/200208/realone_sn/index.html

Tabla F.1.5: In y Out Links del Requerimiento 3

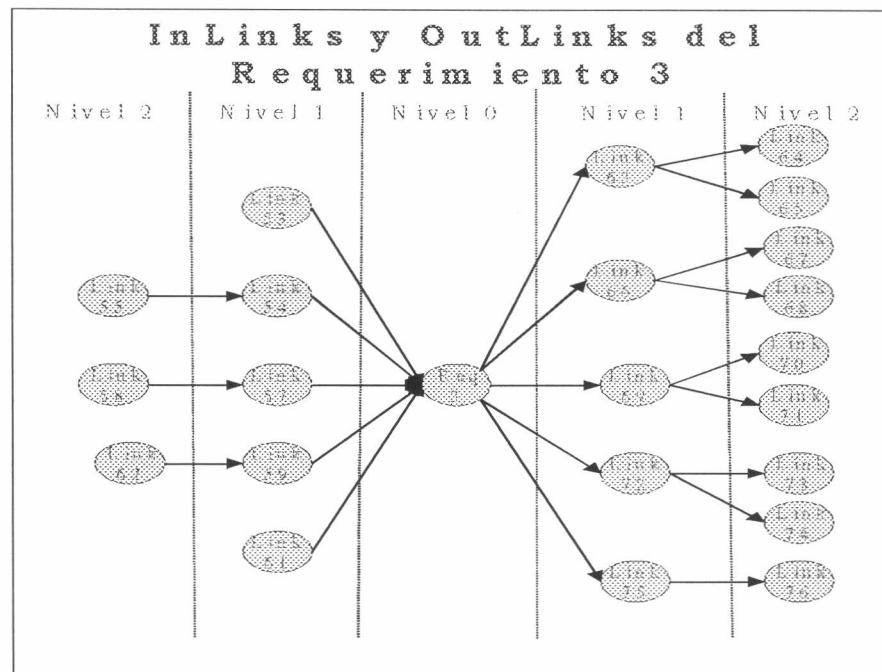


Figura F.1.3: In y Out Links del Requerimiento 3

En algunos casos puede suceder que no se alcance el total de in y out links deseados para un determinado requerimiento. Analicemos, por ejemplo, el requerimiento 2.

En la Figura F.1.2 puede observarse que tanto para el link 27 <http://www.nemeton.demon.co.uk/Links.html> como para el link 31 <http://www.datastatisticsonline.com/lynx.htm>, el sistema no obtuvo ningún IN link. Esto se debe a que *Altavista* no encontró ninguna página que apunte a dichos links. En el caso del link 32, solamente se encontró un IN link.

En el caso del link 38 el sistema no obtuvo ningún OUT link. Como puede observarse en el código html correspondiente al link (Figura F.1.4), no existen links en dicha página.


```

<HTML>
<HEAD>
<META http-equiv='Content-Type' content='text/html; charset=iso-8859-1'>
<META name='Author' content='Andrew Garman' >
<META name='Date' content='09/03/01' >
<TITLE>New HTML Document</TITLE>
<SCRIPT LANGUAGE='Javascript'>
<!--hide from non-script browsers
// convert all characters to lowercase to simplify testing
var agt=navigator.userAgent.toLowerCase();
// *** BROWSER VERSION ***
// Note: On IE5, these return 4, so use is_ie5up to detect IE5.
var is_major = parseInt(navigator.appVersion);
var is_minor = parseFloat(navigator.appVersion);
// Note: Opera and WebTV spoof Navigator. We do strict client detection.
// If you want to allow spoofing, take out the tests for opera and webtv.
var is_nav = ((agt.indexOf('mozilla')!=-1) && (agt.indexOf('spoofer')!=-1) && (agt.indexOf('compatible') == -1) &&
(agt.indexOf('opera')!=-1) && (agt.indexOf('webtv')!=-1)));
var is_nav2 = (is_nav && (is_major == 2));
var is_nav3 = (is_nav && (is_major == 3));
var is_nav4 = (is_nav && (is_major == 4));
var is_nav3up = (is_nav && (is_major >= 3));
var is_nav4up = (is_nav && (is_major >= 4));
var is_navonly = (is_nav && ((agt.indexOf(";nav") != -1) || (agt.indexOf("; nav") != -1)));
var is_nav5 = (is_nav && (is_major == 5));
var is_nav5up = (is_nav && (is_major >= 5));
var is_ie = (agt.indexOf("msie") != -1);
var is_ie3 = (is_ie && (is_major < 4));
var is_ie4 = (is_ie && (is_major == 4) && (agt.indexOf("msie 5.0")!=-1) );
var is_ie4up = (is_ie && (is_major >= 4));
var is_ie5 = (is_ie && (is_major == 4) && (agt.indexOf("msie 5.0")!=-1) );
var is_ie5up = (is_ie && !is_ie3 && !is_ie4);
// end hiding -->
</SCRIPT>
<SCRIPT LANGUAGE='Javascript'>
<!--hide from non-script browsers
var ref= new String(document.location)
ref= ref.substring(0,20)
ref=ref.toLowerCase()
if (ref== 'http://www.grafxreso') {
    if(is_nav4up){ document.location="grafx/index.htm";}
    else if(is_ie4up){ window.location.href="grafx/index.htm";} }
else {
    if(is_nav4up){ document.location="bcoind.htm";}
    else if(is_ie4up){ window.location.href="bcoind.htm";} }
// end hiding -->
</SCRIPT>
</HEAD>
<BODY BGCOLOR=#FFFFFF ></BODY>
</HTML>

```

Figura F.1.4: Código Html del link 38

F.1.3 SWMiner con Normalización Morfológica y con Clustering

F.1.3.1 Itemsets Frecuentes

Las Tablas F.1.6, F.1.7 y F.1.8 muestran los itemsets frecuentes obtenidos de cada uno de los requerimientos y su vecindad.

Id Itemset	Soporte	Item
11	0.30769231915473938	COPY
399	0.30769231915473938	BEHAVIOR
		GIRAFFE
		ANIMAL
		MAMMAL

460	0.30769231915473938	AFRICA
		PHOTOGRAPHY
		BEHAVIOR
		GIRAFFE
		ANIMAL
		MAMMAL
		BIRD
461	0.30769231915473938	AFRICAN
		PHOTOGRAPHY
		GIRAFFE
		ANIMAL
		MAMMAL
		GIRAFFA
		CAMELOPARDALIS
462	0.30761915473938	AFRICAN
		PHOTOGRAPHY
		GIRAFFE
		ANIMAL
		MAMMAL
		AFRICAN
		WILDLIFE
463	0.30769231915473938	IMAGE
		PHOTOGRAPHY
		GIRAFFE
		ANIMAL
		MAMMAL
		AFRICAN
		PHOTO
464	0.30769231915473938	IMAGE
		PHOTOGRAPHY
		GIRAFFE
		ANIMAL
		MAMMAL
		AFRICAN
		IMAGE
		SPOOK

Tabla F.1.6: : Itemsets Frecuentes del requerimiento 1

Id Itemset	Soporte	Item
466	0.2916666567325592	COUGAR
474	0.3333333432674408	YEAR
475	0.375	BOBCAT
		LYNX
476	0.2916666567325592	JAGUAR
		LEOPARD
477	0.3333333432674408	WORLD
		SPECIES
481	0.3333333432674408	CAT
		LEOPARD
484	0.2916666567325592	LEOPARD

486	0.2916666567325592	LYNX
		SPECIES
		CAT
487	0.2916666567325592	LYNX
		CAT
		CONSERVATION
		LYNX

Tabla F.1.7: Itemsets Frecuentes del requerimiento 2

Id Itemset	Soporte	Item
490	0.30434781312942505	INTERNET
491	0.26086956262588501	COUNTRY
492	0.26086956262588501	SIGN
493	0.34782609343528748	SPIRIT
494	0.26086956262588501	ANIMAL
		WORLD

Tabla F.1.8: Itemsets Frecuentes del requerimiento 3

F.1.3.2 Clustering e Identificación de Tópicos

Las Figuras F.1.5, F.1.6 y F.1.7 muestran el árbol de cómo el algoritmo de clustering fue generando los clusters para cada requerimiento con las respectivas medidas de similaridad calculadas en cada agrupación. En color se muestran los clusters finales obtenidos para cada requerimiento.

Para el requerimiento 1 el algoritmo agrupó todos los itemsets en un único cluster y los respectivos tópicos se muestran en la Tabla F.1.9.

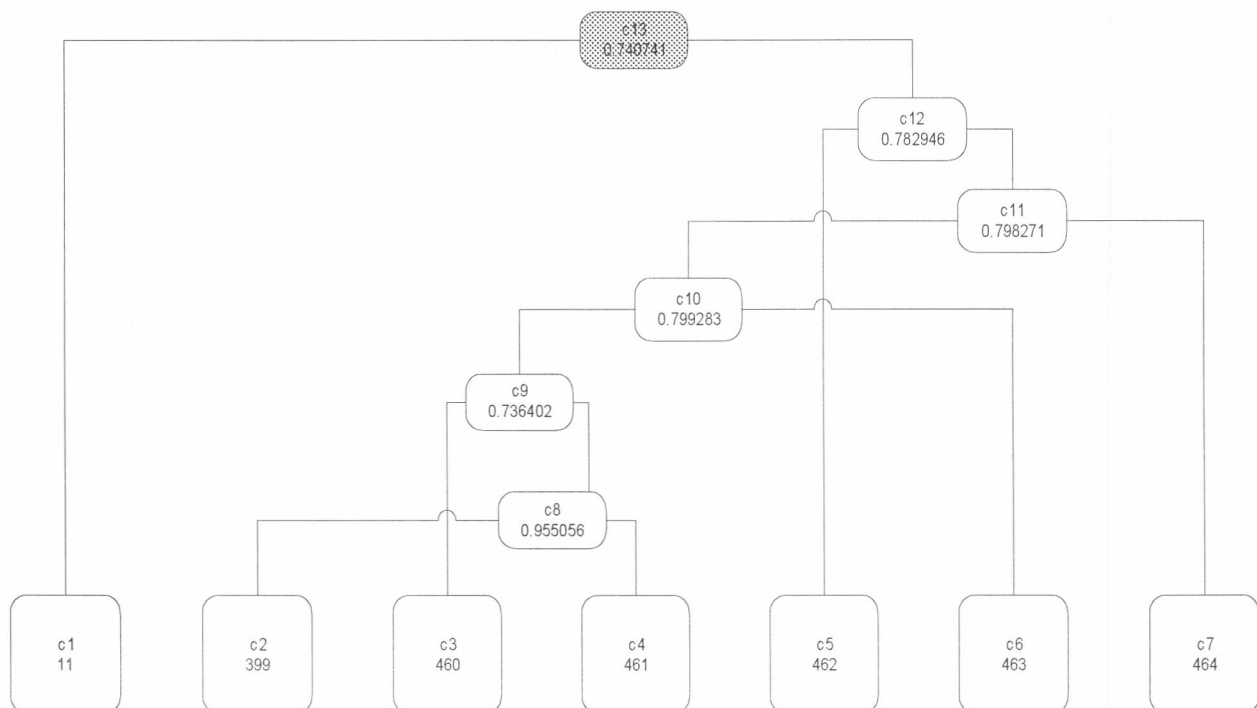


Figura F.1.5: Árbol de clusters del requerimiento 1

Id Cluster	Id Itemset	Tópicos
c13	11, 399, 460, 461, 462, 463, 464	transcript, copy
		copy
		copy, written matter
		representation
		person, individual, someone, somebody, mortal, human, soul
		giraffe, camelopard, Giraffa camelopardalis

Tabla F.1.9: Tópicos de los clusters obtenidos del requerimiento 1

Para el requerimiento 2 el algoritmo agrupó también todos los itemsets frecuentes en un único cluster y los respectivos tópicos se muestran en la Tabla F.1.10.

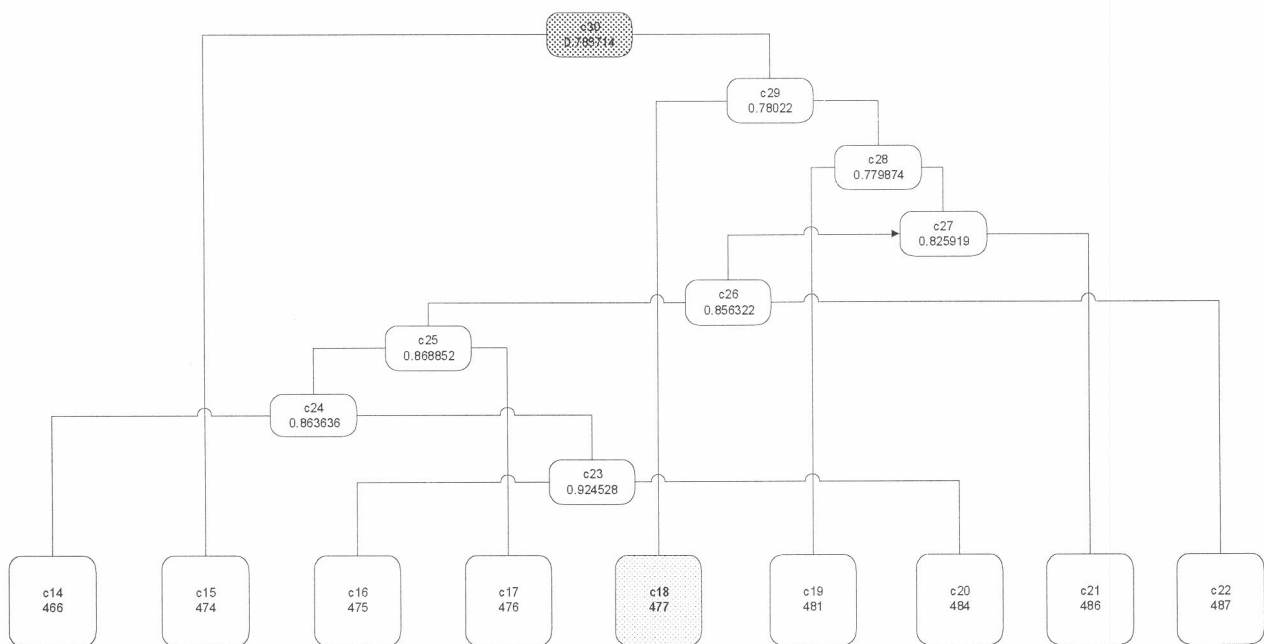


Figura F.1.6: Arbol de clusters del requerimiento 2

Id Cluster	Id Itemset	Tópicos
c30	466, 474, 475, 476, 477, 481, 484, 486, 487	year, twelvemonth, yr
		year
		class, year
		natural object
		feline, felid
		instrumentality, instrumentation
		big cat, cat
		cougar, puma, catamount, mountain lion, painter, panther, Felis concolor
		bobcat, bay lynx, Lynx rufus

Tabla F.1.10: Tópicos de los clusters obtenidos del requerimiento 2

Para el requerimiento 3 el algoritmo generó los clusters que se muestran en la Figura F.1.7 y los tópicos de cada cluster se muestran en la Tabla F.1.11.

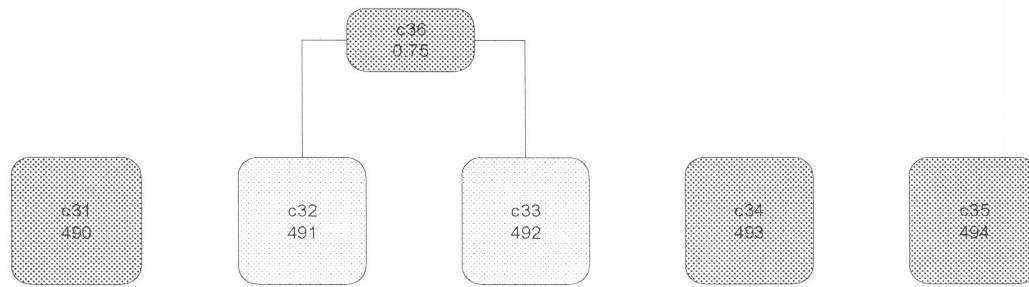


Figura F.1.7: Árbol de clusters del requerimiento 3

Id Cluster	Id Itemset	Tópicos
c31	490	internet, cyberspace
c34	493	spirit
		spirit, tone, feel, feeling, flavor, look, smell
		emotional state, spirit
		intent, purport, spirit
		liveliness, life, spirit, sprightliness
		heart, spirit
c35	494	--
c36	491, 492	country, state, land, nation
		state, nation, country, land, commonwealth, res publica, body politic
		nation, nationality, land, country, a people
		area, country
		country, rural area
		sign, mark
		sign
		signal, signaling, sign
		signboard, sign
		sign of the zodiac, sign, mansion, house, planetary house
		polarity, sign
		augury, sign

Tabla F.1.11: Tópicos de los clusters obtenidos del requerimiento 3

En la última iteración, el algoritmo de clustering agrupó los clusters obtenidos para cada requerimiento como se muestra en la Figura F.1.8. En la Tabla F.1.12 se muestran los tópicos de cada uno de los clusters finales obtenidos.

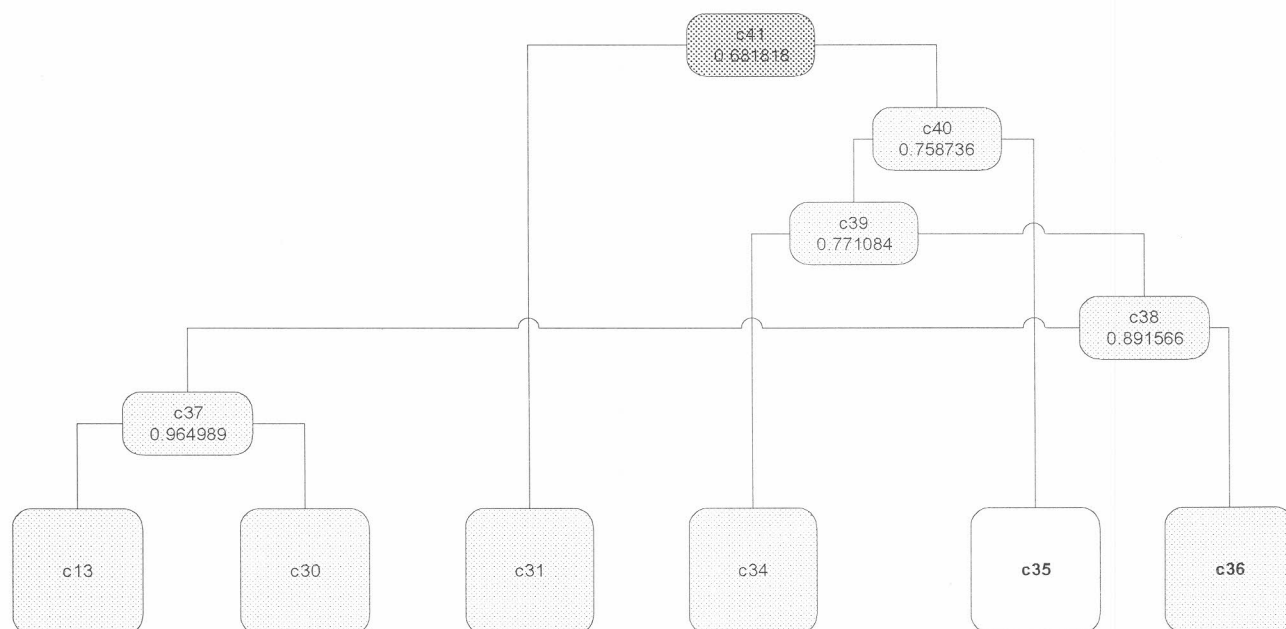


Figura F.1.8: Arbol de clusters final

Id Cluster	Id Itemset	Temas
c41	11, 399, 460, 461, 462, 463, 464, 466, 474, 475, 476, 477, 481, 484, 486, 487, 490, 491, 492, 493, 494	person, individual, someone, somebody, mortal, human, soul natural object feline, felid cougar, puma, catamount, mountain lion, painter, panther, Felis concolor bobcat, bay lynx, Lynx rufus big cat, cat giraffe, camelopard, Giraffa camelopardalis Copy instrumentality, instrumentation Representation signboard, sign Spirit heart, spirit liveliness, life, spirit, sprightliness copy, written matter transcript, copy intent, purport, spirit sign, mark signal, signaling, sign Sign augury, sign nation, nationality, land, country, a people state, nation, country, land, commonwealth, res publica, body politic class, year area, country country, state, land, nation country, rural area

	sign of the zodiac, sign, mansion, house, planetary house
	internet, cyberspace
	polarity, sign
	emotional state, spirit
	spirit, tone, feel, feeling, flavor, look, smell
	Year
	year, twelvemonth, yr

Tabla F.1.12: Tópicos de los clusters finales

Analizando paso a paso los árboles de clusters generados por el algoritmo de clustering a partir de los itemsets frecuentes, podemos aseverar que el mismo se comporta de la manera esperada, agrupando en primer lugar aquellos clusters más similares según la función de similaridad definida. Sin embargo, podemos observar que se obtuvieron clusters poco cohesivos semánticamente hablando; es decir, se agruparon en un mismo cluster itemsets de contenido semántico diferente. Esto se puede mejorar aumentando la cota de *Mínima Similaridad entre Clusters*. A continuación mostramos los resultados obtenidos al trabajar con la medida de similaridad mínima entre clusters en 0.8.

En la tabla F.1.13, F.1.14 y F.1.15 se muestran los clusters obtenidos para cada uno de los requerimientos y los respectivos tópicos.

Id Cluster	Id Itemset	Tópicos
1	11	copy copy, written matter transcript, copy
2	460	person, individual, someone, somebody, mortal, human, soul
3	462	representation
4	463	representation
5	464	representation
6	399, 461	giraffe, camelopard, Giraffa camelopardalis

Tabla F.1.13: Clusters del requerimiento 1

Id Cluster	Id Itemset	Tópicos
7	474	class, year year year, twelvemonth, yr
8	477	natural object
9	481	feline, felid
10	486, 487, 476, 466, 475, 484	cougar, puma, catamount, mountain lion, painter, panther, Felis concolor bobcat, bay lynx, Lynx rufus big cat, cat instrumentality, instrumentation

Tabla F.1.14: Clusters del requerimiento 2

Id Cluster	Id Itemset	Tópicos
11	490	internet, cyberspace
12	491	nation, nationality, land, country, a people state, nation, country, land, commonwealth, res publica, body politic

		area, country country, state, land, nation country, rural area
13	492	signboard, sign sign, mark signl, signaling, sign sign augury, sign sign of the zodiac, sign, mansion, house, planetary house polarity, sign
14	493	spirit heart, spirit liveliness, life, spirit, sprightliness intent, purport, spirit emotional state, spirit spirit, tone, feel, feeling, flavor, look, smell
15	494	--

Tabla F.1.15: Clusters del requerimiento 3

En la Tabla F.1.16 se muestran los clusters finales obtenidos y sus respectivos tópicos luego de la iteración final del algoritmo de Clustering sobre los clusters de todos los requerimientos.

Id Cluster	Id Itemset	Tópicos
16	11	copy copy, written matter transcript, copy
17	474	class, year year year, twelvemonth, yr
18	477	natural object
19	481	feline, felid
20	490	internet, cyberspace
21	491	nation, nationality, land, country, a people state, nation, country, land, commonwealth, res publica, body politic area, country country, state, land, nation country, rural area
22	492	signboard, sign sign, mark signl, signaling, sign sign augury, sign sign of the zodiac, sign, mansion, house, planetary house polarity, sign
23	493	spirit heart, spirit liveliness, life, spirit, sprightliness intent, purport, spirit emotional state, spirit spirit, tone, feel, feeling, flavor, look, smell
24	494	--
25	399, 460, 461, 462, 463, 464, 466, 475, 476, 484, 486, 487	person, individual, someone, somebody, mortal, human, soul cougar, puma, catamount, mountain lion, painter, panther, Felis concolor bobcat, bay lynx, Lynx rufus big cat, cat giraffe, camelopard, Giraffa camelopardalis instrumentality, instrumentation representation

Tabla F.1.16: Clusters Finales

Podemos confirmar entonces que los clusters obtenidos recientemente son mucho más cohesivos semánticamente que los obtenidos con la medida de *Mínima Similitud entre clusters* definida en 0.70.

F.1.4 SWMiner con Normalización Morfológica y sin Clustering

Las Tablas F.1.17, F.1.18 y F.1.19 describen los tópicos asignados a cada itemset de cada uno de los requerimientos del log.

Id Itemset	Item	Tópico
11	COPY	copy copy, written matter transcript, copy
399	BEHAVIOR GIRAFFE ANIMAL MAMMAL AFRICA	giraffe, camelopard, Giraffa camelopardalis
460	PHOTOGRAPHY BEHAVIOR GIRAFFE ANIMAL MAMMAL BIRD AFRICAN	person, individual, someone, somebody, mortal, human, soul
461	PHOTOGRAPHY GIRAFFE ANIMAL MAMMAL GIRAFFA CAMELOPARDALIS AFRICAN	giraffe, camelopard, Giraffa camelopardalis
462	PHOTOGRAPHY GIRAFFE ANIMAL MAMMAL AFRICAN WILDLIFE IMAGE	representation
463	PHOTOGRAPHY GIRAFFE ANIMAL MAMMAL AFRICAN PHOTO IMAGE	representation
464	PHOTOGRAPHY GIRAFFE ANIMAL MAMMAL AFRICAN IMAGE SPOOK	representation

Tabla F.1.17: Tópicos de los itemsets frecuentes del requerimiento 1

Id Itemset	Item	Tópico
466	COUGAR	cougar, puma, catamount, mountain lion, painter, panther, Felis concolor
474	YEAR	class, year year year, twelvemonth, yr

475	BOBCAT LYNX	bobcat, bay lynx, Lynx rufus
476	JAGUAR LEOPARD	big cat, cat
477	WORLD SPECIES	natural object
481	CAT LEOPARD	feline, felid
484	LEOPARD LYNX	--
486	SPECIES CAT LYNX	instrumentality, instrumentation
487	CAT CONSERVATION LYNX	instrumentality, instrumentation

Tabla F.1.18: Tópicos de los itemsets frecuentes del requerimiento 2

Id Itemset	Item	Item
490	INTERNET	internet, cyberspace
91	COUNTRY	nation, nationality, land, country, a people state, nation, country, land, commonwealth, res publica, body politic area, country country, state, land, nation country, rural area
492	SIGN	signboard, sign sign, mark signal, signaling, sign sign augury, sign sign of the zodiac, sign, mansion, house, planetary house polarity, sign
493	SPIRIT	spirit heart, spirit liveliness, life, spirit, sprightliness intent, purport, spirit emotional state, spirit spirit, tone, feel, feeling, flavor, look, smell
494	ANIMAL WORLD	--

Tabla F.1.19: Tópicos de los itemsets frecuentes del requerimiento 3

F.1.5 SWMiner sin Normalización Morfológica y con Clustering

La Tabla F.1.20 describe los itemsets frecuentes y clusters obtenidos para cada requerimiento del log junto con los tópicos asignados a cada cluster. La Tabla F.1.21 muestra los clusters finales obtenidos resaltando en negrita los tópicos de mayor bondad de cada cluster.

Requerimiento	Cluster	Itemsets	Items	Tópicos
1	1	10	COPY	transcript, copy

		307	PHOTOGRAPHY AFRICAN PHOTO IMAGES IMAGE SPOOK	copy copy, written matter representation mammal
		305	PHOTOGRAPHY BEHAVIOR AFRICA AFRICAN IMAGES SPOOK	
		314	BEHAVIOR AFRICA AFRICAN IMAGES MAGE SPOOK	
		306	PHOTOGRAPHY GIRAFFE AFRICAN WILDLIFE IMAGES SPOOK	
		316	GIRAFFE AFRICAN WILDLIFE IMAGES IMAGE SPOOK	
		317	BEHAVIOR GIRAFFE AFRICA AFRICAN WILDLIFE IMAGES SPOOK	
		119	GIRAFFE AFRICA GIRAFFES	
		8	MAMMAL	

		104	BEHAVIOR AFRICA GIRAFFES	
		261	PHOTOGRAPHY GIRAFFE GIRAFFA CAMELOPARDALIS AFRICAN	
2	2	336	CONSERVATION	conservation, preservetion
		325	FELINE	conservation, conservancy, conserving
		324	SPECIES	conservation, conservation law
		553	CAT	feline, felid
			CLOUDED LEOPARD	coinage, mintage, specie, metal money
		321	WORLD	world, human race, humanity, humankind, human beings, humans, mankind, man
		471	COUGAR CAT LINX	universe, existence, nature, creation, world, cosmos, macrocosm world, reality world, domain Earth, world, globe Worldly concern, earthly concern, world, earth world populace, public, world wildcat
3	3	4937	ANIMAL	animal, animate being, beast, brute, creature, fauna
	4	4938	INTERNET	internet, cyberspace
	5	4942	CREATE	--
	6	321	WORLD	world, human race,

		4939	WORLD	humanity, humankind, human beings, humans, mankind, man universe, existence, nature, creation, world, cosmos, macrocosm world, reality world, domain Earth, world, globe worldly concern, earthly concern, world, earth world populace, public, world
	7	4940	HISTORY	History
		4941	SIGN	history, account, chronicle, story sign, mark sign signal, signaling, sign signboard, sign sign of the zodiac, sign, mansion, house, planetary house popularity, sign augury, sign

Tabla F.1.20: Clusters por requerimiento de *SWMiner* sin Normalización y con Clustering

Cluster	Itemsets	Items	Tópicos
1	4942	CREATE	--
2	8	MAMMAL	animal, animate being, beast, brute, creature, fauna
	10	COPY	conservation, conservancy, conserving
	104	BEHAVIOR	mammal
		AFRICA	feline, felid
		GIRAFFES	wildcat
	119	GIRAFFE	copy
		AFRICA	representation
		GIRAFFES	signboard, sign
	261	PHOTOGRAPHY	worldly concern, earthly concern, world, earth
		GIRAFFE	world, reality
		GIRAFFA	conservation, conservation law
		CAMELOPARDALIS	history
		AFRICAN	copy, written matter
			transcript, copy

	305	PHOTOGRAPHY BEHAVIOR AFRICA AFRICAN IMAGES SPOOK	history, account, chronicle, story sign, mark signal, signaling, sign sign augury, sign conservation, preservation
	306	PHOTOGRAPHY GIRAFFE AFRICAN WILDLIFE IMAGES SPOOK	world, human race, humanity, humankind, human beings, humans, mankind, man world, domain populace, public, world sign of the zodiac, sign, mansion, house, planetary house world
	307	PHOTOGRAPHY AFRICAN PHOTO IMAGES IMAGE SPOOK	universe, existence, nature, creation, world, cosmos, macrocosm earth, world, globe internet, cyberspace coinage, mintage, specie, metal money polarity, sign
	314	BEHAVIOR AFRICA AFRICAN IMAGES IMAGE SPOOK	
	316	GIRAFFE AFRICAN WILDLIFE IMAGES IMAGE SPOOK	
	317	BEHAVIOR GIRAFFE AFRICA AFRICAN WILDLIFE IMAGES SPOOK	
	321	WORLD	
	324	SPECIES	
	325	FELINE	

	336	CONSERVATION	
	553	CAT CLOUDED LEOPARD	
	4937	ANIMAL	
	4938	INTERNET	
	4939	WORLD	
	4940	HISTORY	
	4941	SIGN	

Tabla F.1.21: Clusters finales de *SWMiner* sin Normalización y con Clustering

F.1.6 SWMiner sin Normalización Morfológica y sin Clustering

La Tabla F.1.22 describe los tópicos obtenidos para cada uno de los requerimientos con sus respectivos itemsets e ítems al ejecutar *SWMiner sin Normalización Morfológica ni Clustering* y resaltamos en negrita los tópicos de mayor bondad (categorías).

Requerimiento	Itemsets	Items	Tópicos
1	10	COPY	transcript, copy copy copy, written matter
	307	PHOTOGRAPHY AFRICAN PHOTO IMAGES IMAGE SPOOK	representation
	305	PHOTOGRAPHY BEHAVIOR AFRICA AFRICAN IMAGES SPOOK	representation
	314	BEHAVIOR AFRICA AFRICAN IMAGES MAGE SPOOK	representation

	306	PHOTOGRAPHY GIRAFFE AFRICAN WILDLIFE IMAGES SPOOK	representation
	316	GIRAFFE AFRICAN WILDLIFE IMAGES IMAGE SPOOK	representation
	317	BEHAVIOR GIRAFFE AFRICA AFRICAN WILDLIFE IMAGES SPOOK	representation
	119	GIRAFFE AFRICA GIRAFFES	--
	8	MAMMAL	mammal
	104	BEHAVIOR AFRICA GIRAFFES	--
	261	PHOTOGRAPHY GIRAFFE GIRAFFA CAMELOPARDALIS AFRICAN	--

2	321	WORLD	worldly concern, earthly concern, world, earth world, reality world, human race, humanity, humankind, human beings, humans, mankind, man world, domain populace, public, world world universe, existence, nature, creation, world, cosmos, macrocosm Earth, world, globe
	324	SPECIES	coinage, mintage, specie, metal money
	325	FELINE	feline, felid
	336	CONSERVATION	conservation, preservation conservation, conservancy, conserving conservation, conservation law
	471	COUGAR CAT LYNX	wildcat
	553	CAT CLOUDED LEOPARD	
3	321	WORLD	worldly concern, earthly concern, world, earth world, reality world, human race, humanity, humankind, human beings, humans, mankind, man world, domain populace, public, world world universe, existence, nature, creation, world, cosmos, macrocosm Earth, world, globe
	4937	ANIMAL	animal, animate being, beast, brute, creature, fauna
	4938	INTERNET	internet, cyberspace

	4939	WORLD	world, human race, humanity, humankind, human beings, humans, mankind, man universe, existence, nature, creation, world, cosmos, macrocosm world, reality world, domain Earth, world, globe wordly concern, earthly concern, world, earth world populace, public, world
	4940	HISTORY	history history, account, chronicle, story
	4941	SIGN	sign, mark sign signal, signaling, sign signboard, sign sign of the zodiac, sign, mansion, house, planetary house popularity, sign augury, sign
	4942	CREATE	--

Tabla F.1.22: Tópicos de *SWMiner* sin Normalización y con Clustering

F.2 Experimentos con Log General

F.2.1 Parámetros de SWMiner

En la Tabla F.2.1 se detalla el valor de los parámetros y/o variables del sistema utilizados en este experimento.

Parámetro	Valor
Cantidad de In Links por Requerimiento	5
Cantidad de Out Links por Requerimiento	5
Cantidad de In Links por Link	2
Cantidad de Out Links por Link	2
Cantidad de Niveles	2
Cantidad de Palabras por Página	50
Soporte Association Rules	0.3
Similaridad Mínima entre Clusters	0.7

WordNet – Ratio	0.68
WordNet – Profundidad	2

Tabla F.2.1: Parámetros del SWMiner

F.2.2

Requerimientos y su Vecindad

Requerimientos del Log

Id	Requerimiento
1	http://java.sun.com
2	http://www.eff.org
3	http://www.dw-institute.com

Tabla F.2.2: Requerimientos del Log

Vecindad de los requerimientos

Las Tablas F.2.3, F.2.4 y F.2.5 muestran los links y out links obtenidos para cada uno de los requerimientos anteriores.

Id Link	Id Link Padre	Tipo	Url
5		IN	http://www.sun.com/
6	5	IN	http://www.seti-inst.edu/
8	5	IN	http://foundation.gnome.org/
10		IN	http://www.helixdemos.com/
11	10	IN	http://java.sun.com/javaone/
12	10	IN	http://servlet.java.sun.com/javaone/home/0-sf2002.jsp
14		IN	http://chat-forum.com/
16	14	IN	http://www.thestudentzone.com/articles/chat.html
17	14	IN	http://www.joblo.com/chat.htm
21		IN	http://www.endymion.com/
22	21	IN	http://www.its.mx/
24	21	IN	http://www.xtec.es:8800/usuarios/correuweb
26		IN	http://www.law.cornell.edu/
28	26	IN	http://www.law.cornell.edu/lii.html
30	26	IN	http://www.informatik.uni-trier.de/%7Eley/db/journals/index.html
31		OUT	http://search.java.sun.com/search/java/advanced.jsp
32	31	OUT	http://developer.java.sun.com/developer/earlyAccess/
33	31	OUT	http://wireless.java.sun.com/
34		OUT	http://www.jcp.org/
35	34	OUT	http://www.sun.com/share/text/termsfuse.html

36	34	OUT	http://www.sun.com/privacy/
37		OUT	http://www201.ikiosk.com/cgi-shl/index.cgi?shop=33293&date=2002%3A04%3A26%3A10%3A29%3A26
38	37	OUT	http://www.java.sun.com
39	37	OUT	http://java.sun.com/docs/books/
40		OUT	http://java.sun.com/
41	40	OUT	http://servlet.java.sun.com/logRedirect/frontpage-nav/http://java.sun.com/products/
42	40	OUT	http://servlet.java.sun.com/logRedirect/frontpage-nav/http://java.sun.com/j2ee/
43		OUT	http://servlet.java.sun.com/logRedirect/frontpage-nav/http://java.sun.com/j2se/
44	43	OUT	http://wwws.sun.com/software/java2
45	43	OUT	http://www.onjava.com/pub/a/onjava/2002/03/06/topten.html

Tabla F.2.3: In y Out Links del Requerimiento1

Los links 4, 7, 9, 13, 18, 20, 23, 25, 27 y 29 no aparecen en la Tabla F.2.3 porque SWMiner obtuvo documentos html de 0 Kb al realizar el ftp de cada uno de ellos y los links 15 y 19 no aparecen porque no se encontraron las urls (*error http 404*).

Id Link	Id Link Padre	Tipo	Url
46		IN	http://www.gnu.org/
47	46	IN	http://www.mandrakebizcases.com/
53	46	IN	http://www.freedevelopers.net/
56		IN	http://bpdg.blogs.eff.org/
57	56	IN	http://www.eff.org/br/
60	56	IN	http://www.epic.org/
62		IN	http://neuro-www.mgh.harvard.edu/
64	62	IN	http://neuro-www.mgh.harvard.edu:16080/forum/
66	62	IN	http://www.chas.uchicago.edu/healthlinks/index.html
68		IN	http://www.bluesnews.com/
69	68	IN	http://www.shacknews.com/
72	68	IN	http://www.bluesnews.com/plans/1/
74		IN	http://www.stallman.org/
76	74	IN	http://icf.possenet.org/project/www/glossary.html
78	74	IN	http://www.teledyn.com/
79		OUT	http://action.eff.org/subscribe/
80	79	OUT	http://blogs.eff.org/
81	79	OUT	http://www.eff.org/
82		OUT	http://www.archive.org/bookmobile
83	82	OUT	http://www.hp.com/
84	82	OUT	http://www.computerhistory.org/

85		OUT	http://www.gilc.org/
86	85	OUT	http://www.gilc.nl/
87	85	OUT	http://www.effi.org/
88		OUT	http://www.ifea.net/
89	88	OUT	http://www.supremecourtus.gov
90	88	OUT	http://www.consumerreports.org/Special/ConsumerInterest/Reports/0103fil0.html
91		OUT	http://www.dfc.org/
92	91	OUT	http://www.dfc.org/
93	91	OUT	http://www.4CITE.org

Tabla F.2.4: In y Out Links del Requerimiento 2

En la Tabla F.2.4 están ausentes los links 48, 50, 52, 54, 55, 58, 59, 61, 63, 65, 67, 70, 71, 73, 75 y 77 porque *SWMiner* obtuvo documentos html de 0 kb para cada uno de ellos y los links 49 y 51 están ausentes porque al realizar el ftp para cada uno de ellos se obtuvo un documento html indicando un error http (error *http 500- Server Error*).

Id Link	Id Link Padre	Tipo	Url
94		IN	http://www.101com.com/
96	94	IN	http://www.101com.com/privacy.asp
97	94	IN	http://www.e-gov.com/
100		IN	http://www.sqlmag.com/
101	100	IN	http://www.2000exams.com/
103	100	IN	http://www.ntsecurity.net/
105		IN	http://www.fcw.com/events.asp
107	105	IN	http://www.iaconline.org/budget_plan.html
112	105	IN	http://www.idoc-inc.com/OutsideStories/VAIT_b.htm
113		IN	http://www.informatik.uni-trier.de/%7Ehoff/links/webquerying.html
114	113	IN	http://www.eecs.wsu.edu/%7Ecodyreson/teaching/advancedDatabase/011/resources.htm
115	113	IN	http://www.cs.uregina.ca/%7Exuifei/index.html
117		IN	http://dw.ittoolbox.com/
118	117	IN	http://windows.ittoolbox.com/
120	117	IN	http://scm.ittoolbox.com/
123		OUT	http://subscribe.101com.com/tdwi/whatworks14/
124		OUT	http://lists.101com.com/NLS/pages/main.asp?NL=tdwi_whatworks
125	124	OUT	http://dw-institute.com
126	124	OUT	http://lists.101com.com/NLS/pages/unsub_login.asp?NL=tdwi_whatworks
127		OUT	http://www.inquisiteasp.com/surveys/W9G7BR
129		OUT	http://www.adtmag.com/index.asp
130	129	OUT	http://www.xmlconference.com/santaclara/

148		OUT	http://www.dw-institute.com/education/main.asp?pageName=on_track
150	148	OUT	http://education/conferences/neworleans2003/sessions2.asp?session_code=25
151	148	OUT	http://education/conferences/neworleans2003/sessions2.asp?session_code=16

Tabla F.2.5: In y Out Links del Requerimiento 3

Por último en la Tabla F.2.5 no aparecen los links 95, 98, 99, 102, 104, 106, 108 a 111, 116, y 119 porque *SWMiner* obtuvo documentos html de 0 Kb, el link 121 porque contiene frames (fue reemplazado por otro link), el 122, 128, 131 a 147 y 149 por error *http 204 (No Content)*.

F.2.3 SWMiner con Normalización Morfológica y con Clustering

F.2.3.1 Itemsets Frecuentes

A continuación presentamos los itemsets frecuentes obtenidos al aplicar el algoritmo de *Association Rules* [GON96, ASI93, AS94] sobre todas los sustantivos que representan a cada uno de las páginas involucradas en este análisis.

Las Tablas F.2.6, F.2.7 y F.2.8 describen los itemsets frecuentes obtenidos para cada uno de los requerimientos.

Id Itemset	Soporte	Item
1	0.35483869910240173	TM
3	0.35483869910240173	DOCUMENTATION
5	0.35483869910240173	TECHNOLOGY DEVELOPER

Tabla F.2.6: Itemsets frecuentes del Requerimiento 1

Id Itemset	Soporte	Item
6	0.32258063554763794	PRIVACY
7	0.32258063554763794	PUBLIC
8	0.35483869910240173	INTERNET
9	0.38709676265716553	ELECTRONIC
10	0.35483869910240173	FREEDOM

Tabla F.2.7 : Itemsets Frecuentes del Requerimiento 2

Id Itemset	Soporte	Item
12	0.3333333432674408	INDUSTRY
13	0.3333333432674408	COMMUNITY
14	0.3333333432674408	APPLICATION
16	0.3333333432674408	DATABASE
17	0.3333333432674408	EMAIL
18	0.3333333432674408	TECHNOLOGY CONFERENCE

Tabla F.2.8: Itemsets Frecuentes del Requerimiento 3

F.2.3.2 Clustering e Identificación de Tópicos

En la Tabla F.2.9 detallamos los clusters obtenidos al aplicar el algoritmo de *Clustering* sobre los itemsets correspondientes a todos los requerimientos y los tópicos correspondientes a cada uno de los clusters.

Requerimiento	Cluster	Itemsets	Tópicos
1	1	1	thulium, Tm, atomic number 69
	2	3	documentation, support , software documentation, certification, corroboration
	3	5	
2	4	6	privacy, privateness, seclusion, secrecy, concealment
	5	7	populace, public, world
	6	8	internet, cyberspace
	7	9	
	8	10	exemption, freedom
3	9	16	Database
	10	17	electronic mail, e-mail, email
	11	12, 13, 14, 18	application, diligence, oating, covering, practical application, lotion, application program, applications programme, profession, community, residential district, residential area, Indus, Indus River, community of interests, diligence, coating, covering,

Tabla F.2.9: Clusters y Tópicos

Una vez obtenidos todos los clusters correspondientes a cada requerimiento del usuario, el sistema aplica una vez más el algoritmo de *Clustering* sobre todos estos con el fin de agrupar los clusters con tópicos similares y reducir así la cantidad de tópicos que le asigna al usuario. En este caso particular se obtuvieron los mismos clusters (ver Tabla F.2.9).

La Tabla F.2.10 describe los tópicos de mayor bondad para cada uno de los clusters.

Requerimiento	Cluster	Tópicos de Mayor Bondad
1	1	thulium, Tm, atomic number 69
	2	documentation, support
	3	--
2	4	privacy, privateness, secrecy, concealment
	5	Public
	6	internet, cyberspace
	7	--
	8	exemption, freedom
3	9	Database
	10	electronic mail, e-mail, email
	11	application, diligence

Tabla F.2.10: Tópicos de Mayor Bondad

Observando los clusters obtenidos en la Tabla F.2.9 notamos que los itemsets 12,13, 14 y 18 del requerimiento 3 deberían haberse agrupado con el itemset 5 del requerimiento 1.

La razón por la cual no se agruparon se debe a los valores de similaridad obtenidos al aplicar el algoritmo de *Clustering*, los cuales no superan la cota (*Similaridad Mínima entre Clusters*), por lo tanto decidimos disminuir esta cota para lograr obtener mejores resultados.

También notamos que *SWMiner* no obtuvo tópicos para algunos de los clusters, esto se debe a que no se obtuvieron tópicos en común para los ítems de los itemsets del clusters en la jerarquía semántica de *WordNet*.

La Tabla F.2.11 describe los clusters obtenidos al ejecutar *SWMiner* con *Similaridad Mínima entre Clusters* del 50%.

Cluster	Itemsets	Tópicos
1	5	
2	6	privacy, privateness, seclusion, secrecy, concealment
3	8	internet, cyberspace
4	9	
5	10	exemption, freedom
6	17	electronic mail, e-mail, email
7	1, 3, 7, 12, 13, 14, 16 y 18	documentation, support, application, diligence, coating, covering, practical application, lotion,, application program, applications programme, software documentation, documentation, database, certification, corroboration, profession, community, populace, public, world, community, residential district, residential area, Indus, Indus River, community of interests, thulium, Tm, atomic number 69

Tabla F.2.11: Clusters y Tópicos con Similaridad Mínima entre Cluster del 50%

La Tabla F.2.12 detalla los tópicos de mayor bondad correspondientes a cada cluster.

Cluster	Tópicos de Mayor Bondad (Categorías)
1	--
2	privacy, privateness, secrecy, concealment
3	internet, cyberspace
4	--
5	exemption, freedom
6	Electronic mail, e-mail, email
7	application, diligence

Tabla F.2.12: Tópicos de Mayor Bondad con Similaridad Mínima entre Cluster del 50%

Si comparamos los tópicos de mayor bondad de la Tabla F.2.10 con los de la Tabla F.2.12 notamos que son prácticamente los mismos, por lo tanto disminuir la cota de *Similaridad Mínima entre Clusters* no mejoró los resultados finales.

F.2.4 SWMiner con Normalización Morfológica y sin Clustering

Los itemsets frecuentes obtenidos para cada requerimiento son los mismos que se detallan en las Tablas F.2.6, F.2.7 y F.2.8.

La Tabla F.2.13 describe los tópicos obtenidos para cada uno de los itemsets frecuentes.

Requerimiento	Itemsets	Tópicos
1	1	thulium, Tm, atomic number 69
		documentation, support, software documentation
	3	documentation, certification, corroboration
	5	
2	6	privacy, privateness, seclusion, secrecy, concealment
	7	populace, public, world
	8	internet, cyberspace
	9	
	10	exemption, freedom
3	12	Indus, Indus River
	13	profession, community, residential district, residential area, community of interests
	14	application, diligence, covering, practical application, lotion
	16	database
	17	electronic mail, e-mail, email
	18	

Tabla F.2.13: Tópicos de los Itemsets Frecuentes

La Tabla F.2.14 describe los tópicos de mayor bondad correspondientes a cada uno de los itemsets frecuentes.

Requerimiento	Itemsets	Tópico de Mayor Bondad (Categorías)
1	1	thulium, Tm, atomic number 69
	3	documentation, support
	5	
2	6	privacy, privateness, secrecy, concealment
	7	Public
	8	internet, cyberspace
	9	
	10	exemption, freedom
3	12	Indus, Indus River
	13	residential district, residential area, community
	14	application, diligence
	16	Database
	17	electronic mail, e-mail, email
	18	

Tabla F.2.14: Tópicos de Mayor Bondad de los Itemsets Frecuentes

F.2.5 SWMiner sin Normalización Morfológica y con Clustering

F.2.5.1 Itemsets Frecuentes

Las Tablas F.2.15, F.2.16 y F.2.17 describen los itemsets frecuentes obtenidos para cada uno de los requerimientos.

Id Itemset	Soporte	Item
3	0,322580635547638	COMPANY
7	0,322580635547638	TECHNOLOGY
		ADVANCED
11	0,322580635547638	TERMS
		PRIVACY
		POLICY

Tabla F.2.15: Itemsets frecuentes del Requerimiento 1

Id Itemset	Soporte	Item
14	0,322580635547638	SOFTWARE
15	0,419354826211929	INTERNET
17	0,354838699102402	PUBLIC
18	0,322580635547638	PRIVACY
		ELECTRONIC

Tabla F.2.16: Itemsets frecuentes del Requerimiento 2

Id Itemset	Soporte	Item
20	0,333333343267441	INDUSTRY
22	0,407407402992249	CONTACT
23	0,333333343267441	CONTENT
25	0,333333343267441	NEWSLETTERS
		EMAIL

Tabla F.2.17: Itemsets frecuentes del Requerimiento 3

F.2.5.2 Clustering e Identificación de Tópicos

En la Tabla F.2.18 y F.2.19 detallamos los clusters obtenidos al aplicar el algoritmo de *Clustering* sobre los itemsets correspondientes a todos los requerimientos y los tópicos correspondientes a cada uno de los clusters.

Requerimiento	Cluster	Itemset	Tópicos
1	1	7	--
	2	3, 11	communication company company, troupe ship's company, company party, company caller, company company, comradeship, companionship, good fellowship, fellowship, society

2	3	14	software, software system
	4	15	internet, cyberspace
		17	public
	5		populace, public, world
	6	18	
3	7	20	Indus, Indus River
		22	contact contact, contacting. contact, tangency liaison, link, contact, inter-group communication contact, touch contact, impinging, striking contact, middleman
	8		
		23, 25	subject, content, depicted object content, cognitive content, mental object message, content, subject matter, substance content capacity, content
	9		contentedness, content

Tabla F.2.18: Clusters y Tópicos por requerimiento

Cluster	Itemsets	Tópicos
1	7	--
2	14	software, software system
3	15	internet, cyberspace
4	17	populace, public, world public
5	18	--
6	20	Indus, Indus River
7	22	contact contact, contacting contact, tangency liaison, link, contact, inter-group communication contact, touch contact, impinging, striking contact, middleman
8	3, 11, 23, 25	communication subject, content, depicted object content, cognitive content, mental object message, content, subject matter, substance content

		company company, troupe ship's company, company party, company caller, company capacity, content company, comradeship, companionship, good fellowship, fellowship, society contentedness, content
--	--	---

Tabla F.2.19: Clusters y Tópicos finales

F.2.6 SWMiner sin Normalización Morfológica y sin Clustering

La Tabla F.2.20 describe los tópicos que caracterizan a cada uno de los itemsets frecuentes.

Id Itemset	Tópicos
3	Company company, troupe company, comradeship, companionship, good fellowship, fellowship, society party, company caller, company ship's company, company
7	--
11	Communication
14	software, software system
15	internet, cyberspace
17	populace, public, world public
18	--
20	Indus, Indus River
22	contact contact, contacting contact, impinging, striking contact, middleman liaison, link, contact, inter-group communication contact, tangency contact, touch
23	content message, content, subject matter, substance capacity, content content, cognitive content, mental object contentedness, content

	subject, content, depicted object
25	--

Tabla F.2.20: Tópicos de los Itemsets Frecuentes