UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA INDUSTRIAL

ANÁLISIS DEL COMPORTAMIENTO DEL USUARIO EN LA WEB A PARTIR DE LA SIMULACIÓN DE SU NAVEGACIÓN USANDO OPTIMIZACIÓN DE COLONIA DE HORMIGA

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL INDUSTRIAL

PABLO SALVADOR LOYOLA HEUFEMANN

PROFESOR GUÍA:
JUAN D. VELÁSQUEZ SILVA

MIEMBROS DE LA COMISIÓN:
PABLO ROMÁN ASENJO
SEBASTIÁN RÍOS PÉREZ

SANTIAGO, CHILE
ABRIL 2011

RESUMEN DE LA MEMORIA
PARA OPTAR LA TÍTULO DE
INGENIERO CIVIL INDUSTRIAL
POR: PABLO LOYOLA HEUFEMANN
FECHA: 04/04/2011
PROF. GUÍA: JUAN D. VELÁSQUEZ S.

ANÁLISIS DEL COMPORTAMIENTO DEL USUARIO EN LA WEB A PARTIR DE LA SIMULACIÓN DE SU NAVEGACIÓN USANDO OPTIMIZACIÓN DE COLONIA DE HORMIGA

Este trabajo de memoria tiene por objetivo principal el diseño y aplicación de un modelo de comportamiento del usuario en un sitio web basado en la metaheurística de Optimización de Colonia de Hormiga (ACO por sus siglas en inglés).

Desde el comienzo de la Web se ha buscado saber cuál es la mejor estructura y contenido para un sitio de tal forma que se asegure la captura y recepción de usuarios. Una de las posibles soluciones consiste en personalizar la navegación del usuario web, es decir, adaptar el sitio dependiendo de las preferencias y hábitos que se detectan, con el fin de facilitar el acceso a la información requerida. Lo anterior se puede lograr a partir de la extracción de información y conocimiento desde los datos que origina el usuario en su navegación, que quedan registrados en los archivos de web logs. De esta forma surge la necesidad de generar modelos y herramientas que asistan en el proceso de *personalización* de los sitios web, cuyo mayor problema radica principalmente en los altos volúmenes de datos a utilizar como también en la diversa naturaleza de los mismos. Frente a esto, se propone explorar la factibilidad ACO como una alternativa basada en la generación colaborativa de sesiones de usuario.

Inicialmente se realizó un estudio de los métodos involucrados dentro de Web Mining, que abarcan desde la selección y preprocesamiento de los datos, hasta la obtención de conocimiento. Posteriormente se investigó la metaheurística de ACO desde sus bases biológicas hasta sus aplicaciones en problemas estándar, como también su utilización dentro del campo de Web Intelligence.

En base a lo anterior, se diseñó un modelo de ACO que incorpora el aprendizaje desde las sesiones de usuario real, a través de la modificación continua de un vector de preferencias por texto, simulando la utilidad que percibe un individuo al enfrentarse con el contenido de una página web. Luego, conjuntos de hormigas entrenadas generan, a través de un proceso colaborativo y autocatalítico, sesiones artificiales, las cuales son posteriormente contrastadas con sesiones reales. El modelo fue aplicado en el sitio web del Departamento de Ingeniería Industrial de la Universidad de Chile, del cual se extrajo tanto su estructura y contenido como las sesiones de sus visitantes. Los resultados obtenidos muestran que el modelo propuesto es capaz de ajustar en aproximadamente un 81 % los patrones de navegación reales, en relación con una medida de similitud que incorpora tanto las páginas visitadas como también el orden de éstas dentro de las sesiones.

En conclusión, es factible modelar el comportamiento del usuario en la Web a través de ACO en un nivel agregado, es decir, la identificación de las tendencias de comportamiento global por sobre las secuencias individuales. Asimismo, se debe señalar que dadas las múltiples variables existentes en los problemas relativos a Web Mining, es necesario realizar cambios considerables en la formulación tradicional de la metaheurística en estudio con el fin de lograr una adaptación coherente. Se propone como trabajo futuro el continuar con el desarrollo de mejores técnicas de extracción y preprocesamiento de los datos originados en la web, como también implementar los modelos en sitios más dinámicos tanto en su estructura como en contenido.

Agradecimientos

A mis padres, Katiuska Heufemann y Manuel Loyola, por el constante apoyo y cariño brindado durante todos estos años.

A mis profesores, Juan Velásquez y Pablo Román, por las oportunidades y enseñanzas que me dieron.

Pablo Loyola Heufemann
Abril, 2011

Índice general

Índice de figuras

Capítulo 1

Introducción

La evolución que ha tenido la Web en el último tiempo, ha transformado la forma en la que las empresas enfrentan el desafío constante de adaptarse a las preferencias de los usuarios, quienes cada vez concentran mayor protagonismo no sólo en términos del consumo sino también en relación con la generación del contenido. Un ejemplo de lo anterior lo representa el Cloud Computing, es decir, movimiento tanto de las aplicaciones de escritorio como de los medios de almacenamiento hacia productos y plataformas instaladas completamente en la red. La prestigiosa consultora Gartner[13] la situó como una de las diez tecnologías clave para el 2011. Por otra parte, varias empresas han migrado a la plataforma Google Apps en lugar de utilizar la suite Office de Microsoft[1], tal como ya lo ha realizado el City Counsil de la ciudad de Los Angeles, para reemplazar su antiguo sistema perteneciente a Novell[2].

En ese sentido la web está dejando de ser sólo un repositorio de datos e información, sino que ahora está convirtiéndose en una plataforma de trabajo real que permite la elaboración de productos y servicios. Luego, la disposición y estructura de los componentes y del contenido se vuelven factores determinantes a la hora de proporcionar una experiencia de usuario eficiente. Por ejemplo, entidades encargadas de la calidad de atención al cliente en la industria bancaria ya no sólo debe considerar la disposición de sus sucursales y cajas para disminuir las colas, sino también deben poner atención en cómo está construido su sitio web en relación con la curva de aprendizaje que debe seguir el usuario para realizar una transacción o el tiempo y número de clics que le toma encontrar su resumen financiero.

La gran competencia por captar nuevos clientes hace necesario elaborar herramientas que permitan estudiar su comportamiento y así poder focalizar estrategias tanto de marketing como de mejoramiento en la usabilidad de los sitios web.

En tal dirección, el Departamento de Ingeniería Industrial está llevando a cabo una iniciativa

[1]Google: Firms can 'get rid' of Office in a year - ZDNet Asia
[2]Los Angeles opts for Google: http://blog.seattlepi.com/microsoft/archives/183396.asp

que se relaciona con el estudio y análisis del comportamiento humano a través de Internet, trabajo encabezado por los profesores Juan Velásquez y Pablo Román.

Uno de los objetivos de tal iniciativa es el modelamiento del comportamiento del usuario web a través de diversas metodologías, con el fin de generar un contraste robusto que permita realizar un análisis más acabado. Una de las técnicas que se ha implementado tiene que ver con la simulación a través de Algoritmos Genéticos [2].

El presente informe tiene por objetivo presentar otro enfoque de simulación que se relaciona con el uso de la metaheurística de Optimización de Colonia de Hormiga. Esta técnica fue desarrollada hace aproximadamente veinte años y se inspira en el comportamiento que presentan las hormigas al buscar alimento. Esta metodología ha ganado cierta relevancia debido a los buenos resultados obtenidos al resolver tanto problemas estándar, como el ruteo de vehículos o la optimización de portafolios, hasta problemas clásicos como el Problema del Vendedor Viajero [10].

1.1. Objetivos

1.1.1. Objetivo General

Aplicar algoritmos de Optimización de Colonia de Hormiga (ACO por sus siglas en inglés) para simular el comportamiento que sigue un usuario en una página web en términos de la distribución de las preferencias por texto que presenta.

1.1.2. Objetivos específicos

1. Utilizar nuevas técnicas de optimización matemática como son los algoritmos de ACO.
    - Metodología: Llevar a cabo una recopilación bibliográfica sobre el tema, tomando como referencia el trabajo guía de Dorigo et al[11]. Estudiar sus bases teóricas como también sus aplicaciones prácticas.
2. Estudiar la factibilidad de aplicar la metaheurística señalada para el modelamiento de usuarios en la web.
    - Metodología: Realizar una revisión en términos de publicaciones que aborden el tema. Posteriormente diseñar una prueba de concepto que permita delinear los pasos a seguir y someterla a una aplicación real.
3. Evaluar el uso de librerías de simulación multiagente.
    - Metodología: Realizar un búsqueda de todas las librerías y paquetes disponibles. Posteriormente evaluar la factibilidad en función de los requerimientos técnicos y de la capacidad

de adaptabilidad al problema abordado. También se debe tener en cuenta las curvas de aprendizaje asociadas a su uso. Considerando todas esas variables se deberá tomar una decisión.

4. Realizar un procesamiento de los registros provenientes de los sitios del Departamento de Ingeniería Industrial, llevando a cabo un proceso de sesionalización y extracción de contenido que sirva como base para los experimentos. Adicionalmente, caracterizar el sitio en función de indicadores y estadísticas estándares.
   - Metodología: Explorar dentro de las técnicas de Web Mining disponibles y evaluar la factibilidad realizando pruebas exploratorias. Gestionar la obtención de los datos a procesar y sobre estos realizar un proceso que involucre desde la limpieza inicial hasta la transformación en un formato que sea compatible con los requisitos del modelo.
5. Realizar un análisis de los resultados obtenidos y evaluar la factibilidad extensiones o mejoras.
   - Metodología: Elaborar indicadores y métricas sobre las cuales poder evaluar los resultados obtenidos.

1.2. Hipótesis

Se pretende generar un modelo de comportamiento de usuario que tome como principal herramienta las preferencias por texto que siguen usuarios cuando visitan un sitio. Esto, a partir del análisis de las trayectorias seguidas y del contenido de las páginas visitadas, siguiendo un enfoque multiagente modelado a través de ACO.

1. Los usuarios llegan con una preferencia por texto a un sitio web.
2. La distribución de la preferencia por texto varía a medida que el usuario navega por el sitio.
3. El proceso de búsqueda de información dentro de un sitio web por parte de un usuario tiene una analogía con el proceso de búsqueda de alimento por parte de una colonia de hormigas, en el sentido de maximizar una utilidad a través de caminos factibles.
4. Los usuarios navegan por el sitio hasta que la utilidad que han obtenido (nivel de similitud con su distribución de preferencia por texto) llega a un máximo.
5. Una aproximación a la distribución de preferencias por texto sirve para caracterizar a los usuarios de un sitio, en el sentido de tener una noción de lo buscan.

1.3. Resultados esperados

En términos preliminares, se busca que el proyecto cumpla con los siguientes puntos:

- Sea capaz de modelar de forma coherente el comportamiento promedio de las distribuciones de preferencias por texto de los usuarios de los sitios asociados al DII, o por lo menos a un segmento significativo de ellos.
- Que la metaheurística de ACO aplicada al comportamiento de usuario web represente una alternativa real a las técnicas clásicas en el sentido de encontrar ventajas comparativas en función principalmente del planteamiento de los problemas con un enfoque multiagente.

1.4. Alcances del Trabajo

Los esfuerzos se orientan a lograr un modelo que posea una similitud coherente con la realidad. En ese sentido, la profundidad tanto a nivel de elaboración del modelo como de sofisticación técnica están condicionadas por la utilidad práctica que produzca el uso de la herramienta en desarrollo, es decir, se iterará un mejoramiento continuo hasta que el trabajo realizado se transforme en un verdadero factor para realizar una conclusión que permita dar término al trabajo y que englobe un aprendizaje global sobre los temas abordados.

1.5. Contribuciones

El trabajo realizado en esta memoria permitió la participación en las siguientes publicaciones:

- Pablo Loyola, Pablo E. Román, Juan D. Velásquez. Ant Colony Surfer: Discovering the Distribution of Text Preferences from Web Usage. Presentado en BAO '10: First Workshop on Business Analytics and Optimization. Santiago, Chile. Enero 2010.
- Pablo E. Román, Robert Dell, Juan D. Velásquez, Pablo Loyola. Optimization Models for Sessionization. Enviado a Intelligent Data Analysis.
- Pablo Loyola, Pablo E. Román, Juan D. Velásquez. Clustering-Based Learning Approach for Ant Colony Optimization Model to Simulate Web User Behavior. Enviado a The 2011 IEEE/WIC/ACM International Conference on Web Intelligence.

Capítulo 2

Marco Conceptual

En este capítulo se presenta una descripción general de los principales componentes conceptuales sobre los cuales será construido el trabajo. Este marco conceptual está formado por los siguientes elementos : Descripción de la Web en términos tanto de origen como de funcionamiento y estructura; Aproximación a la disciplina de Web mining, enfatizando sus características generales, usos y alcances; Descripción de la metaheurística de ACO.

2.1. World Wide Web

Para comprender la gestación de la World Wide Web, primero se debe tener una noción acerca del surgimiento de Internet.

La red Internet nace como parte de un proyecto científico en donde diversas universidades e institutos de investigación norteamericanos trabajaron en conjunto con el fin de desarrollar estructuras distribuidas para mejorar el acceso a la información. Esto generó un gran interés por parte del Ministerio de Defensa de los Estados Unidos, a través de su Agencia de Investigación de Proyectos Avanzados de Defensa (DARPA), quienes requerían construir una red que permitiera el acceso a información de forma descentralizada y segura, requisitos esenciales bajo el clima de inestabilidad derivado de la Guerra Fría. Desde ese momento, pasó a ser un proyecto científico militar y uno de los primeros hitos relevantes ocurrió en 1969 cuando se abrió el primer nodo de la red ARPANET y comenzó una expansión a través de entidades universitarias. Este crecimiento no sólo permitió la comunicación directa entre computadores, sino también a través de redes internas. Este fenómeno le otorgó el nombre inicial a red : Inter-Net.

El acceso a esta red se mantuvo restringido por un largo tiempo, debido a la complejidad requerida para operarla. Su uso fue principalmente la comunicación a través de un incipiente correo electrónico. A principios de la década de 1990 el físico británico Tim Berners-Lee propone una nuevo

modelo que permite mejorar la calidad del acceso a la información dentro de la red utilizada por el CERN, Centro Europeo para la Investigación Nuclear [36].

Esta nueva forma contemplaba la función de compartir hipertexto, el cual se define como la representación electrónica de texto que permite, a través de algún tipo de interacción, enlazar otro texto, ya sea del tipo hipertexto o plano.

Los posteriores avances tanto a nivel de la red Internet como el propio desarrollo de la World Wide Web han permitido llegar hasta el concepto que hoy en día se tiene, es decir, un conjunto de protocolos con el cual es posible compartir todo tipo de documentos, ya sean en forma de texto como en otros medios como sonido o imagen.

2.1.1. Funcionamiento

Paradigma Cliente/Servidor

El funcionamiento de la Web se basa en la utilización de paradigma Cliente/ Servidor [40], el cual comprende las siguientes etapas:

En primer lugar un cliente web, comúnmente un navegador, ejecuta una petición a un servidor web determinado por medio de una URL[1], que sirve de identificador único de documentos en la Web y que tiene la siguiente estructura estandarizada [30]:

```
<protocolo>://<maquina>/<objeto>
```

Donde:

- `<protocolo>`

  Representa el protocolo a utilizar para la comunicación. Existen diversos protocolos,siendo HTTP[2] y FTP[3] los más utilizados en un entorno Web.

- `<maquina>`

  Representa el nombre del servidor asociado o su dirección.

- `<objeto>`

  Representa la ruta dentro del servidor donde se encuentra alojado el documento que el cliente está requiriendo.

[1] Uniform Resource Locator.
[2] Hyper Text Transfer Protocol.
[3] File Transfer Protocol.

El servidor web recibe el requerimiento y retorna el documento solicitado. Esta respuesta es enviada a través del mismo protocolo utilizado por el cliente.

Lenguaje HTML

Los documentos necesitan un lenguaje estandarizado que permita una visualización y funcionamiento coherente. HTML[4] es el principal lenguaje utilizado para la elaboración de hipertexto. Se basa en la disposición de contenido, a través de etiquetas o *tags*, cada una con una función específica.

Un documento típico en lenguaje HTML está divido en dos secciones principales. La primera es el HEAD, en donde se incorporan etiquetas que sirven para identificar el documento, añadir metadata o incluir archivos externos como hojas de estilo CSS[5] o archivos Javascript[6].

- link, script, style: Permiten incluir archivos.
- title, meta: Permiten añadir metadata.

La segunda sección principal es la llamada BODY, la cual incluye todo el contenido propio del documento, como también su estructura visual. Las etiquetas más importantes son:

- Etiquetas relacionadas con la estructura del sitio y la disposición del los elementos y contenido:
  - div: Utilizada principalmente para la división de bloques temáticos.
  - ul, li,ol: Utilizadas para mostrar listas de elementos.
- Etiquetas relacionadas con el contenido en texto:
  - h1 $\rightarrow$ h6: Utilizadas para representar jerarquías en títulos.
  - b, italic: Utilizadas para añadir formato a texto.
  - p, table: Utilizadas para separar párrafos y generar tablas.
- Etiquetas relacionadas con la inclusión de elementos multimedia:
  - img, object: Inserción tanto de imágenes como de objetos (sonido o animaciones) .
- Etiquetas relacionadas con la generación de hipervínculos:
  - a: Permite enlazar un documento con otro.

[4] HyperText Markup Language.
[5] Cascading Style Sheets : Lenguaje utilizado para gestionar la presentación visual de un documento web.
[6] Lenguaje interpretado que se ejecuta en el lado de cliente y que es usado para mejorar la experiencia de usuario.

Las etiquetas admiten la asignación de atributos específicos que permiten por un lado establecer parámetros, como también identificar de forma única un determinado elemento para, por ejemplo, su posterior modificación a través de una hoja de estilo CSS o de alguna función ejecutada desde Javascript.

De esta modo la formulación estandarizada de una etiqueta en HTML tendría la siguiente forma:

```
<nombre_etiqueta atributo="valor_atributo">contenido</nombre_etiqueta>
```

Inicialmente los atributos permitían explícitamente manejar el aspecto visual tanto de la estructura como del contenido del documento web. Las nuevas versiones del lenguaje han ido privilegiando el uso de hojas de estilo para tal fin, dejando al lenguaje HTML el rol de articulador del contenido y divisor de bloques temáticos.

La última versión del lenguaje, HTML 5[7], representa un cambio sustancial en las directrices de desarrollo, ya que incluye nuevas etiquetas que permiten añadir un componente semántico a la estructura del documento.

Un documento HTML usual debería tener la siguiente forma:

```
<html>

<head>
<title>Título del documento</title>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<link rel="stylesheet" href="hoja_estilo.css" type="text/css" />
<script src="js_script.js" type="text/javascript"></script>
</head>

<body>
<div id="contenedor">
<h1>Título</h1>
<p>
Párrafo de ejemplo con un enlace a
<a href="http://www.google.cl">Google</a>
</p>

<ul>
<li>Item 1</li>
<li>Item 2</li>
```

[7] http://dev.w3.org/html5/spec/Overview.html

```
<li>Item 3</li>
</ul>
</div>

</body>

</html>
```

2.1.2. Características Generales de la Web

Siguiendo el texto de Bing Liu [21], la Web posee las siguientes características:

- La cantidad de información en la Web es enorme y crece día a día cubriendo básicamente todos los campos.
- La información y los datos están en todos los formatos existentes. Desde tablas estructuradas, pasando por documentos semiestructurados hasta textos sin estructura alguna, como también archivos multimedia.
- La diversidad de autores, dada la facilidad de publicación de documentos, genera heterogeneidad en la información.
- La mayoría de la información presente tiene la propiedad de estar enlazada. Esta unión se establece a través del uso de hipervínculos, los cuales existen entre documentos pertenecientes a un mismo sitio, donde sirven como medio de organización, como también entre sitios, donde representan una medida de autoridad.
- La información contenida posee mucho ruido. Típicamente un documento web, a parte de su contenido principal, posee bloques adicionales que contienen datos poco relevante para el análisis como anuncios publicitarios, menús, etc. Adicionalmente, el ruido se genera por la inexistencia de un control de calidad sobre los documentos generados.
- La Web es dinámica. Cada día se generan nuevos documentos y los ya existentes sufren cambios.
- La Web incluye también servicios el comercio electrónico, las operaciones bancarias, etc.
- La Web es una sociedad virtual. Más allá de la transferencia de datos, información y servicios, es un medio de interacción entre las personas.

Como se puede apreciar, la complejidad asociada a la Web representa una oportunidad en términos del desafío que implica la construcción de métodos y modelos que sean efectivos a la hora de análisis.

2.1.3. Datos originados en la Web

El proceso de petición y respuesta que rige el funcionamiento de la Web queda registrado íntegramente a través del uso de web logs, archivos de texto plano que se almacenan en la máquina donde se aloja el servidor web y que contienen cada una de las peticiones junto con datos útiles para realizar análisis. Si bien existen algunas diferencias en la forma en la que los distintos servidores web registran los datos que conforman cada registro, éste posee un generalmente el siguiente conjunto de campos:

- Marca de Tiempo, fecha y hora de la petición.
- Dirección IP del cliente solicitante.
- Ruta del documento solicitado.
- Host asociado dentro del servidor donde se encuentra el documento.
- Estado de la solicitud, código definido en función del éxito de la respuesta del servidor. Puede tomar diversos valores, por ejemplo, 200 (Éxito), 404 (No se encontró archivo solicitado), 403 (No tiene los privilegios suficientes), etc.
- User Agent, identificador de la aplicación que utiliza el cliente para realizar la petición, comúnmente un navegador web.
- Referer, identificación del documento desde el cual se hizo la petición. Comúnmente hace referencia a una página que puede estar dentro o fuera del servidor.

```
2010-09-12 00:39:03 WEB4 GET /ajax/ShadowBox/src/js/shadowbox.js - - 190.110.155.237 Mozilla/4.0+(compatible;+MSIE+8.0;+Windows+NT+6.1;+Trident/4.0;
2010-09-12 00:39:03 WEB4 GET /_css/P0018_General.css - - 201.220.233.84 Mozilla/4.0+(compatible;+MSIE+8.0;+Windows+NT+6.1;+Trident/4.0;+GTB6.5;+SLCC
2010-09-12 00:39:03 WEB4 GET /ajax/menuRecursosPortal/java.js - - 187.132.87.31 Mozilla/4.0+(compatible;+MSIE+7.0;+Windows+NT+5.1;+.NET+CLR+2.0.5072
2010-09-12 00:39:03 WEB4 GET /UserFiles/P0024/Image/ok_bot_familia.gif - - 186.20.182.80 Mozilla/5.0+(Windows;+U;+Windows+NT+6.0;+en-US)+AppleWebKit
2010-09-12 00:39:04 WEB4 GET /UserFiles/P0001/Image/Mod_3_contenidos_estudiantes_ciencias_biologia/Dibujo+7.jpg - - 189.131.115.239 Mozilla/4.0+(com
2010-09-12 00:39:04 WEB4 GET /Portal.Base/Web/verContenido.aspx ID=185730 - 186.20.182.80 Mozilla/5.0+(Windows;+U;+Windows+NT+6.0;+en-US)+AppleWebKi
2010-09-12 00:39:04 WEB4 GET /ntg/estudiante/1626/channels-917_titulo_recursos.gif - - 190.152.155.208 Mozilla/5.0+(Windows;+U;+Windows+NT+5.1;+es-E
2010-09-12 00:39:04 WEB4 GET /ajax/menuRecursosPortal/navegador.js - - 187.132.87.31 Mozilla/4.0+(compatible;+MSIE+7.0;+Windows+NT+5.1;+.NET+CLR+2.0
2010-09-12 00:39:04 WEB4 GET /Portal.Base/Web/verContenido.aspx ID=41 - 186.10.28.175 Mozilla/4.0+(compatible;+MSIE+8.0;+Windows+NT+6.0;+Trident/4.0
2010-09-12 00:39:04 WEB4 GET /_css/P0012_General.css - - 186.10.28.175 Mozilla/4.0+(compatible;+MSIE+8.0;+Windows+NT+6.0;+Trident/4.0;+SLCC1;+.NET+C
2010-09-12 00:39:04 WEB4 GET /ntg/estudiante/1626/propertyvalues-39406_ordena_estudiante.gif - - 190.152.155.208 Mozilla/5.0+(Windows;+U;+Windows+NT
2010-09-12 00:39:04 WEB4 GET /Portal.Base/js/jsFuncionesPortal.js - - 201.220.233.84 Mozilla/4.0+(compatible;+MSIE+8.0;+Windows+NT+6.1;+Trident/4.0;
2010-09-12 00:39:04 WEB4 GET /ntg/estudiante/1626/propertyvalues-38432_bullet_estudiante.gif - - 190.152.155.208 Mozilla/5.0+(Windows;+U;+Windows+NT
2010-09-12 00:39:04 WEB4 GET /ntg/estudiante/1626/propertyvalues-38434_bullet_r_estudiante.gif - - 190.152.155.208 Mozilla/5.0+(Windows;+U;+Windows+
2010-09-12 00:39:04 WEB4 GET /ntg/estudiante/1626/propertyvalues-38433_bullet_estudiante.gif - - 190.152.155.208 Mozilla/5.0+(Windows;+U;+Windows+NT
2010-09-12 00:39:04 WEB4 GET /ajax/SpryAssets/SpryMenuBar.js - - 190.110.155.237 Mozilla/4.0+(compatible;+MSIE+8.0;+Windows+NT+6.1;+Trident/4.0;+SLC
2010-09-12 00:39:04 WEB4 GET /ntg/estudiante/1626/channels-917_green.gif - - 190.152.155.208 Mozilla/5.0+(Windows;+U;+Windows+NT+5.1;+es-ES;+rv:1.9.
2010-09-12 00:39:04 WEB4 GET /UserFiles/P0001/Image/home2010/barrasuperior/p_esencial.gif - - 187.132.87.31 Mozilla/4.0+(compatible;+MSIE+7.0;+Windc
2010-09-12 00:39:04 WEB4 GET /UserFiles/P0001/Image/home2007/varias/p_familia.gif - - 187.132.87.31 Mozilla/4.0+(compatible;+MSIE+7.0;+Windows+NT+5.
2010-09-12 00:39:04 WEB4 GET /_css/css_docente/css_docente2009.css - - 186.10.28.175 Mozilla/4.0+(compatible;+MSIE+8.0;+Windows+NT+6.0;+Trident/4.0;
2010-09-12 00:39:04 WEB4 GET /UserFiles/P0001/Image/home2007/varias/educarchile.gif - - 187.132.87.31 Mozilla/4.0+(compatible;+MSIE+7.0;+Windows+NT+
2010-09-12 00:39:04 WEB4 GET /UserFiles/P0001/Image/home2007/varias/p_estudiantes.gif - - 187.132.87.31 Mozilla/4.0+(compatible;+MSIE+7.0;+Windows+N
2010-09-12 00:39:04 WEB4 GET /UserFiles/P0001/Image/home2007/varias/p_home.gif - - 187.132.87.31 Mozilla/4.0+(compatible;+MSIE+7.0;+Windows+NT+5.1;+
2010-09-12 00:39:04 WEB4 GET /UserFiles/P0001/Image/home2007/varias/buscador.gif - - 187.132.87.31 Mozilla/4.0+(compatible;+MSIE+7.0;+Windows+NT+5.1
2010-09-12 00:39:04 WEB4 GET /UserFiles/P0001/Image/home2007/centrado/fondo_sombra2.jpg - - 187.132.87.31 Mozilla/4.0+(compatible;+MSIE+7.0;+Windows
```

Figura 2.1: Ejemplo de un archivo web log. Elaboración propia

Como se puede apreciar, los archivos web logs contienen bastantes datos en relación a la actividad que se produce constantemente en el servidor web, por lo que se convierten en una fuente valiosa para realizar posteriores estudios sobre las preferencias y el comportamiento que siguen los usuarios. En la próxima sección se estudian técnicas de extracción de información teniendo como fuente los web logs.

2.2. Web Mining

Dadas las características propias de la Web anteriormente expuestas, cuando se requiere realizar un análisis que permita encontrar información más allá de la que se pueda obtener de manera directa a través de la estadística básica, se requiere de herramientas más sofisticadas que guíen el procedimiento desde la recolección de datos hasta el análisis de las conclusiones y resultados. Tales herramientas están englobadas dentro de la disciplina de la Minería de Datos. Luego, la adaptación de éstas al procesamiento y análisis de los datos en la Web, da paso a la Minería de Web, o Web Mining. La clasificación general de disciplina toma en cuenta las dimensiones sobre las cuales se forja el estudio de la Web. La literatura hace la siguiente distinción [21]:

- Web Structure Mining (WSM): Se centra en el estudio de la estructura de un sitio web, en relación con los documentos que lo componen y de cómo éstos están interrelacionados a través de enlaces o hipervínculos.
- Web Content Mining (WCM): Tiene que ver con el análisis del contenido dentro de los documentos asociados al sitio web. Extrae conocimiento útil entorno a clasificaciones del texto.
- Web Usage Mining (WUM): Se refiere a la obtención de los patrones de navegación que siguen los usuarios dentro de un sitio web.

Si bien esta división es útil para organizar y caracterizar los problemas y la forma de abordarlos, comúnmente se utilizan en conjunto, debido a la estrecha relación que existe entre los tres componentes y al enfoque multivariado de los problemas que se deben enfrentar.

La Figura 2.2 muestra las tareas principales en las que intervienen los diferentes tipos de enfoque como también las que se realizan cuando tales se complementan.

2.2.1. Proceso KDD

El proceso KDD, Knowledge Discovery in Databases, es decir, el Descubrimiento de Conocimiento en Bases de Datos, es la columna vertebral de la metodología clásica en Minería de Datos.

Este procedimiento abarca desde la obtención de los datos iniciales y su procesamiento, hasta el análisis de los resultados, es decir, es un proceso ordenado e iterativo que permite pasar de datos a conocimiento [21].

Cuando se adapta el proceso KDD a una tarea de Web Mining, la filosofía inicial sigue intacta. Se podría decir que las diferencias con la Minería de datos tradicional se encuentran en la fase de preprocesamiento de los datos a utilizar [21]. Esto porque comúnmente los datos de la Web

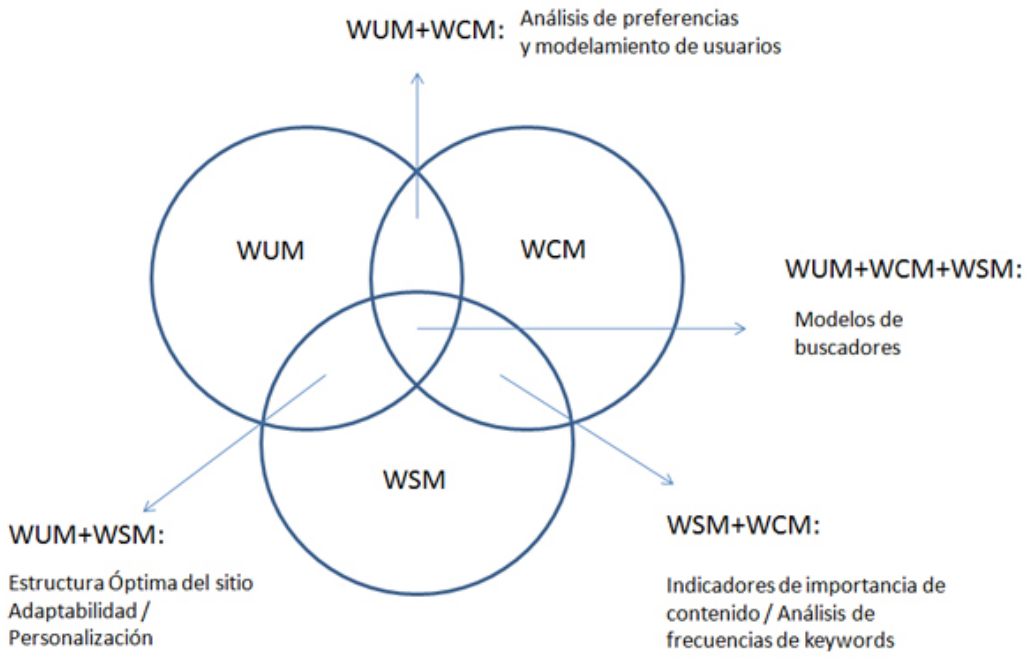

Figura 2.2: Interacción entre los componentes de Web Mining. Elaboración propia.

no se encuentran debidamente estructurados, al contrario de lo que sucede en Minería de Datos tradicional, donde se asume que los datos están almacenados en un medio que posee cierto orden. Los pasos del proceso KDD son los siguientes [17, 24]:

- **Selección**: En primer lugar se evalúan las fuentes de datos disponibles en relación a la utilidad o afinidad con el problema que se desea abordar.
- **Preprocesamiento** : Dada la diversidad de fuentes y métodos de recolección, los datos comúnmente contienen errores tanto de consistencia, manipulación o son simplemente irrelevantes para el análisis posterior [40], por lo que es necesario llevar a cabo una limpieza. Adicionalmente se debe pasar desde la forma semiestructurada que proporciona el lenguaje HTML a una estructura de tabla o vector.
- **Transformación** : Una vez con los datos limpios, se llevan a una estructura vectorial que permita un mejor manejo operativo, como también su adaptación a los formatos requeridos por los algoritmos de a utilizar. En el caso particular de Web Usage Mining, es en esta etapa en la cual se lleva a cabo el proceso de Sesionalización.
- **Minería de Datos** : Los datos son procesados a través de algoritmos afines a los requerimientos del problema abordado.
- **Evaluación Experta** : Los resultados obtenidos son evaluados por el criterio de un experto, el cual decide sobre la coherencia y viabilidad del procedimiento seguido. Se delinea la estrategia

a seguir, en términos de realizar cambios en las siguientes iteraciones. Adicionalmente se realizan análisis de sensibilidad con el fin de probar la robustez de los modelos [7].

- **Obtención de Conocimiento** : Una vez que el proceso ha alcanzado la calidad requerida y se asegura que los resultados son generados de manera rigurosa, se extrae la información y el conocimiento derivado y se concluye sobre los mismos.

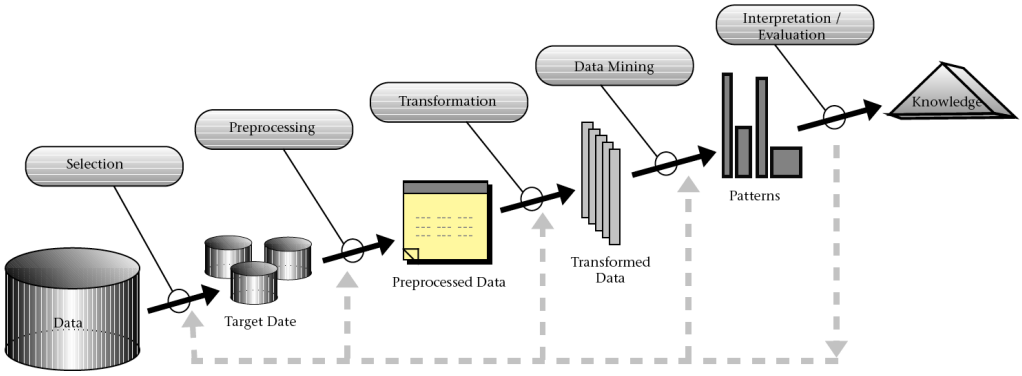

Figura 2.3: Proceso KDD. Fuente: http://www.infovis-wiki.net

2.2.2. Técnicas de Minería de Datos

A continuación se presentan ciertas técnicas que han sido elaboradas con el fin de obtener información y conocimiento útil desde los datos.

Reglas de Asociación

Esta técnica consiste en encontrar correlaciones entre atributos o conjuntos de datos. Una representación formal del problema es la siguiente: Sea $I = i_1, ...i_m$ un conjunto de items y $T = t_1, ..., t_n$ un conjunto de transacciones. Una transacción se define como un elemento que contiene ciertos elementos de I, luego, para $X \subseteq I$, se dirá que una transacción t_i contendrá a X si $X \subseteq t_i$. De esta forma, una *regla de asociación* es una implicación de la forma $X \rightarrow Y$, donde $X \subseteq I$, $Y \subseteq I$ y $X \cap Y = \emptyset$ [40].

Por ejemplo, se requiere analizar cómo están relacionados los productos vendidos en un supermercado. Sea I el conjunto de todos los productos disponibles. Una transacción será simplemente el conjunto de productos comprados por una persona en particular, por ejemplo $t = \{carne, pollo, queso\}$.

Una regla de asociación podría ser: $\{carne, pollo\} \rightarrow queso$, donde $\{carne, pollo\}$ será el conjunto X y *queso* será el conjunto Y.

Existen dos variables por las cuales se puede medir la fuerza de una regla de asociación:

- **Soporte**: El soporte de una regla $X \rightarrow Y$ es el porcentaje de transacciones en T que contienen $X \cup Y$, de esta forma se puede interpretar como una aproximación de la *factibilidad* de que se de tal regla en el conjunto T. Sea n el número de transacciones, el soporte de la regla $X \rightarrow Y$ viene dada por:

$$soporte = \frac{\text{Nº de elementos en } (X \cup Y)}{n} \tag{2.1}$$

- **Confianza**: La confianza de una regla $X \rightarrow Y$ es el porcentaje de transacciones que dado que contienen X, también contienen a Y, formalmente:

$$confianza = \frac{\text{Nº de elementos en } (X \cup Y)}{\text{Nº de elementos en } X} \tag{2.2}$$

  De esta forma se puede estimar la capacidad predictiva de la regla, ya que si por ejemplo su valor es muy bajo, implica que no es posible inferir Y a partir de X.

Como toda técnica de minería de datos, el criterio del especialista es requerido. En ese sentido, se evalúan las reglas en función de la definición de soportes y/o confianzas mínimas, las cuales son establecidas en dependiendo del problema que se esté atacando y del conocimiento experto del analista.

Aprendizaje Supervisado

El aprendizaje supervisado ha tenido un uso bastante generalizado dentro de las tareas propias de la Minería de Datos. Fue una de las primeras herramientas analíticas en este campo debido a que en el fondo sigue el paradigma del aprendizaje humano, es decir, al tomar las experiencias vividas en el pasado, se obtiene conocimiento útil para mejorar ciertas habilidades y respuestas en el futuro [21].

Formalmente, se tiene un conjunto de datos que poseen una serie de atributos $A = A_1, A_2, ..., A_n$. Este conjunto también posee un atributo especial C llamado *atributo clase*. Dado un conjunto de datos D, el objetivo del proceso de aprendizaje es generar una función de clasificación y/o predicción que relacione los atributos en el conjunto A con las clases pertenecientes a C.

A este tipo de aprendizaje se le llama *supervisado* ya que los valores que puede tomar la clase C ya están contenidos en los datos, es decir, existen como categorías a priori.

Se requiere en primer lugar de un *Conjunto de entrenamiento*, es decir, datos del pasado que contengan tanto los atributos de A como los valores de C. Una vez que el modelo ha sido entrenado, se debe evaluar su precisión. Para esto se utiliza un nuevo conjunto, llamado *Conjunto de prueba*, que también corresponde a datos del pasado (usualmente tanto el Conjunto de entrenamiento como el de prueba provienen de una misma fuente la cual es dividida en dos según una proporción arbitraria).

El modelo entrenado es alimentado con el conjunto de prueba, pero sustrayendo el valor de la clase. Luego el modelo es ejecutado y entregará un valor predictivo de clase para cada elemento del conjunto de prueba. Finalmente se debe comparar los valores originales de la clase con los obtenidos desde el modelo.

La exactitud de un modelo de clasificación se define como:

$$exactitud = \frac{\text{Nº de clasificaciones correctas}}{\text{Nº de elementos en el conjunto de prueba}} \tag{2.3}$$

El entrenamiento es un proceso iterativo, en cual el especialista debe ir realizando cambios sucesivos que permitan un incremento en la tasa de exactitud.

Especial atención subyace en uno de los supuestos implícitos dentro de este procedimiento: se asume que las distribuciones presentes en el conjunto de entrenamiento son iguales a las que presenta el conjunto de prueba. En aplicaciones reales lo anterior no siempre se cumple, por lo que es necesario asegurar que ambos conjuntos poseen una similitud aceptable antes de comenzar.

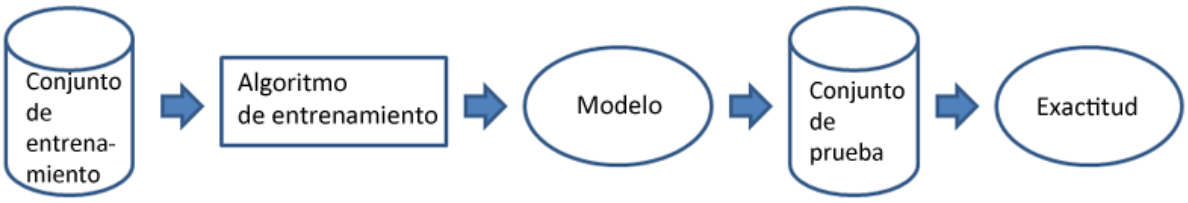

Figura 2.4: Proceso de entrenamiento. Basado en [21].

Con respecto a los métodos de evaluación, existen ciertas técnicas para visualizar y explorar la coherencia de los resultados:

- **Muestreo Aleatorio Múltiple**: Utilizando el conjunto completo de datos disponible, se extraen n duplas de conjuntos entrenamiento-prueba distintos. Se entrena y evalúa el modelo con cada uno de ellas, lo cual entregará n *exactitudes*. El promedio es éstas puede ser utilizado como un valor representativo de la exactitud global del modelo.

- **Validación Cruzada**: Este método consiste en dividir todos los datos disponibles en n conjuntos disjuntos de igual tamaño. Cada subconjunto es usado como conjunto de prueba, mientras que los $n-1$ restantes son combinados y pasan a conformar el conjunto de entrenamiento. Este proceso se repite n veces, lo cual se traduce en el cálculo de n exactitudes, cuyo promedio puede ser utilizado como un valor representativo. Usualmente n toma un valor entre 5 y 10.

- **Precision, Recall y F-score**: Estos indicadores son útiles cuando las clases en el conjunto de datos están muy desvalanceadas y/o se está interesado solamente en la clase que se sabe minoritaria [21] (llamada *clase positiva*). Es el caso de los problemas relacionados con la detección de fraude financiero o la intrusión en redes. La utilización de la exactitud en estos casos no es una medida recomendable, ya que su valor puede ser alto, pero esto no necesariamente implica que el modelo sea un buen predictor (si se analizan datos de fraude financiero, probablemente el 99 % no represente fraude, luego el clasificador puede lograr una exactitud del 99 %, pero este valor no tiene utilidad, ya que lo relevante es encontrar al 1 % que si representa fraude). La utilización de los valores de *precision* y *recall* en estos casos es más conveniente. Su cálculo requiere inicialmente de la construcción de una matriz de confusión sobre los resultados reales y los predichos por el modelo.

	clasificado positivo	clasificado negativo
positivo	TP	FN
negativo	FP	TN

donde :

- **TP**: Número de clasificaciones correctas de las instancias positivas (verdadero positivo).
- **FN**: Número de clasificaciones incorrectas de las instancias positivas (falso negativo).
- **FP**: Número de clasificaciones incorrectas de las instancias negativas (falso positivo).
- **TN**: Número de clasificaciones correctas de las instancias negativas (verdadero negativo).

Luego, en base a la matriz de confusión, *precision* p y *recall* r se definen de la siguiente forma:

$$p = \frac{TP}{TP+FP} \tag{2.4}$$

$$r = \frac{TP}{TP+FN} \tag{2.5}$$

Así, *precision* es el porcentaje de instancias correctamente clasificadas como positivas dentro dentro de todas las que fueron clasificadas como positivas y *recall* representa el número de instancias correctamente clasificadas como positivas dividido por el número de instancias positivas en la realidad. En base a estos dos indicadores , se define el F-score como:

$$F = \frac{2pr}{p+r} \tag{2.6}$$

que representa la media armónica entre los dos. Esto ayuda a unificar el diagnóstico asociado el modelo, balanceando la relevancia de los valores anteriores.

Por ejemplo, sean 100 casos positivos (no se saben cuales a priori) dentro de un conjunto de 10.000 instancias. Se entrenó un modelo y éste entregó como respuesta que 200 casos son positivos. Se comparó con los datos reales para ver la precisión de la predicción y se obtuvo la siguiente matriz:

	clasificado positivo	clasificado negativo
positivo	60	40
negativo	140	9.760

De esta forma, para saber qué porcentaje de los casos positivos fueron predichos, utilizando la medida de *recall*, se obtiene un 60 %. Si se desea saber el porcentaje total de predicciones correctas, la medida de *precision* muestra un 30 %.

Aprendizaje No Supervisado

La principal diferencia entre el aprendizaje supervisado y el no supervisado es que en éste último los valores que pueden llegar a tomar las clases no están definidas *a priori*. Luego, la tarea es llevar a cabo una exploración de los datos con el fin de encontrar estructuras intrínsecas que permitan obtener información y conocimiento útil. *Clustering* es el término que engloba el proceso de aprendizaje no supervisado. Consiste en la progresiva organización de las instancias en grupos o *clusters*. Los elementos conformantes de cada grupo son *similares* entre ellos, pero *disímiles* con respecto a los demás grupos en función de la definición de una *distancia*.

Los métodos de clustering se dividen principalmente en dos categorías:

- **Clustering particional**: La idea detrás de este método tiene que ver con, dado un conjunto de datos iniciales, definir un número de clusters y llevar a cabo un particionamiento de los datos en función de una función de distancia. A continuación se especifica la noción de clustering particional a través del algoritmo K-Means, uno de los más extendidos [40].

  Dado un conjunto de datos, se define el conjunto $X = x_1, x_2, ..., x_n$, donde $x_i = (x_{i1}, x_{i2}, ..., x_{ir})$ es un vector que representa el valor de los r atributos posibles que pueden tener los elementos de X. K-Means particionará el conjunto en k clusters, número definido a priori. Cada uno de los clusters resultantes tendrá un valor central, o *centroide*, el cual servirá como entre representativo.

  Al inicializar el algoritmo, se escogen a aleatoriamente k puntos, los cuales son designados como centroides iniciales. Luego se calcula la distancia entre cada punto y cada centroide, lo cual conlleva a la asociación de cada punto con el centroide con el que esté más cerca. Posteriormente, cada centroide es re-calculado utilizando los datos pertenecientes a su cluster. Este proceso se repite hasta que se cumple al menos uno de los siguientes criterios de convergencia (queda a juicio del experto cuales utilizar dependiendo de la naturaleza del problema) :

- Al recalcular, ninguna instancia cambió de cluster.
- Al recalcular, no hubo cambio de centroide dentro de los clusters.
- No hubo una reducción significativa en el valor de la suma de los errores cuadráticos (SSE), la cual se define como:

$$SSE = \sum_{j=1}^{k} \sum x \in C_j dist(x, m_j)^2 \tag{2.7}$$

donde k es el número de clusters requerido, C_j es el *j-ésimo* cluster, m_j es el centroide del cluster C_j y $dist(x, m_j)$ es la función de distancia evaluada entre la instancia x y el centroide m_j.

Una de las principales virtudes del algoritmo de K-Means es su simplicidad, como también su facilidad de implementación. Adicionalmente se puede mencionar que se considera lineal en tiempo con respecto a la cantidad de datos. Por otra lado, sus debilidades se concentran principalmente en los siguientes aspectos:

- Sólo es aplicable en un conjunto de datos donde la noción de *media* exista (existen variaciones para datos con atributos categóricos).
- Se requiere especificar el número de clusters antes de ejecutar el algoritmo, lo cual en ciertos casos puede condicionar los resultados.
- El algoritmo es sensible ante la presencia de outliers, lo cual genera clusters incoherentes en ciertos casos. Para remediar esta situación se pueden llevar a cabo eliminaciones selectivas en función de ciertos límites establecidos (por ejemplo, mediante una inspección previa). También se puede realizar un muestreo aleatorio a través del cual pre-clusterizar un subconjunto pequeño de elementos (en donde la probabilidad de seleccionar un outlier es menor) para posteriormente asignar el resto sobre la base de una partición de mejor calidad.
- El comportamiento del algoritmo está condicionado a los centroides iniciales (escogidos aleatoriamente). Luego, distintos centroides iniciales llevarán a distintas soluciones de clusterig.

- **Clustering jerárquico**: Este método consiste en generar una representación de tipo árbol con los clusters. A esta estructura se le llama *dendograma*. Clusters de un sólo elemeto (*singleton*) se encuentran en la parte inferior del árbol y un cluster raíz en la parte superior. Cada cluster interno contiene a los clusters que están bajo él. A continuación se presenta un ejemplo basado en el texto de Bing Liu [21].

  En la parte inferior del árbol en la Figura 2.5 se pueden ver 5 clusters con un elemento cada uno, que representan el conjunto de datos a clusterizar. En el siguiente nivel, el cluster 6 contiene los datos 1 y 2 y el cluster 7 los datos 4 y 5. El número de clusters disminuye conforme se sube hacia la raíz.

  Existen dos tipos de construcción del clustering y están dados por el sentido de recorrido del árbol:

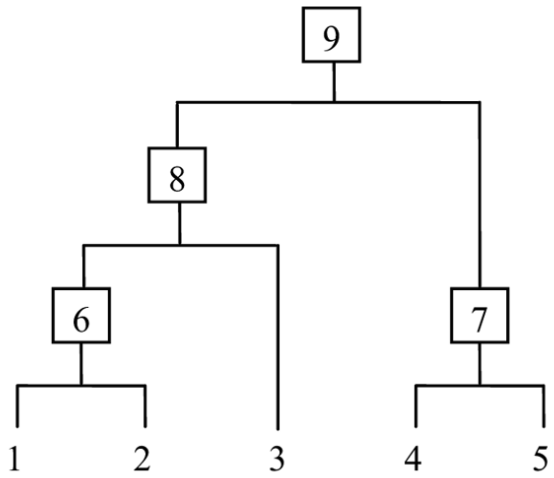

Figura 2.5: Ejemplo de clustering jerárquico. Basado en [21].

- **Aglomerativo**: Recorre el árbol desde abajo hacia arriba uniendo el par de clusters que resulten más similares en cada nivel.
- **Divisivo**: Recorre el árbol desde arriba hacia abajo, comenzado en la raíz que contiene todos los elementos y en cada paso se va dividiendo en clusters hijos en relación al par de clusters que tenga la distancia mayor.

Con respecto al tratamiento de la distancia, existen métodos específicos, los cuales son analizados a continuación:

- **Método de enlace simple (Single-Link Method)**: Este método une en cada paso los dos clusters cuyos elementos más cercanos (uno de cada cluster) tengan la mínima distancia. Este método es susceptible a sufrir el llamado *efecto de encadenamiento*, es decir, cuando ruido presente entre dos clusters se transforma en un puente entre ambos con lo cual terminan uniéndose.
- **Método de enlace completo (Complete-Link Method)**: Este método une en cada paso los dos clusters cuyos elementos más lejanos (uno de cada cluster) tengan la mínima distancia. Este método puede tener dificultades en presencia de datos *outliers*. Usualmente produce mejores resultados que el método de enlace simple [21].
- **Método de enlace medio (Average-Link Method)**: Buscando un equilibro entre los métodos anteriores, en este caso la distancia se define como la distancia promedio entre todos los pares de elementos, cada uno en un cluster diferente. Luego se busca el menor valor para unir.

Las principales ventajas del clustering jerárquico, comparándolo con el particional, es que tiene la capacidad de computar cualquier tipo de distancia o función de similitud. Asimismo, a diferencia del clustering particional, donde se debe fijar a priori un número de clusters,

el clustering jerárquico permite la exploración progresiva en cualquier nivel, lo cual resulta bastante útil, por ejemplo, al aplicarlo en texto. Por otro lado, las mayores debilidades del clustering jerárquico provienen de su intensivo uso recursos y espacio. En relación con esto, la literatura señala que tiene un pobre desempeño cuando el conjunto de datos es considerablemente grande [21] (más aún en términos de la visualización del dendograma).

En conclusión, no es que un método de clustering sea mejor que otro, sino que dependiendo de las propiedades intrínsecas del problema, emergen ciertas ventajas de unos sobre otros. En aplicaciones reales, la elección del tipo de clustering a utilizar es un desafío debido a múltiples factores como la heterogeneidad de las distribuciones presentes en los datos, la necesidad de definir criterios de convergencia y funciones de distancia entre otros parámetros. En ese sentido, la comparación exploratoria entre métodos como también la experiencia al momento de interpretar los resultados tienen una relevancia importante.

2.2.3. Análisis del contenido y estructura de un sitio Web

Extracción y preprocesamiento

El primer paso consiste en la extracción del contenido desde el propio sitio web. El método de extracción más común es el basado en la utilización de *web crawlers*, programas que esencialmente recorren el sitio web a través de la estructura del hiperlinks, registrando cada documento, su contenido y los enlaces que posee. Si la exploración es continua a través del tiempo, puede también identificar los cambios que han realizado en el sitio. Los datos recolectados son enviados a un medio de almacenamiento estandarizado.

En la Figura 2.6 se puede apreciar en mejor medida el funcionamiento de un web crawler convencional. En primer lugar se configura un documento raíz que representa el punto de partida para el programa, el cual comienza a viajar por los documentos siguiendo la estructura de hipervínculos. Cada documento es capturado a través del módulo *Web Page Downloader*, el cual registra la url en el medio temporal *Pile* además de añadir una marca en *Scheduler* el que se encarga de gestionar las peticiones. Los datos quedan inicialmente almacenados en *Staging Area* donde ya se posee una estructura relacional. En último término se lleva a cabo una labor de limpieza y corrección, en la etapa de *Pre-processing* antes de su registro final en los *Web Data*.

La labor de preprocesamiento merece atención especial debido a que representa una de las fases más críticas de todo proceso de Web Mining. Tal como se señaló al explicar el proceso KDD, si los datos que se utilizan son pobres en calidad, también lo serán los resultados. La heterogeneidad de la Web hace necesario llevar a cabo una serie de pasos con el fin de estandarizar el contenido. A continuación se presentan los más relevantes:

- Remoción del código HTML presente en el documento, con el fin de dejar solamente el texto

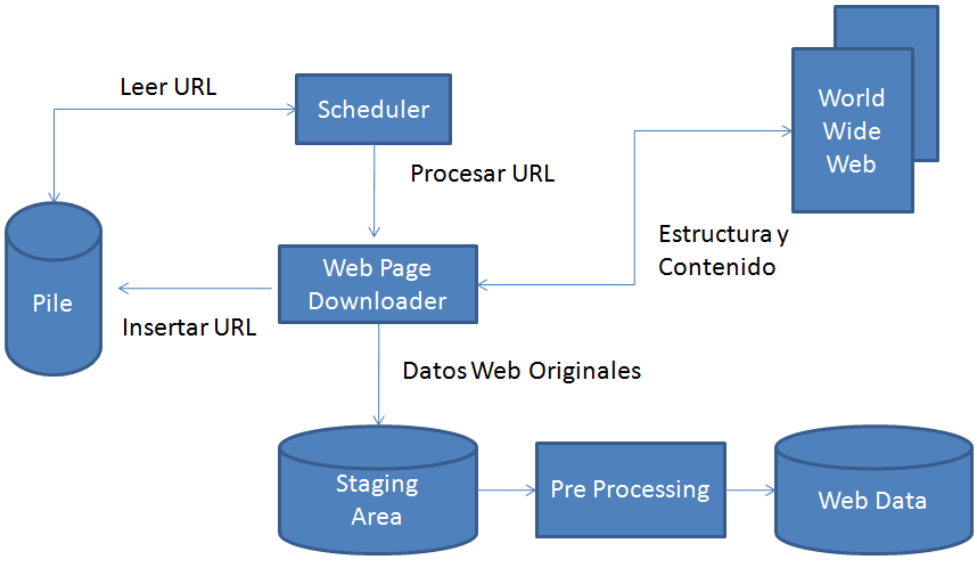

Figura 2.6: Modelación general de un web crawler. Elaboración propia basado en [39].

plano asociado al contenido.

- Remoción de las *stop words*, que engloban a todos los componentes gramaticales que permiten construir frases, pero que no representan contenido propiamente tal. Dentro de este conjunto encontramos los artículos, preposiciones y conjunciones.
- Las palabras restantes deben pasar por un proceso de *stemming* o lematización, que consiste en llevar una palabra a su forma raíz a través de la eliminación de prefijos y sufijos. Esto ayuda a que la frecuencia de un palabra no se penalice debido a las transformaciones propias de su uso. Uno de los algoritmos más extendidos es el de Porter [21].

Adicionalmente se pueden utilizar las etiquetas del código HTML para asociar un peso específico (o un nivel de importancia) a ciertas palabras. Por ejemplo, el contenido dentro de las etiquetas *title* o *h1* tiene una connotación diferente al texto un párrafo.

Análisis de la estructura

Los sistemas de información tienden a evolucionar en arquitecturas jerárquicas [34]. Esto tiene un trasfondo en la necesidad de generar *robustez*, en el sentido de que la falla en un componente no afecte al resto, como también la búsqueda de la *eficiencia* en la búsqueda. Estudios sugieren que los sistemas de información eficientes tienen un origen en un proceso social descentralizado que evoluciona a través del tiempo hasta alcanzar un configuración estable [26].

En relación con la Web, se ha demostrado que los sitios web tienden a concentrar sus links entre sus propias páginas y no hacia otros sitios. En [12] se estudió la estructura jerárquica de más de 600 millones de páginas, en donde se muestra que el 75 % de los links se producen entre páginas del mismo sitio. La estructura de links en la Web ha evolucionado en ausencia de un diseño global centralizado, en ese sentido ha sido necesaria la creación de una clasificación que permita visualizar la importancia de las páginas:

- **Authorities**: Páginas que son apuntadas por un número importante páginas provenientes de otros sitios.
- **Hubs** : Páginas que apuntan a un número importante de páginas, principalmente del tipo Autorities. Tienen un rol organizador y clasificador de la información.

En función de esta noción de *importancia* asociada al número de enlaces (algo análogo a los indicadores de citación en publicaciones científicas), se han desarrollados algoritmos que permiten establecer un ranking, elemento primordial en sistemas de búsqueda.

- **Pagerank**: Este algoritmo tiene una naturaleza estática, donde el *score* que se le asigna a cada página no depende de una búsqueda en particular. Está inspirado en la noción de prestigio en redes sociales:
  - Un hipervínculo que nace una página y apunta a otra se puede interpretar como un traspaso de *autoridad* desde la primera a la segunda. Así, mientras más links reciba una página, más prestigio tiene asociado.
  - Las páginas que apuntan a otras también tienen su propio prestigio. De esta forma, que una página con alto prestigio apunte tiene mayor importancia que si lo hace una de menor prestigio.

  Con lo anterior, el Pagerank de un página i se determina por la suma de los valores de Pagerank de todas las páginas que apuntan a i. Como una página puede apuntar a conjunto de otras páginas, en este caso su Pagerank se comparte para realizar el cálculo. Las ventajas de este algoritmo tienen que ver con su tendencia natural a subvalorar las páginas de *spam* [21]. Por otro lado, sus mayores críticas apuntan a que no toma en cuenta el contexto de búsqueda, es decir, no distingue entre páginas que puedan ser autoritativas globalmente de las que lo son relacionadas con un tópico en particular.

  El algoritmo Pagerank fue introducido en 1998 por Sergei Brin [8] y sirvió como base para la construcción del motor de búsqueda de Google.
- **HITS**: Al contrario de Pagerank, HITS (Hypertext Induced Topic Search), es un algoritmo que depende de la consulta. Cuando el usuario realiza una búsqueda, HITS retorna todas las páginas que estén asociadas con el término ingresado y posteriormente elabora dos rankings, uno en función de páginas autorities, y otro de páginas hubs. El supuesto principal del algoritmo se basa en que una buena página hub apunta buenas páginas autorities y una buena página

autorities es apuntada por buenas página hub [21]. De esta forma se genera una relación de reforzamiento mutuo. Las ventajas de este algoritmo tienen relación con que la elaboración del ranking conlleva una dependencia del término buscado por el usuario. Por el contrario, posee algunas desventajas como su baja habilidad para detectar páginas de spam como también el uso intensivo en tiempo de procesamiento, dada la elección inicial del conjunto de páginas a rankear.

Cabe señalar que la descripción anterior de los algoritmos corresponde a sus versiones iniciales. Muchos de los problemas presentados han encontrado solución, pero tales avances se encuentran protegidos y no han sido publicados en la literatura especializada [21].

Análisis del contenido

En primer lugar se debe llevar a cabo una transformación del contenido preprocesado con el fin de que pueda ser interpretado y analizado de manera cuantitativa. Una de las representaciones más utilizadas viene dada por el Vector Space Model. En este modelo cada documento se representa como un vector donde cada valor representa el peso específico que posee cada término dentro del documento [21].

Para calcular este peso se utilizan los términos TF (term frequency) e IDF (inverse document frequency) :

Sea N el número total de documentos a analizar y df_i el número de documentos en los cuales el término t_i aparece al menos una vez. Sea f_{ij} el número de apariciones (frecuencia) del término t_i en el documento d_j. Luego, el valor de la frecuencia del término normalizada (*normalized term frequency*) del término t_i en el documento d_j viene dada por la Ecuación 2.8.

$$tf_{ij} = \frac{f_{ij}}{max f_{1j}, f_{2j}, ..., f_{|V|j}}, \tag{2.8}$$

donde el máximo se calcula sobre todos los términos que aparecen en el documento d_j. Si el término t_i no aparece en el documento d_j, entonces el valor de $tf_{ij} = 0$. El valor de $|V|$ representa el tamaño del vocabulario asociado a la colección de documentos.

La frecuencia de documento inversa del término t_i se calcula a través de la Ecuación 2.9.

$$idf_i = \log \frac{N}{df_i} \tag{2.9}$$

Luego el valor del peso de una palabra en un documento viene dada por el parámetro TF-IDF, el cual se calcula de la forma:

$$TF - IDF_{ij} = tf_{ij} * idf_i \tag{2.10}$$

Asimismo, una consulta q efectuada por un usuario puede ser representada de la misma forma vectorial. En este caso, el peso de cada término w_{iq} dentro de la búsqueda puede ser calculado como un valor de TF-IDF.

Uno de los aspectos relevantes del modelo Vector Space Model es que permite comparar ya sea dos documentos como también un documento y una consulta, a través del uso de la medida similitud del coseno. Sea q un vector de consulta y d_j el vector correspondiente al documento j, la medida de similitud en función del ángulo entre ambos está dada por la Ecuación 2.11.

$$cos(d_j, q) = \frac{\langle d_j \bullet q \rangle}{||d_j||||q||} \tag{2.11}$$

De esta forma el Vector Space Model proporciona tanto una forma de cuantificar el contenido como también una función de comparación.

2.2.4. Análisis del comportamiento del usuario en la Web

Extracción de los datos

Una sesión de usuario se define como la secuencia de documentos visitados desde que inició su recorrido por el sitio web hasta que lo abandonó. Intuitivamente se puede avizorar la el volumen de datos que se puede llegar a extraer, cuyo posterior procesamiento implicaría la obtención de información valiosa tanto sobre el comportamiento como sobre las preferencias que siguen los visitantes.

El proceso a través del cual se reconstruyen las sesiones recibe el nombre de *sesionalización*. La literatura divide los métodos de sesionalización en dos categorías principales [39]:

Estrategias proactivas: En primer lugar están aquellas que hacen uso de las *cookies*, medio de almacenamiento temporal de datos que consiste en archivos planos que se guardan dentro de la máquina del usuario y que son modificados y recuperados a través del navegador web a petición del servidor web. Los datos que comúnmente se almacenan tienen que ver con la identificación del usuario, los documentos que visitó y el tiempo en que lo hizo. Las desventajas de este tipo de métodos provienen de dos fuentes:

- Las cookies pueden ser borradas por los usuarios: Los navegadores web entregan a sus usuarios herramientas tanto propias como a través de extensiones para controlar la generación y almacenamiento de cookies. En ese sentido, usuarios informados pueden tomar medidas en caso de considerarlas una amenaza a su privacidad.
- El seguimiento a través de cookies es considerado ilegal bajo ciertas jurisdicciones : La nula notificación de las intenciones por parte de sitios web convierte al uso de cookies en un medio de seguimiento que no posee el consentimiento explícito del usuario.

En segundo lugar se encuentra la utilización se software de tipo *spyware* que se instala sin el consentimiento del usuario y que tiene como función monitorear la actividad que éste realiza para luego enviarla a un servidor externo donde se analiza y se ocupa para fines comerciales o de segmentación.

En síntesis, las estrategias proactivas se basan en la identificación exacta y el seguimiento total del usuario para la reconstrucción de las sesiones.

Estrategias reactivas: A diferencia de las estrategias proactivas, las reactivas se basan en la utilización de fuentes de datos anónimos sobre los que se aplican metodologías que entregan sesiones de manera aproximada.

La fuente de datos más común es la que proviene de los archivos web logs. Si bien éstos son un fiel reflejo de la actividad generada en el servidor web, al mismo tiempo poseen la problemática de la suciedad de sus datos. Por una parte se almacenan peticiones de elementos poco relevantes para el proceso de sesionalización, como son las imágenes, scripts, archivos de estilo, etc., y por otro lado se registra la actividad de rastreadores o *crawlers*. De esta forma, se hace necesario inicialmente realizar una labor de limpieza y preprocesamiento de los datos. Posteriormente se ejecutan heurísticas de reconstrucción que pueden ser clasificadas de la siguiente forma [39]:

- **Heurísticas tradicionales**: Se basan en un agrupamiento de los registros dentro de los archivos web log en base a los atributos que en mayor medida podrían caracterizar a un usuario, por ejemplo, la dirección IP o el valor de User Agent. Dentro de esta categoría está la sesionalización basada en tiempo máximo, que consiste en, luego de un agrupamiento por dirección IP y User Agent, cortar arbitrariamente cada 30 minutos, bajo el supuesto de que una sesión real no debería durar más allá de tal intervalo de tiempo. También se encuentran los métodos de reconstrucción basados en la topología propia del sitio web, en los cuales el criterio para terminar una sesión y comenzar otra es que la secuencia de páginas registrada se apegue estrictamente a la estructura del links. Por lo tanto, si luego de agrupar por dirección IP y por User Agent se aprecia que el usuario ha pasado desde una página a otra no habiendo un enlace real entre ellas, entonces tal hito debe marcar el fin de una sesión y el comienzo de otra.
- **Heurísticas basadas en ontologías**: Utilizan el enriquecimiento de los registros iniciales con información semántica proveniente de las mismas páginas visitadas. Sobre estos datos

adicionales se define una medida de similitud y posteriormente se agrupan las páginas en función de su cercanía. Esto bajo el supuesto de que en la secuencia de páginas que el usuario sigue dentro del sitio subyace un propósito o una preferencia definida por contenido.

- **Heurísticas basadas en modelos de Programación Lineal**: La naturaleza inherentemente combinatorial del proceso de sesionalización ha permitido la implementación de modelos de programación lineal como un método de resolución que incluya las múltples restricciones e instancias del problema. Al contrario de los métodos anteriores, donde las sesiones se generan a medida que se recorren los registros, en los métodos basados en Programación Lineal éstas generan simultáneamente bajo funciones objetivo que implican la maximización o minimización del número de sesiones.

El comportamiento del usuario Web

La búsqueda y el uso de la información es una característica intrínseca de la especie humana en su necesidad por adaptarse a los problemas y situaciones del día a día. Esta propensión ha llevado a investigadores como George Miller a designar a los humanos como *infornívoros*: una especie que busca, reúne y almacena información con el propósito de adaptarse al mundo que le rodea. Esta noción lleva a la necesidad de comprender cómo está configurada la interacción entre los humanos y la información. Citando a Simon se puede entender que el principio que guía esta relación puede ser entendido de la siguiente forma: *La información consume la atención de quien la recibe. Luego, la abundancia de información genera un pobreza de atención y es ahí donde surge la necesidad de distribuir esa atención de manera eficiente* [35].

En las sociedades modernas la gente interacciona con la información a través de la tecnología, la cual en teoría tiene como función mejorar la calidad y los tiempos de búsqueda. En ese sentido se tiene el supuesto de que si se mejoran las condiciones con las cuales las personas buscan, interpretan y hacen uso de la información, se mejora la probabilidad de producir un comportamiento inteligente [26].

La interacción entre los seres humanos y la información debe tender a maximizar el valor del conocimiento adquirido por unidad de costo involucrado en tal interacción. Por ejemplo, la evidencia muestra que ciertos sistemas sensoriales han evolucionado a lo largo del tiempo en función de absorber más unidades de información por gasto calórico asociado a tal actividad [26].

El sentido del párrafo anterior tiene una connotación evolucionista. La selección natural favoreció a los organismos que fueron más eficientes en la tarea de obtener energía desde el medio ambiente para llevar a cabo el proceso de reproducción. Luego, los organismos constantemente han evolucionado sus estrategias de adaptación como de búsqueda de alimento. En la actualidad, los sistemas perceptivos y cognitivos asociados a la búsqueda de información provienen de los primitivos métodos utilizados por nuestros ancestros para buscar alimento.

Uno de los supuestos principales al momento de estudiar el comportamiento del usuario es

el *Principio de racionalidad*, sobre el cual el sistema cognitivo es capaz de optimizar la adaptación del comportamiento del organismo [3].

Adicionalmente existe un segundo componente a considerar. A diferencia de actividades comunes estudiadas bajo la disciplina de la *Interacción Humano-Computador* como el uso, por ejemplo, de procesadores de texto, donde las utilidades asociadas a cada acción están bien definidas, la búsqueda a través de grandes volúmenes de información, como la Web, involucra incertezas que forman entornos probabilísticos donde los agentes deben tomar decisiones.

En [26] se presenta un modelo de comportamiento de usuario llamado *A Stochastic Model of Patch Foraging*, es decir, un modelo estocástico de búsqueda de *trozos* de información. Un trozo de información es una instancia o contenedor que es accesible y que contiene cierta cantidad de información determinada. Por ejemplo, una página web. En este modelo se tiene una regla principal bajo la cual un agente busca información en una instancia hasta que el potencial esperado de esa instancia sea menor que el promedio esperado de moverse a otra. El estado de un agente buscador de información en un instante i se representa por X_i ($X_i = x$ es un estado en particular), y se asume que esta variable contiene el identificador de la página web en la que se encuentra. La utilidad U es una función del estado y puede ser representada como :

$$U(x) \quad = \quad E[U|X_i = x] \tag{2.12}$$

El costo de tiempo esperado en la exploración posterior de otro estado, dado que se encuentra en el estado x, viene dado por:

$$t \quad = \quad E[T|X_i = x] \tag{2.13}$$

donde T es una variable aleatoria que representa el costo de tiempo futuro.

El valor de $U(x)$ por buscar por un tiempo t en la instancia actual debe ser balanceado dado el costo oportunidad $C(t)$ de explorar para encontrar otra instancia. Luego, se define la *función potencial* $h(x)$ de buscar en la intancia actual como :

$$h(x) \quad = \quad U(x) - C(t) \tag{2.14}$$

La idea es que el buscador de información debe maximizar esta función. En ese sentido, se debe cumplir que la utilidad por buscar en la instancia actual debe ser mayor o igual al promedio de los retornos de hacerlo en otras instancias. De lo contrario se detiene.

Este tipo de comportamiento en su forma agregada produce un fenómeno que la literatura llama *The Law of Surfing*, es decir, la Ley de Navegación, que se representa a través de la obtención de una distribución de los largos de las sesiones por número de usuarios y puede ser representada por una Gaussiana inversa [15], tal como se muestra en la Figura 2.7.

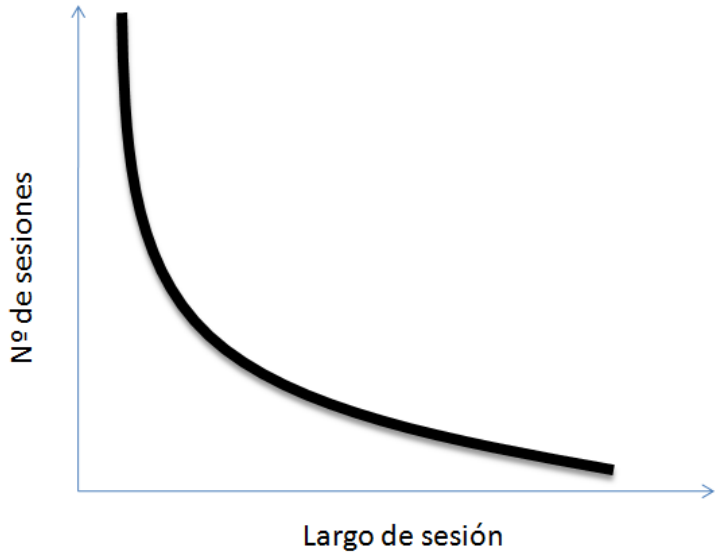

Figura 2.7: Ejemplo de la distribución del largo de las sesiones. Elaboración propia basado en [26].

La Ley de navegación tiene dos supuestos relevantes:

- Se asume que las sesiones que generan los usuarios pueden ser modeladas como caminos aleatorios o *random walks*. Cada paso del usuario tiene asociado un nivel de utilidad relativo a la similitud del contenido de la página destino con su preferencia. Formalmente, se asume que la utilidad esperada de continuar al próximo estado (página web) X_t está estocásticamente relacionada a la utilidad esperada del estado actual X_{t-1};

$$U(X_t) \quad = \quad U(X_{t-1}) + \varepsilon_t \tag{2.15}$$

- El segundo supuesto tiene que ver con que el agente se mueve hasta que se se sobrepasa un umbral de utilidad determinado.

Caracterización vectorial de las preferencias

Un enfoque se puede ver en [40] donde se define un vector de comportamiento de usuario (UBV) de la forma:

$$\upsilon \quad = \quad [(p_1, t_1), ..., (p_n, t_n)] \tag{2.16}$$

donde la dupla (p_i, t_i) se compone del identificador de la i-esima página visitada y la proporción del tiempo total de sesión que el usuario estuvo en esta página. Junto a esta definición se propone una medida de similitud entre vectores de comportamiento. Sean α y β dos vectores de comportamiento de usuario UBV con cardinalidad C^α y C^β respectivamente. Se tiene la función Γ que retorna los identificadores de las páginas provenientes de los UBV en la forma de una secuencia de navegación. La medida de similitud entre los vectores viene dada por:

$$sm(\alpha, \beta) \quad = \quad dG(\Gamma(\alpha), \Gamma(\beta)) \frac{1}{\eta} \sum k = 1^{\eta} \tau_k * dp(p_{\alpha,k}, p_{\beta,k}) \tag{2.17}$$

donde $\eta = min C^\alpha, C^\beta$, $\tau_k = min \frac{t_k^\alpha}{t_k^\beta}, \frac{t_k^\beta}{t_k^\alpha}$ es un indicador del interés del usuario en las páginas visitadas y $dp(p_{\alpha,k}, p_{\beta,k})$ es una medida de similitud entre la k-esima página del vector α y la k-esima página del vector β [30, 40].

Los supuestos detrás de esta configuración asumen que existe una relación directa entre el interés que un usuario le asigna a una página y el tiempo que destina en su visita.

Siguiendo la misma lógica, si se desea tener una noción cuantitativa de las páginas más importantes para un usuario dentro de un sitio web, se define el vector de páginas importantes (IPV) :

$$\vartheta_i(\upsilon) \quad = \quad [(\rho_1, \tau_1), ..., (\rho_i, \tau_i)], \tag{2.18}$$

que representa las i páginas más importantes para el usuario en relación con el tiempo.

2.2.5. Limitaciones y Alcances de Web Mining

Si se analiza tanto la literatura especializada como los anuncios de prensa, se puede concluir que las limitantes de las disciplinas asociadas a la extracción de conocimiento desde bases de datos tienen que ver principalmente con dos aspectos. En primer lugar está la limitante técnica que supone la enorme cantidad de contenido que se genera día a día en la Web. La revolución que ha representado la instauración de la llamada Web 2.0, es decir, una Web donde el usuario es el protagonista tanto en términos de audiencia como también en el rol de autor o generador de contenido, ha representado un salto cuántico en términos los ordenes de magnitud de los datos involucrados. En el siguiente gráfico se muestra la evolución de los *tweets* que se generan diariamente a través de Twitter[8], llegando a la cifra de noventa millones, en donde el 26 % contiene enlaces a otros documentos.

[8] http://www.twitter.com

Luego las preguntas que se generan son ¿Cómo gestionar tal cantidad de datos? ¿Qué técnicas tanto de almacenamiento como de procesamiento utilizar para encontrar patrones útiles en un tiempo coherente?

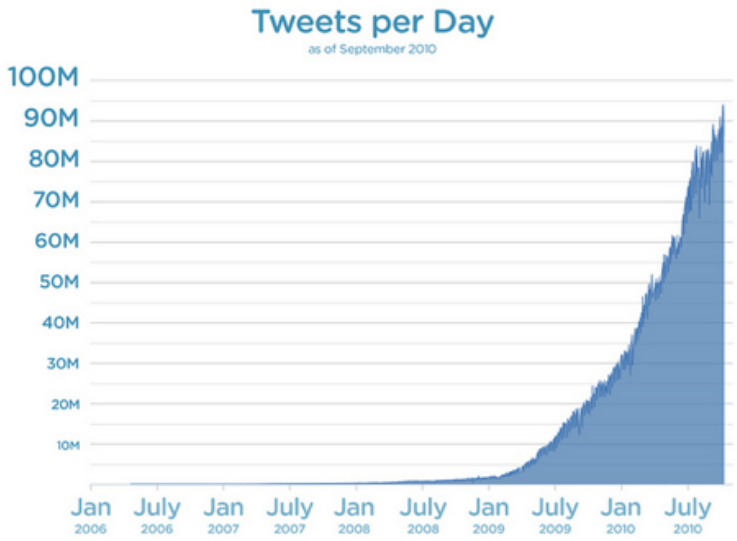

Figura 2.8: Número de *tweets* diarios. Fuente : http://techcrunch.com/2010/09/14/twitter-seeing-90-million-tweets-per-day

Lo anterior también puede ser visto como un desafío en términos de generar cada vez algoritmos y modelos más eficientes, dada una cantidad de recursos limitada, que representen verdadera utilidad en relación con el mejoramiento continuo de las plataformas web y un mayor conocimiento de los usuarios (potenciales clientes), ambos aspectos requeridos por las empresas e instituciones actualmente.

La segunda limitante identificada tiene relación con los marcos legales que rigen las actividades ligadas con el análisis de datos y que tienen que ver principalmente con el resguardo legítimo de la privacidad de las personas. Se puede apreciar cierta heterogeneidad en las normas que regulan la actividad, lo cual repercute en los grados de acción y la profundidad en la que pueden ser utilizadas ciertas herramientas [41].

Si bien nominalmente el propósito de la utilización de Web Mining es netamente altruista, en el sentido de mejorar la experiencia del usuario como también asistirlo para que encuentre lo que busca de manera eficaz y eficiente, no es difícil avizorar su utilidad para lograr beneficios pecuniarios [41].

El problema señalado es de bastante interés, en el sentido de que involucra una dimensión ética en un ambiente eminentemente técnico. Por ejemplo, cuando se habla de la dirección IP, ¿Es adecuado catalogarla como un dato personal? o ¿ Solamente debe tomarse como un medio de reconocimiento para el establecimiento de una conexión exitosa entre diferentes máquinas?, ¿Qué criterios deberían aplicarse? Estas preguntas no tienen una única respuesta y dependen del

contexto en el que se esté trabajando.

En Chile existe legislación al respecto, principalmente relacionada con la Ley 19.628 [38] la cual establece ciertas restricciones al procesamiento de los datos con el fin de garantizar la privacidad de los usuarios. La tendencia mundial apunta a transparentar los procesos de recolección y análisis de los datos pidiendo la autorización explícita a los usuarios, explicando las políticas de privacidad. Esto se debe a los cambios paradigmáticos de la Web (Web 2.0), como también el creciente interés, aún minoritario, por parte de la población por ejercer su derecho a la privacidad.

En conclusión, ambos elementos que se consideran limitantes a la hora de llevar a cabo un procedimiento de Web Mining probablemente estén siempre presentes, por lo que es labor tanto de investigadores y profesionales del área el saber internalizarlos y poder utilizarlos a su favor, en el sentido de visualizar nuevas oportunidades como también forjar un ética coherente no olvidando que en el fondo se está trabajando con datos e información perteneciente a seres humanos.

2.3. Optimización de Colonia de Hormiga

En esta sección se presenta una descripción acabada de la metaheurística a utilizar, desde su inspiración biológica, hasta sus aplicaciones en problemas relacionados con Web Intelligence. Cabe señalar que se utilizará la abreviación ACO para referirse a Optimización de Colonia de Hormiga.

2.3.1. Bases biológicas

Las ciencias biológicas, en su constante exploración y estudio de nuestro medio, han visualizado un fenómeno bastante particular en ciertos tipos de insectos. Se trata de la presencia de un componente social en su estructura organizativa, lo cual permite llevar a cabo tareas complejas a través de un proceso de cooperación y coordinación. Las hormigas son una de las especies en las que este fenómeno se ha estudiado con mayor profundidad, detectando el componente colaborativo en las siguientes actividades:

- Búsqueda de alimento
- Transporte cooperativo
- Agrupamiento de cadáveres

Todas estas actividades tienen en común el uso de un medio de comunicación indirecto que se basa en la modificación del entorno. Este concepto se conoce como *estigmergia*, acuñado inicialmente por Pierre-Paul Grassé en 1959 [11], quien lo definió de la siguiente forma:

La estimulación de los trabajadores en función del rendimiento que han logrado.

Los estudios biológicos han demostrado que los comportamientos a nivel de colonia observados en insectos sociales, pueden ser explicados a través de modelos basados en la utilización de la comunicación estigmérgica, en relación a los niveles de auto organización obtenidos.

Ahora la pregunta que surge es ¿Cómo se lleva cabo el proceso de comunicación estigmérgica? ¿Qué elementos físicos intervienen o son intervenidos?

Las hormigas poseen un sentido de la visión bastante limitado, y algunas especies son completamente ciegas. Una respuesta a esta situación es la utilización de una sustancia química para comunicarse. Esta sustancia es llamada *feromona* y es producida por las mismas hormigas [11].

Se conocen diversos tipos de feromona, pero hay una que particularmente tiene relevancia en el comportamiento social, llamada *feromona de rastreo*, la cual está presente en algunas especies de hormigas como la *Lasius niger* o la *Iridomyrmex humilis* (también conocida como hormiga argentina), que es utilizada específicamente para marcar caminos en la tierra entre el nido y la fuente de alimento. De esta forma, las hormigas reconocen el trazo químico y pueden seguir los caminos descubiertos por sus pares.

A continuación se presentan los experimentos que muestran el funcionamiento de la colaboración indirecta a través del uso de feromonas y que sirvieron como inspiración principal para la elaboración del método heurístico de ACO.

2.3.2. Experimentos del puente doble

La interacción de las hormigas a través del uso de feromona ha sido investigada ampliamente a lo largo del tiempo mediante experimentos controlados. Uno de los experimentos más célebres fue diseñado y llevado a cabo por Jean-Louis Deneubourg en 1990 y que consistió en analizar el comportamiento de la especie *I. humilis* en términos de los caminos recorridos entre el nido y una fuente de alimento a través de un puente doble. El parámetro principal del experimento es la razón entre las longitudes de los brazos que conforman el doble puente.

Sean l_l y l_s el brazo largo y el corto respectivamente. Se define la razón entre ambos como:

$$r = \frac{l_l}{l_s} \tag{2.19}$$

En el primer experimento realizado se dejaron los dos brazos con la misma longitud (r =1) y se dejó viajar a las hormigas libremente por el puente entre el nido y la fuente de alimento. El

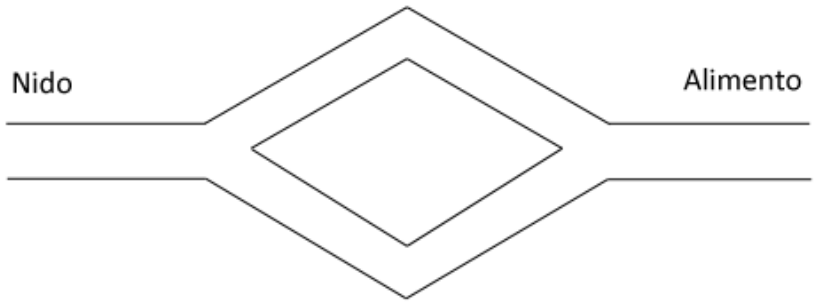

Figura 2.9: Esquema del experimento de puente doble con ambos brazos de igual longitud ($r = 1$) Elaboración propia basado en [10].

resultado mostró que aunque inicialmente se produjo una elección aleatoria entre los dos brazos, al pasar el tiempo las hormigas tendieron a elegir sólo uno de ellos. La respuesta de los investigadores a este fenómeno fue que si bien inicialmente no había rastros de feromona en ninguno de los brazos y las hormigas elegían con la misma probabilidad qué camino tomar, las fluctuaciones propias de una elección probabilística inclinarán una opción sobre la otra. Al suceder esto, los niveles de feromona sufrirán un desequilibrio que progresivamente modificará la elección de cada hormiga. A su vez, el incremento de hormigas por un brazo elevará las niveles de feromona en el mismo. Este ciclo se repetirá hasta que finalmente las hormigas converjan en la utilización de un sólo camino. Esto representa un proceso *autocatalítico* que genera un comportamiento auto-organizativo en las hormigas. Citando a Dorigo [10]:

> *Un patrón macroscópico (la convergencia hacia un sólo camino) emerge de procesos e interacciones que se dan en un nivel microscópico.*

Posteriormente se realizó el segundo experimento en el cual se estableció el brazo más largo con el doble de longitud del más corto ($r = 2$).

En este caso se pudo apreciar una preferencia clara a utilizar sólo el brazo más corto, a pesar de que inicialmente las hormigas elegían con la misma probabilidad cualquiera de los dos. La respuesta a este fenómeno es la siguiente: Una hormiga que elige el brazo más corto necesariamente llegará primero a la fuente de alimento. Cuando emprenda el viaje de regreso, deberá decidir por cual brazo seguir. El mayor nivel de feromona en el brazo corto ejercerá una influencia sobre la

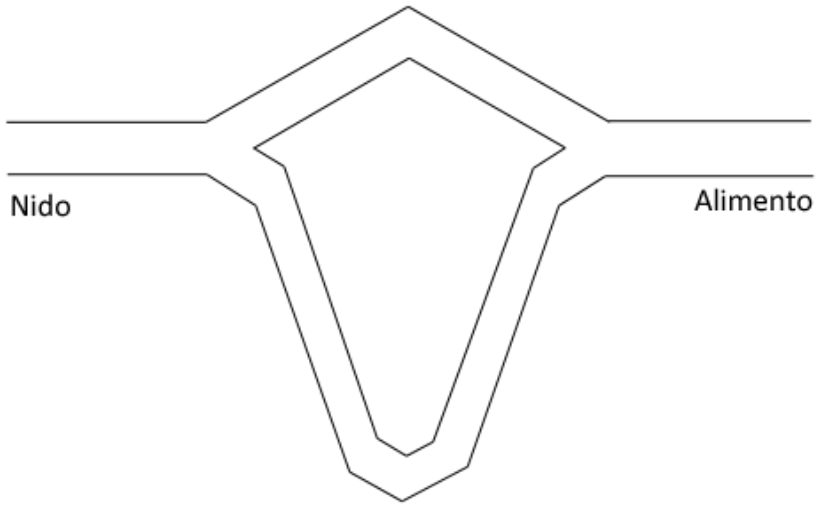

Figura 2.10: Esquema del experimento de puente doble con una brazo el doble de largo que el otro. ($r = 2$) Elaboración propia basado en [10].

hormiga que le llevará a elegirlo. De esta forma se activa el ciclo autocatalítico antes señalado. En este experimento, la influencia de las fluctuaciones aleatorias iniciales juega un menor papel, dejando el protagonismo a los fenómenos de autocatálisis y estigmergia potenciados por la diferencia entre las longitudes de los brazos del puente. Cabe señalar que si bien existe una preferencia clara por el camino más corto, el brazo más largo también fue utilizado, aunque en una proporción mínima, a pesar de la creciente influencia del rastro de feromona en el brazo más corto, cuya interpretación por parte de los investigadores tiene que ver con un *instinto exploratorio* presente en ciertas hormigas, es decir, independiente de cuan reforzado esté un camino, ciertos agentes continúan buscando alternativas que pueden eventualmente dar paso a una mejor solución.

Esto último dio paso para la generación de un tercer experimento, en el cual se explora qué sucede si luego de alcanzar la convergencia, se añade artificialmente un camino más corto entre el nido y la fuente de alimento.

Inicialmente se liberan las hormigas sobre un puente que sólo contiene el brazo largo, las hormigas viajan por éste y depositan rastros de feromona. Luego de 30 minutos, se acopla el brazo más corto, dejando el puente con los dos brazos. El comportamiento observado muestra que las hormigas siguieron viajando por el brazo más largo. Esto puede explicarse por la alta concentración de feromona en el brazo largo y el lento proceso de evaporación de la misma. En ese sentido, la aparición de una ruta más favorable no es percibida por la colonia dado el constante reforzamiento del brazo inicial. La evaporación de la feromona resulta ser bastante lenta, no permitiéndole a la colonia *olvidar* los caminos subóptimos encontrados y poder *descubrir* los nuevos y más cortos.

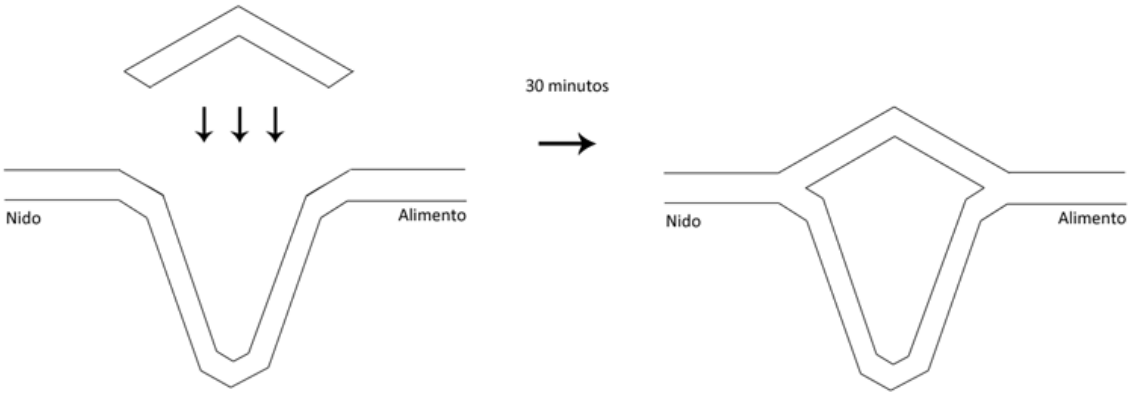

Figura 2.11: Esquema del tercer experimento de puente doble. Elaboración propia basado en [10].

2.3.3. Modelo Estocástico

En base a la evidencia empírica obtenida, se propuso un modelo estocástico que permitiera describir el comportamiento que la colonia de hormigas siguió durante el experimento del puente doble.

Siguiendo la notación de [10], sean ψ hormigas por segundo que cruzan el puente en cada dirección a una velocidad constante de v cm/s depositando una unidad de feromona. Sean l_s y l_l las longitudes del brazo corto y del largo respectivamente, una hormiga que elige el brazo corto atravesará el puente en $t_s = l_s/v$, mientras que si elige el brazo largo tardará $r * t_s$, donde $r = l_l/l_s$.

La probabilidad $p_{ia}(t)$ de que una hormiga al llegar al punto de bifurcación seleccione el brazo $a \in \{s, l\}$ en el instante t será una función relativa al total de feromona $\varphi_{ia}(t)$ contenida en el brazo, la cual será proporcional al número de hormigas que han utilizado tal brazo hasta el instante t. Así, por ejemplo, la probabilidad $p_{is}(t)$ de elegir el brazo más corto estaría dada por:

$$p_{is}(t) = \frac{(t_s + \varphi_{ia}(t))^\alpha}{(t_s + \varphi_{is}(t))^\alpha + (t_s + \varphi_{il}(t))^\alpha} \tag{2.20}$$

Para un modelo funcional, se estableció el valor del parámetro α en 2, de acuerdo con los resultados provenientes de experimentos [11].

Este modelo no considera la evaporación de la feromona y sólo asume que la cantidad de esta

sustancia química es proporcional al número de hormigas que utilizaron un determinado brazo del puente en el pasado. La razón para no incluir la evaporación de la feromona tiene que ver con los tiempos necesarios que para que se lleve a cabo tal proceso en una situación real.

La dinámica del sistema en términos de su evolución, puede ser modelada por las siguientes ecuaciones diferenciales:

$$\frac{d\varphi_{is}}{dt} = \psi p_{js}(t - t_s) + \psi p_{is}(t), (i = 1, j = 2; i = 2, j = 1) \quad (2.21)$$

$$\frac{d\varphi_{il}}{dt} = \psi p_{jl}(t - r * t_s) + \psi p_{il}(t), (i = 1, j = 2; i = 2, j = 1) \quad (2.22)$$

La primera ecuación se puede interpretar de la siguiente manera: la variación instantánea del nivel de feromona en el brazo s en el punto de decisión i está dada por el flujo de hormigas ψ multiplicado por la probabilidad de escoger el brazo s en el punto de decisión j en el instante $t - t_s$ más el flujo de las hormigas multiplicado por la probabilidad de escoger el brazo s en el punto de decisión i en el instante t. La constante t_s representa el tiempo necesario que le lleva a una hormiga atravesar el brazo s. La segunda ecuación es análoga para el caso del brazo l, es decir, el más largo.

Este modelo fue evaluado utilizando el método de Monte Carlo, a través de dos experimentos que consistieron en 1000 simulaciones cada uno y en donde la razón entre los brazos, fue establecida en $r = 1$ y $r = 2$ [11].

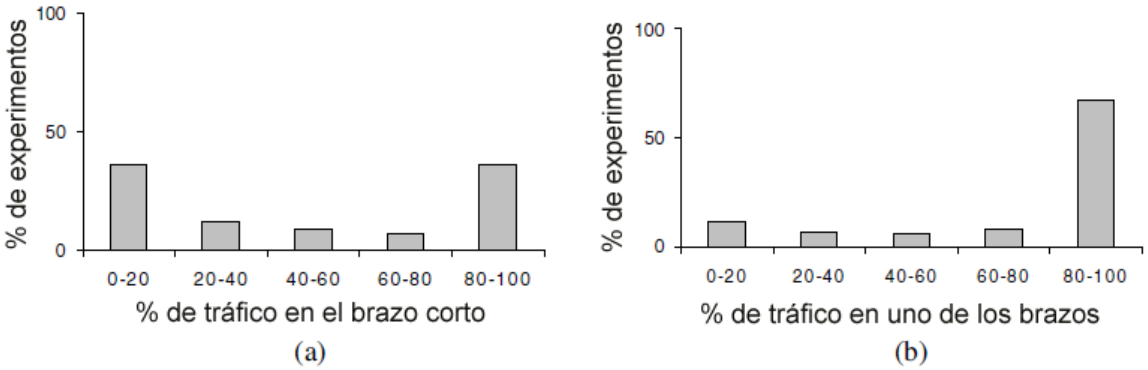

Figura 2.12: Resultado de la simulación a través del método de Monte Carlo del modelo estocástico. Elaboración propia basado en [11].

En la Figura 2.12 se muestran los resultados de ambos experimentos. Se puede observar que cuando los dos brazos tienen la misma longitud ($r = 1$), se converge al uso de cualquiera de los dos con igual probabilidad. Por el contrario, cuando uno de los brazos es el doble en longitud que el otro, se puede ver que en la mayoría de los casos las hormigas eligen el brazo más corto.

Cabe señalar que en este modelo las hormigas depositan feromona tanto en el viaje de ida como en el de regreso. Si se realiza el experimento permitiendo el depósito de feromona sólo en un sentido, el modelo no logra converger al uso predominante del brazo más corto. Algo similar ocurre con las colonias de hormiga reales: si solamente depositan feromona en su camino de regreso al nido, no serán capaces de encontrar el camino mínimo entre éste y la fuente de comida [9]. Luego, un reforzamiento doble parece ser necesario.

Posteriormente se modificaron ciertos elementos del modelo con el fin de caracterizar una hormiga artificial en su comportamiento dentro de un grafo. Se asumió el tiempo como una dimensión discreta y en cada instante la hormiga se mueve hacia el nodo vecino a una velocidad constante de una unidad de longitud por unidad de tiempo. Al hacer esto, cada hormiga adhiere una unidad de feromona a los arcos que utiliza. Las hormigas se mueven por el grafo eligiendo su nodo destino de manera probabilística, $p_{is}(t)$, es decir, la probabilidad que una hormiga situada en el nodo i en el instante t elija el camino más corto, y $p_{il}(t)$, la probabilidad de elegir el más largo.

2.3.4. La metaheurística de Optimización de Colonia de Hormiga

Los problemas de optimización combinatorial, han representado siempre un desafío ya que si bien su definición y caracterización suele ser simple, su solución es difícil. Muchos de estos problemas terminan siendo clasificados como NP-completos, es decir, se tiene casi certeza absoluta de que no pueden ser resueltos dentro de un tiempo polinomial. Una respuesta a esta problemática operacional ha sido la generación de métodos aproximados que entregan soluciones cercanas al óptimo, pero en un tiempo coherentemente reducido. Estos métodos reciben el nombre de *heurísticas*.

Una *metaheurística* es un conjunto de conceptualizaciones algorítmicas que pueden ser usadas para definir métodos heurísticos aplicables a un amplio rango de problemas. En ese sentido puede ser entendida como un metodología que busca guiar a una o más heurísticas a través de una región que les permita alcanzar soluciones de mejor calidad. En palabras simples, es un *framework* que puede ser adaptada y aplicada a diversos problemas de optimización. Ejemplos de metaheurísticas son : Simulated Annealing (Cerný, 1985), Búsqueda Tabú (Glover, 1989) y Computación Evolutiva (Holland, 1975).

ACO es una metaheurística en la cual un conjunto, o colonia, de hormigas artificiales cooperan entre sí para encontrar buenas soluciones a problemas de optimización discreta a través del uso de la estigmergia o comunicación indirecta a través de la modificación del entorno. A continuación se caracterizan los componentes arquetípicos que conforman la metaheurística.

La Representación del Problema

En ACO, una *hormiga artificial* es un procedimiento estocástico que incrementalmente va añadiendo componentes a una solución en construcción. De esta forma, se puede aplicar en cualquier

problema de optimización combinatorial en donde una o más heurísticas constructivas puedan ser definidas.

El principal desafío que conlleva la utilización del método en estudio es cómo modelar y adaptar el problema a atacar de tal forma que pueda ser factible como un ambiente propicio para las hormigas y para que éstas puedan generar soluciones coherentes.

Siguiendo la notación de [11] se considera un problema de minimización (S, f, Ω) , donde S es el *conjunto de soluciones candidatas*, f es la *función objetivo* la cual asigna un costo $f(s,t)$ a cada solución candidata $s \in S$, y $\Omega(t)$ es el *conjunto de restricciones*. El parámetro t indica que tanto la función objetivo como las restricciones pueden depender del tiempo (por ejemplo en problemas de optimización dinámica).

El objetivo es encontrar una solución factible $s*$ que sea *óptima global*, es decir, una solución que represente el costo mínimo para el problema de minimización en todos los casos posibles.

El problema de optimización combinatorial (S, f, Ω) debe ser transformado en una estructura que englobe los siguientes elementos:

- Un conjunto de *componentes* $C = c_1, c_2, ..., c_{N_C}$.
- Los *estados* del problema son definidos en términos de secuencias $x = \langle c_i, c_j, ..., c_h, ... \rangle$ de una longitud finita sobre los elementos de C. Este conjunto de todos los posibles estados se denota por X.
- El conjunto de soluciones candidatas S es subconjunto de X.
- Un conjunto de estados factibles $\bar{X}$, con $\tilde{X} \subseteq X$, tal que no sea imposible completar una secuencia $x \in \tilde{X}$ en una solución que satisfaga las restricciones Ω.
- Un conjunto no vacío de S^* de soluciones óptimas, donde $S^* \subseteq \tilde{X}$ y $S^* \subseteq S$.
- Un *costo* $g(s,t)$ asociado con cada solución candidata $s \in S$.
- En algunos casos el costo, o la estimación de un costo $J(x,t)$ podría ser asociada a los estados más que a las soluciones candidatas. Si el estado x_j puede ser obtenido añadiendo *componentes de solución* al estado x_i, entonces $J(x_i,t) \leq J(x_j,t)$.

Tomando esta formulación del problema, las hormigas artificiales construyen soluciones llevando a cabo caminos aleatorios a través de un grafo conexo $G_C = (C, L)$ cuyos nodos son el conjunto de los componentes anteriormente definidos, C, y L es el conjunto de las *conexiones* entre ellos.

El comportamiento de la Hormigas Artificiales

Tal como se señaló al comienzo de la sección, las hormigas artificiales son elementos estocásticos que construyen soluciones a través de sus movimientos dentro de un grafo G_C. Los componentes $c_i \in C$ y sus conexiones $l_{ij} \in L$ tienen asociados dos elementos principales:

- Un *rastro de feromona* τ (τ_i si está asociada a un componente, τ_{ij} si lo está a una conexión). Este elemento contiene un componente de memoria a largo plazo relativo al proceso completo de búsqueda y es actualizado iterativamente por las mismas hormigas artificiales.
- Un *valor heurístico* (en algunas publicaciones también llamado *información heurística*) η (η_i y η_{ij}, respectivamente) que contiene información asociada al problema, pero que no es provista ni modificada por las hormigas, sino que proviene de una fuente externa y ha sido introducida a priori. Comúnmente, en este elemento se almacena el valor del costo (o una estimación de éste) de añadir un componente c_i o su conexión asociada a la solución que se está construyendo. Luego, este valor es utilizado en el proceso probabilístico de toma de decisiones que lleva a cabo cada hormiga en su camino a través del grafo.

Siguiendo textualmente el esquema del libro de Marco Dorigo [11], cada hormiga k perteneciente a la colonia debe tener las siguientes propiedades:

- Utiliza un grafo de construcción $G_C = (C, L)$ para encontrar soluciones óptimas $s^* \in S^*$.
- Posee una memoria $\mathcal{M}^k$ que puede ser utilizada para almacenar información sobre la ruta (solución) construida hasta el momento. Más específicamente, sus funciones serían:
  - Construcción de las soluciones factibles (internalizando el conjunto de restricciones Ω).
  - Reconocer e integrar los valores heurísticos η.
  - Evaluar las soluciones encontradas.
  - Permitir a la hormiga recorrer el camino generado. Por ejemplo, bajo una configuración nido-fuente de alimento, permitir volver al nido una vez alcanzada la fuente de alimento.
- Posee un *estado inicial* ${x_s}^k$ y una o más *condiciones de término* e^k. Usualmente, el estado inicial se expresa tanto como una secuencia vacía o como una que posee un sólo elemento.
- Cuando está en el estado $x_r = \langle x_{r-1}, i \rangle$, y la condición de término no ha sido satisfecha, la hormiga se moverá a un nodo j perteneciente a su *vecindario* $\mathcal{N}^k(x_r)$, es decir, hacia el estado $\langle x_r, j \rangle \in \mathcal{X}$. Si al menos una de las condiciones de término se cumple, la hormiga deberá detenerse.
- Selecciona un movimiento a través de la aplicación de una regla de decisión probabilística, la cual se define en función de:

  - Los trazos de feromona disponibles y los valores heurísticos asociados.
  - La memoria privada de cada hormiga que almacena su estado actual.
  - Las restricciones del problema.

- Cuando se añade un componente c_j al estado actual, se puede actualizar el trazo de feromona τ asociado a éste o sus conexiones correspondientes.
- Una vez que se ha construido una solución, se puede recorrer el camino en sentido contrario y actualizar los trazos de feromona de los componentes utilizados.

Estas propiedades constituyen la estructura operativa que permite llevar a cabo la búsqueda global de soluciones al problema considerado en base a la interacción colectiva de las hormigas a través de la comunicación indirecta que subyace en la utilización y modificación concurrente de los trazos de feromona.

Esto representa un proceso de *aprendizaje distribuido* en donde los agentes involucrados van modificando la forma en la cual el problema es percibido por los demás.

La Metaheurística

La literatura ha clasificado los procedimientos existentes dentro de la metaheurística en tres grupos principales [11]:

- **Construct Ants Solutions** : Este conjunto agrupa todas las funciones relativas a la generación de los caminos que conllevan a la construcción de la solución.
- **Update Pheromones** : Agrupa los procesos que gestionan la forma en la cual la feromona es actualizada, lo cual se lleva a cabo mediante las siguientes acciones:
  - *depositar feromona*, que intuitivamente aumenta la probabilidad de que los componentes o conexiones asociadas a ella sean utilizados por las demás hormigas.
  - *evaporar feromona*, que internaliza la acción de *olvidar* componentes o conexiones que no representan una buena solución. En ese sentido, evita la convergencia del algoritmo en regiones suboptimas.
- **Daemon Actions** : Son el conjunto de acciones que se llevan a cabo de manera centralizada y que no pueden ser efectuadas por las hormigas. Ejemplos de este tipo de procedimientos:
  - Activación de procesos de optimización local en ciertas áreas.
  - Recopilación de información global o del estado de la solución parcial para decidir sobre la proporción en que la feromona es depositada y/o evaporada.

- Inhabilitación arbitraria de ciertos agentes que, dado el comportamiento del algoritmo, se apartaron de las regiones factibles.

- **Schedule Activities** : Son métodos que engloban a los anteriores y que tienen como función sincronizar y adaptar las secuencias de funcionamiento.

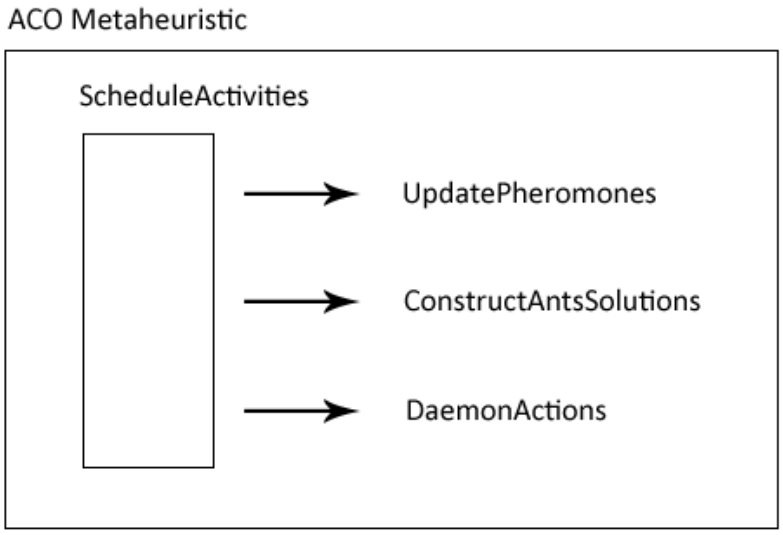

Figura 2.13: Esquema de los componentes de la metaheurística. Elaboración Propia.

2.3.5. Implementación de la metaheurística en problemas estándar

Con el fin de ejemplificar cómo la metaheurística en estudio ha sido utilizada y adaptada para resolver diversos problemas de optimización, se presentan algunos ejemplos concretos.

El Problema del Vendedor Viajero

Es un problema de optimización combinatorial del tipo NP-hard y ha concentrado bastante interés entre los investigadores a lo largo del tiempo. Tiene una connotación especial para este estudio debido a que fue el problema elegido para probar el primer algoritmo de ACO, Ant System, desarrollado por Marco Dorigo en 1992 y ha sido utilizado para testear la gran mayoría de algoritmos que han sido diseñados posteriormente. Formalmente este es el problema que enfrenta un vendedor viajero quien, comenzando desde su ciudad de origen, debe encontrar el viaje más corto posible a través de un conjunto de ciudades determinado, visitando cada ciudad una vez y luego volver a su

ciudad de origen. El TSP (Travelling Salesman Problem) se representa a través de una grafo con pesos (o costos) en todos su arcos $G = (N, A)$, donde N es el conjunto de nodos $n = |N|$ (ciudades) y A el conjunto de arcos que conectan completamente a los nodos. A cada arco (i, j) se le asigna un peso d_{ij} que representa la distancia entre las ciudades i y j. El TSP consiste en encontrar el circuito hamiltoniano de menor longitud presente en el grafo. Un circuito hamiltoniano se define como un camino cerrado dentro del un grafo G tal que cada nodo se visita exactamente una sola vez.

- Representación del grafo: El conjunto de componentes C corresponde a los nodos, las conexiones a los arcos. Y cada conexión tiene un peso que está asociado a la distancia d_{ij} entre los nodos i e j. Los estados del problema son el conjunto de posibles caminos cerrados (*tours*) parciales.
- Restricciones: En este caso se debe cumplir que todas las ciudades deben ser visitadas y que cada ciudad debe ser visitada a lo más una vez. De esta forma cada hormiga en cada paso debe elegir entre las ciudades que no ha visitado anteriormente.
- Trazos de feromona e información heurística: Los trazos de feromona τ_{ij} representan la *conveniencia* (en términos de encontrar una buena solución) de viajar a la ciudad j desde la ciudad i . Por otro lado la información heurística η_{ij} es un valor inversamente proporcional a la distancia existente en las ciudades i y j. Usualmente se utiliza de la forma $\eta_{ij} = 1/d_{ij}$.
- Construcción de la solución : Cada hormiga se coloca en un nodo inicial elegido aleatoriamente y en cada paso va añadiendo a su ruta una ciudad no visitada anteriormente. La construcción de la solución termina cuando todas las ciudades han sido visitadas.

Problema de Ruteo en Redes

Se define una red de telecomunicaciones por un conjunto de nodos N y un conjunto de enlaces L_{net}. También se definen los costos d_{ij} asociados al uso de los enlaces. De esta forma, el problema de ruteo en redes (NRP por sus siglas en inglés) consiste en encontrar los caminos con menor costo entre todos los pares de nodos en la red.

- Representación del grafo: Se tiene un grafo $G_c = (C, L)$, donde C es el conjunto de nodos N y L el conjunto de enlaces. Se debe notar que $L_{net} \subseteq L$.
- Restricciones: Las hormigas sólo pueden usar los enlaces $l_{ij} \in L_{net}$.
- Trazos de feromona e información heurística: Cada enlace $l_{ij} \in L$ debe tener asociado diversos trazos de feromona, tantos como el número de rutas facibles que utilice tal enlace. Con respecto a la información heurística, η_{ij}, debe ser modelada como un valor inversamente proporcional a la cantidad de tráfico existente en el enlace.

- Construcción de la solución: Cada hormiga parte de un nodo inicial s y tiene como misión llegar hasta un nodo final d. Para esto utiliza las reglas de decisión probabilística dados los niveles de feromona y la información heurística.

Multidimensional Knapsack Problem

Dado un conjunto de items $i \in I$ asociados a un vector de requerimientos de recursos r_i y a un vector de beneficios b_i, el Problema de la Mochila o *Knapsack Problem* (KP) consiste en seleccionar un subconjunto de los items de I de tal forma que encajen en una mochila de capacidad limitada y se maximice la suma de los beneficios asociados la los items seleccionados. El problema de la mochila multidimensional, *multidimensional knapsack problem* (MKP) es una extensión de KP en el sentido de que considera múltiples restricciones para las recursos. Sea y_i la variable asociada con el item i y que toma el valor 1 si i fue añadido a la mochila , y 0 si no. Adicionalmente se define r_{ij} como el requerimiento de recursos del item i con respecto a la restricción de recurso j. Sea a_j la capacidad del recurso j y m el número de restricciones de recursos. De esta forma le problema puede ser formulado de la siguiente forma:

$$\begin{aligned} \text{maximizar} \quad & f(y) = \sum_{i=1}^{n} b_i y_i \\ \text{sujeto a} \quad & \sum_{i=1}^{n} r_{ij} y_i \leq a_i, \ j = 1, \ldots, m, \\ & y_i \in \{0, 1\}, \ i = 1, \ldots, n \end{aligned} \tag{2.23}$$

- Representación del grafo: En el grafo asociado $G_C = (C, L)$ el conjunto de componentes C corresponde al conjunto de items y L el conjunto de conexiones entre los items. El beneficio de añadir items puede ser asociada tanto a las conexiones como a los componentes.
- Restricciones: La principal restricción tiene que ver con que no se pueden violar los límites que impone la variable de capacidad a_j.
- Trazos de feromona e información heurística: En este caso se modela asociando los trazos de feromona τ_i con los componentes, y el valor que toma es proporcional a la factibilidad de añadir el elemento i a la solución actual. Asimismo, la información heurística deberá ser modelada de tal forma que privilegie los items que entregan un alto beneficio y que al mismo tiempo tienen bajos costos de alocación. Una representación podría ser el cálculo del cuociente entre el beneficio asociado al item y el requerimiento promedio dadas las m restricciones.
- Construcción de la solución: Cada hormiga va añadiendo items en función de los niveles de feromona que cada uno de éstos posee. Cada item se puede ser utilizado sólo una vez y el algoritmo termina cuando, para añadir un nuevo item, necesariamente se deben violar las restricciones de capacidad.

2.3.6. ACO y Web Intelligence

La implementación de ACO en tareas propias de Web Intelligence ha dejado experiencias notables tanto a nivel de la creatividad plasmada en su adaptación como también a los resultados obtenidos. A continuación se explora el estado del arte a través de una revisión bibliográfica.

Reorganización automática de un sitio web

La tarea del rediseño y reorganización del un sitio web ha incrementado su complejidad al pasar el tiempo dado el crecimiento de Internet y la heterogeneización de los usuarios, para los cuales un cambio específico puede tener diferentes repercusiones [25]. Respondiendo a tal problema, Perkowitz y Eizioni [25] definieron el término *adaptive websites*, es decir, sitios web adaptables, como los sitios que automáticamente mejoraban su organización y presentación en función del aprendizaje que obtenían de los patrones de navegación de los usuarios. En ese sentido, la *eficiencia* de un sitio web podría ser mejorada en función de dos directrices:

- Personalización: Consiste en guiar al usuario entregándole de forma coherente información adicional que le ayudará a tomar mejores decisiones.
- Transformación: Consiste en realizar un reordenamiento de los links entre las páginas que conforman el sitio web para facilitar la navegación.

Uno de los enfoques estudiados hasta el momento lo propuso Lin [18], en donde el problema de la reorganización estructural de un sitio se presenta como un problema especial de optimización en grafos. Utilizando programación lineal, específicamente programación lineal binaria, propuso modelos que toman como principal parámetro la cohesión entre las páginas componentes con el fin de reducir la sobrecarga de información. Si bien se propone adicionalmente un enfoque heurístico, los tiempos asociados al proceso de solución son demasiado al altos, más aún cuando el número de links es considerable.

Como solución a esa problemática, Lin y Tseng propusieron un enfoque de resolución basado en la utilización de ACO [19]. Los resultados muestran que si bien en simulaciones en sitios de mediano tamaño los dos enfoques retornaban soluciones similares, al aumentar progresivamente la cantidad de links, la solución asociada al enfoque de programación lineal binaria perdía calidad y tomaba más tiempo, mientras que el enfoque que utilizaba ACO tenía un desempeño mejor [19].

Descubrimiento de los patrones de navegación

Ling et. al [20] proponen un método basado en ACO para descubrir los patrones de navegación más frecuentes dentro de un sitio web.

En primer lugar establecen un modelo simple siguiendo la representación convencional de un sitio web en función de un grafo $G = (N, E)$ donde N representa el conjunto de nodos o páginas presentes, y E el conjunto de los links entre los nodos.

Posteriormente se define una medida de soporte η_{ij} como :

$$\eta_{ij} = C_{ij}/((\sum_{k=1}^{n} C_{ik} y_i)/n) \tag{2.24}$$

Donde C_{ij} es el frecuencia de transiciones reales desde el nodo i al nodo j. Asimismo se define la función de feromona $\tau_{ij(t)}$:

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \Delta\tau_{ij}, \tag{2.25}$$

donde $\Delta\tau_{ij} = Q\eta_{ij}$, con Q constante y ρ un factor de atenuación.

Por último se define la función de preferencia $p_{ij}(t)$:

$$p_{ij}(t) = \frac{[\tau_{ij}(t)]^{\alpha} * [\eta_{ij}]^{\beta}}{\sum_{k=1}^{n} [\tau_{ik}(t)]^{\alpha} * [\eta_{ik}]^{\beta}}, \tag{2.26}$$

donde α y β son las ponderaciones tanto de la feromona como del soporte, respectivamente.

La interacción de todos estos componentes se refleja en un algoritmo que iterativamente actualiza los depósitos de feromona en el grafo en función de la frecuencia real de transición entre los nodos. Finalmente los patrones de navegación pueden ser extraídos seleccionando los valores más altos de $p_{ij}(t)$.

Otro enfoque para el descubrimiento de patrones de navegación es propuesto por Abraham y Ramos [28] basados en el clustering de sesiones. Como inspiración se toma la capacidad que presentan ciertos tipos de hormiga de poder agrupar cadáveres en el nido con el fin de mantener el orden y las vías de evacuación operativas. A este agrupamiento se le llama comúnmente *cementerios de hormigas* [6] dado que los elementos agrupados son cadáveres, aunque también se registra el fenómeno para agrupar crías. El funcionamiento de estos cementerios se basa en un proceso autocatalítico en donde las hormigas recogen elementos aislados (cadáveres o crías) y son depositados en un lugar donde se encuentran otros del mismo tipo. Esto va generando grupos o *clusters* lo cual, a medida que crecen, refuerzan el atributo que relaciona a sus componentes.

El primer modelo que intentó explicar este fenómeno desde un enfoque biológico fue formulado por Deneubourg et al. [9]. Posteriormente fue generalizado y utilizado para el análisis de datos

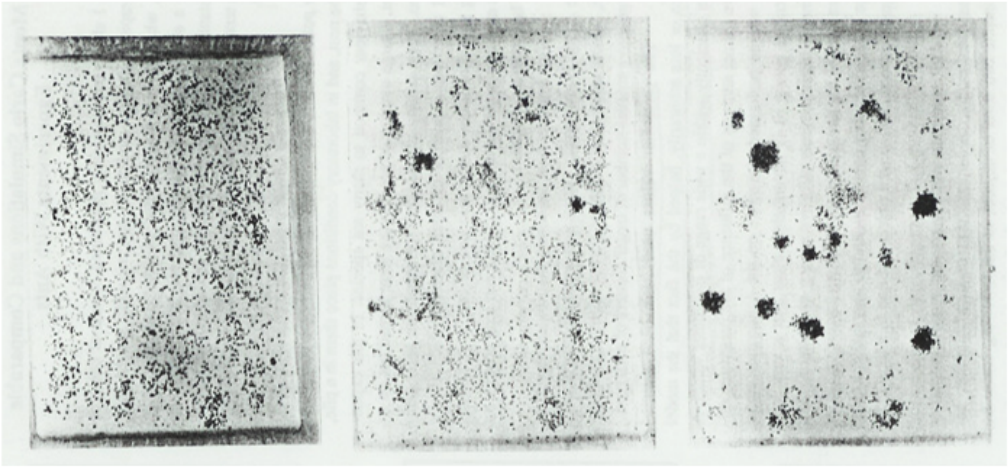

Figura 2.14: Evolución de un agrupamiento de cadáveres en una colonia de hormigas. Las fotos representan la disposición a las 0 , 20 y 68 horas respectivamente. Tomado de [9].

por Lumer and Faieta [23]. Un enfoque más apegado al comportamiento real de las hormigas fue propuesto por Ramos et al. [29], llamado ANTCLUSTER, en el cual se incluyen probabilidades de transición espacial en lugar de movimientos aleatorios, como también se suprime el uso de memorias individuales de corto plazo y se introduce una estructura de comunicación indirecta entre todos los agentes. Los factores que gobiernan el actuar de los agentes agrupadores son el número y el tipo de elementos con los que se enfrenta. La hormiga que recoge un cadáver recorre el terreno hasta encontrar indicios de cadáveres del mismo tipo. Si el número de cadáveres con los que se encuentra supera un umbral determinado, entonces deposita su carga. En [1] se utiliza el algoritmo ANTCLUSTER en una estrategia mixta que utiliza ACO para clusterizar las sesiones y Algoritmos Genéticos para analizar los patrones de navegación. Los experimentos incorporaron la comparación del método mencionado con otras estrategias. Los resultados muestran que ANTCLUSTER tiene un mejor rendimiento que un método basado en Mapas Autorganizativos, pero no es tan eficiente al compararlo con un enfoque basado en Evolutionary Fuzzy-Clustering.

Elección en tiempo real de publicidad basada en preferencias de contenido

White et al. [42] proponen un interesante enfoque basado en ACO para la elaboración de una metodología que permita la elección del contenido de la publicidad que se mostrará a un determinado usuario, con el fin de maximizar la probabilidad de que éste haga click en el aviso, lo cual generará la respectiva ganancia monetaria para el sitio. En primer lugar se configura un entorno de simulación como el se puede ver en la Figura 2.15.

En tal configuración, un usuario que posee una preferencia por contenido dentro del sitio y realiza una petición al servidor web (1), el cual retorna la página solicitada que a su vez contiene una

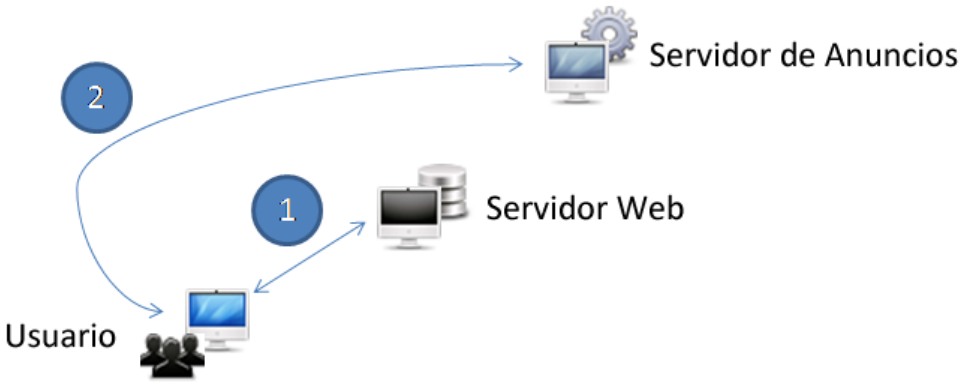

Figura 2.15: Simulación paper White et al. Elaborado en base a [42].

función especial insertada como código Javascript. Esta función se ejecuta dentro del navegador del usuario y tiene como misión extraer las keywords desde el contenido para luego enviarlas al servidor de anuncios (2), el cual decide qué anuncio enviar a la página que el usuario está visitando.

En primer lugar se generan l usuarios. A cada uno se le asigna una preferencia por un conjunto de palabras m ,lo cual se ve plasmado en la utilización de un vector de preferencias $userInterest = [k_0, k_1, ..., k_{m-1}]$, donde cada elemento k_i está entre 0 (que representa nulo interés de parte del usuario por tal palabra) y 1 (que representa total interés del usuario por la palabra). Los valores son generados aleatoriamente, pero forzando una atenuación en los extremos. Del mismo modo se define cada página del sitio como un vector $page = [r_0, r_1, ..., r_{m-1}]$ donde cada elemento r_i representa la probabilidad de existencia de una keyword y está entre 0 (no contiene la palabra) y 1 (contiene la palabra exactamente). Estos valores son continuos, ya que son sólo una aproximación puesto que no se contempla el uso del enfoque del tipo Vector Space Model [42].

La similitud entre un vector de preferencia y una página se calcula a través ángulo generado entre los vectores :

$$angle(a, b) \quad = \quad \arccos\left(\frac{\sum_{i=0}^{m} a_i * b_i}{||a||||b||}\right) \tag{2.27}$$

Cada aviso publicitario se toma como un elemento que contiene pm keywords, de las m posibles ($pm < m$). Mientras más interés tenga el usuario en las pm keywords de un determinado anuncio, mayor será la probabilidad de hacer click en él. Esta probabilidad se determina en base a una escala logarítmica. Adicionalmente se se asocia un vector $adKeywords$ de longitud m y cuyos valores pueden tomar el valor 1 si la keyword asociada está en el conjunto de las pm presentes en el anuncio, 0 si no.

Dentro del modelo, cada anuncio se representa como un camino fijo de m nodos. Como hay n anuncios disponibles , se define un contenedor $M = \{v_{01}, v_{02}, ..., v_{0m-1}, v_{11}, ..., v_{n-1m-1}\}$, donde cada v_{ij} toma un valor inicial τ_0 que representa el nivel de feromona inicial.

Cuando el servidor recibe la petición por parte del usuario, se ejecuta el algoritmo que debe determinar cual de los n anuncios mostrar. Este procedimiento comienza con el cálculo del ángulo entre cada vector de anuncio y el de la página solicitada. Posteriormente el algoritmo espera que el anuncio enviado sea visualizado por el usuario. Si el resultado fue exitoso, es decir, el usuario hizo click en el anuncio, se ejecuta la segunda parte del algoritmo, que consiste en rescatar el índice del vector asociado al anuncio exitoso para reforzar sus niveles de feromona.

Para efectuar la simulación se consideró la comparación entre el modelo y una selección aleatoria de anuncios dado una petición al servidor. Dada esta configuración, en el instante correspondiente a la petición número 500.000 al servidor, el método basado en ACO superaba en un 80 % al método aleatorio.

Capítulo 3

Diseño del modelo

En este capítulo se aborda la construcción del modelo para simular las preferencias por texto del usuario en la Web a través de ACO. En primer lugar se muestra el desarrollo de una Prueba de Concepto, la cual permitió establecer una hoja de ruta para el Modelo Final, el cual se presenta en la segunda parte del capítulo.

3.1. Prueba de Concepto

Tiene como objetivo demostrar la factibilidad de un modelo. En ese sentido, se construye una representación cuantitativa de la idea y se le somete a pruebas experimentales con el fin de evaluar los resultados, los cuales sirven de indicadores para los pasos a seguir en términos de los cambios o adiciones que se deben realizar.

Esta primera aproximación se basa en una transformación de la formulación tradicional de la metaheurística de ACO en una dimensión compatible con los conceptos de Web y Usuario Web. En ese sentido, se procedió a modelar el Ambiente Web, es decir, el análogo al terreno que recorren las hormigas reales. Posteriormente se modelaron los Usuarios Web, representados por las hormigas artificiales con quienes se definió la optimalidad en base a una utilidad de información acumulada. Por último, la interacción entre las entidades anteriormente señaladas en relación a cómo las hormigas modifican el ambiente para crear las rutas óptimas.

3.1.1. Modelación del Ambiente Web

En primer lugar se modela un sitio web como un un grafo G con N nodos que representan los documentos que lo componen y E arcos que representan los enlaces o *links* existentes entre

los documentos. Esta representación ha mostrado factibilidad coherente en términos del estudio y análisis estructural de la web [4, 27]. Sobre esta estructura las hormigas artificiales generarán sus rutas.

En segundo lugar, se requiere un elemento que sea el análogo a la feromona en la formulación inicial del problema de optimización que aborda la metaheurística. En este caso se debe encontrar una propiedad que se genere a partir de los componentes del grafo y que tenga la cualidad de ser modificable a través del tiempo por parte de las hormigas artificiales.

Una propuesta para generar la *feromona web* puede ser extraída a través de la utilización del modelo de comportamiento de usuario Random Surfer[8], en la cual se supone que los usuarios se mueven aleatoriamente a través del sitio.

Este modelo sigue una regla de 80-20 para cubrir los posibles movimientos: Supone que el 80 % del tiempo, el usuario sigue vínculos dentro del sitio de manera aleatoria, y el 20 % restante, simplemente escoge una página al azar o simplemente abandona el sitio.

Este modelo deja ver algunas debilidades dada su simpleza:

- En la realidad existe la equiprobabilidad en cuanto a la elección de los links.
- No siempre se puede ir directamente desde una página a otra dentro de un sitio web.
- La regla 80-20 está fija permanentemente.
- No se toma en cuenta el uso de herramientas propias del los navegadores web como son el *Back button* ni el acceso vía marcadores (*Bookmarks*).

A pesar de las debilidades mencionadas, el modelo ha sido una base sólida para el estudio del comportamiento del usuario en la web y ha sido utilizado en numerosas investigaciones con éxito[16].

A partir de la regla de 80-20, se genera una matriz de transición que, tomando en cuenta la estructura del sitio web, en términos del número de documentos y la forma en la que están enlazados, genera una probabilidad inicial de pasar desde un determinado documento a otro según la siguiente expresión:

$$p_{ij} = \frac{p * g_{ij}}{c_j} + \frac{1-p}{n} \tag{3.1}$$

Donde n es el número de nodos, p es el valor de la probabilidad de continuar navegando dentro del sitio (regla 80-20). La matriz de adyacencia está representada por g_{ij}, esta expresión

toma el valor 1 si existe un enlace entre el documento i y el j, 0 de otro modo. c_j corresponde al número de enlaces que salen desde el nodo i.

El valor de la probabilidad p_{ij} será el utilizado como un valor inicial de la feromona existente en los enlaces y que será modificado por las hormigas artificiales para construir los caminos:

$$p_{ij} \rightarrow \tau_{ij} \tag{3.2}$$

Con respecto a los nodos, cada uno posee un vector L_i que representa los valores TF-IDF asociados a sus palabras clave o *keywords*. El largo de los vectores se ha definido arbitrariamente.

3.1.2. Caracterización del las hormigas artificiales

El principal componente de una hormiga artificial en este modelo es el vector de preferencias por texto μ. Este vector es generado de manera aleatoria,y simula los valores de TF-IDF asociados a sitio web.

En primer lugar se genera un repositorio de vectores μ de la siguiente forma:

- Inicializar repositorio R.
- Asignar a cada componente de un vector μ un valor aleatorio definido por mediante una distribución Uniforme(0,100).
- Normalizar el vector resultante.
- Calcular el producto punto entre el vector generado μ y cada vector μ' que ya haya sido insertado en R. Si el valor obtenido es siempre menor que un límite d, insertar en el repositorio R. Si no, eliminar.
- Continuar el procedimiento hasta llegar a un número predeterminado de vectores.

Posteriormente, cada uno de los vectores generados es asignado a una hormiga artificial, la cual es dejada en el grafo para su posterior exploración y modificación de la feromona.

El largo de los vectores μ debe ser igual al de los vectores de los nodos.

3.1.3. Modificación de la feromona

Sea i el identificador de una hormiga. Una vez que recibe su vector de preferencias por texto,μ_i, comienza su viaje a través de grafo.

Cada vez que llega a un nuevo nodo k, la hormiga i calcula la utilidad V_k de haber llegado a éste, a través de la siguiente expresión:

$$V_k = \frac{\mu_i * L_k}{\|\mu_i\| \, \| \, L_k\|} \tag{3.3}$$

Este valor de utilidad V_k es utilizado para dos fines :

- En primer lugar, el valor de la utilidad calculada sirve para actualizar el nivel de *saciedad de información* que tiene la hormiga, el cual, si llega a un límite definido, hará que la hormiga detenga su viaje.
- En segundo lugar, el valor de V_i será utilizado para modificar la probabilidad de transición desde el nodo anterior al actual. La modificación de la probabilidad señalada se realizará en forma proporcional a la probabilidad Logit asociada :

$$\Delta p_{k-1k} = \frac{e^{V_k}}{\sum_1^j e^{V_j}} \tag{3.4}$$

Donde j representa los vecinos del nodo $k-1$.

Esto se basa en que cuando el usuario web está decidiendo si optar o no por una determinada página, está enfrentando un problema de elección discreta.[31, 33]

Por último, al final de cada ciclo, se llama una función de *evaporación*, la cual permite disminuir los niveles de feromona existentes en el grafo, lo que ayuda a una rápida convergencia del algoritmo en términos de la definición de los caminos más recurrentes dado un determinado vector del preferencias por texto μ_i.

3.1.4. Obtención del vector de preferencias por texto μ

Un ves terminado el funcionamiento del algoritmo, se tendrá una nueva matriz de transición. El estado actual del problema puede ser representado de la siguiente forma:

$$P(l \to k) = \sum_{\mu} P(\mu) \cdot P(l \to k|\mu) \tag{3.5}$$

Donde $P(l \to k)$ representa la probabilidad de pasar desde el nodo l al nodo j, valor que puede ser obtenido a través del cálculo empírico de la frecuencia de saltos entre páginas a través del análisis de los web logs.

$P(l \rightarrow k|\mu)$ representa la probabilidad de salto dado un determinado vector de preferencias por texto μ, el cual se obtiene desde la matriz de transición modificada durante el desarrollo del algoritmo. Por último, está la expresión $P(\mu)$ que representa la distribución de los vectores μ y que es justamente la incógnita que pretende encontrar el modelo. Una solución puede ser obtenida a través de regresión lineal, debido a que los demás elementos de la ecuación son conocidos.

De esta forma se obtiene la distribución de las preferencias por texto $P(\mu)$. De esta distribución se pueden obtener los valor mayores de μ y de éste sus keywords, las cuales representarán las mayores preferencias de los usuarios.

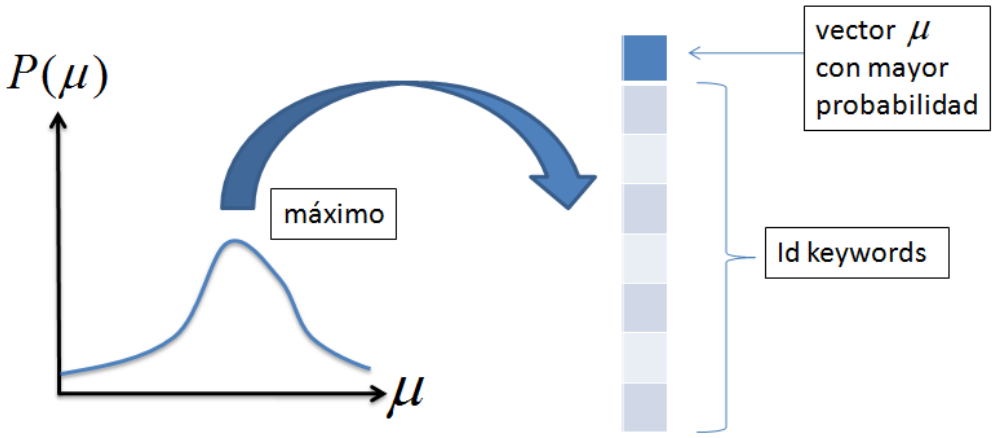

Figura 3.1: Esquema obtención de preferencias por texto. Elaboración propia.

3.1.5. Validación experimental de la prueba de concepto

Se realizó una prueba experimental utilizando el sitio del Departamento de Ingeniería Industrial[1].

Se realizó un procedimiento de limpieza de los archivos web logs y posteriormente un trabajo de sesionalización tradicional. Además, para recoger el contenido y estructura del sitio, se utilizó el web crawler WebSphinx[2]. Todos los datos fueron almacenados en un base de datos MySQL[3] siguiendo el paradigma de Vector Space Model.

Para el experimento se usaron:

- Nodos : 602

[1] http://www.dii.uchile.cl
[2] http://www.cs.cmu.edu/ rcm/websphinx
[3] http://www.mysql.com

- Arcos : 4386
- Número de hormigas utilizadas:4386
- Iteraciones por hormiga : 1000
- Largo vector μ : 1219, basado en rango de valores TF-IDF entre 0.2 y 0.9

El algoritmo fue desarrollado en lenguaje Java utilizando la librería MASON[4], especializada en problemas multiagente[22], como también la librería COLT[5] para los cálculos matemáticos.

El procedimiento fue ejecutado en un computador con procesador Intel Core 2 Duo 1.6 Ghz, 3 Gb RAM y el tiempo tomado fue aproximadamente 6 horas.

Los resultados obtenidos muestran que los vectores μ con mayor probabilidad tienen asociadas las keywords más comunes presentes en el sitio. A continuación se listan los principales vectores y sus keywords más relevantes en función de su valor de TF-IDF:

- Vector 1: *Manag (Management)*, *ingenieri (ingeniería)*, *pregrad (pregrado)*, *empres(empresa)*.
- Vector 2: *Mgpp*, *descripcion*, *Horario*, *Patrici (Patricio)*.
- Vector 3: *Capit (Capital)*, *susten (sustentabilidad)*, *Merc (Mercado)*.

  Cabe señalar que las palabras han sido procesadas a través de stemming, por lo que se representan a través de su raíz.

Como se puede apreciar,el primer vector, es decir, el con mayor probabilidad de representar las preferencias de los usuarios, contiene keywords comúnmente relacionadas con los conceptos principales del Departamento de Ingeniería Industrial. En segundo lugar se encuentra un vector cuyos elementos hacen referencia al programa de Magíster en Políticas Públicas, que posee un subsitio[6] dentro del sitio principal del DII. Lo interesante es que si se revisan las estadísticas de visitas generales, el sitio del Magíster en Políticas Públicas obtiene también el segundo lugar, superados solamente por páginas descriptivas de la carrera y del departamento (asociadas al primer vector). De esta forma existe una relación entre los flujos de visitas totales y los vectores encontrados por el modelo de prueba.

Los elementos del tercer vector pueden ser relacionados con conceptos económicos, pero no es posible asociarles a priori un conjunto de páginas afín.

[4]http://cs.gmu.edu/ eclab/projects/mason/
[5]http://acs.lbl.gov/software/colt/
[6]http://www.dii.uchile.cl/ webmgpp/

Luego, el primer experimento realizado, sin bien no permite bajo ningún punto de vista concluir sobre las preferencias por texto de los usuarios, si da la posibilidad de aventurar hacia donde de debe dirigir el trabajo en términos de la construcción de un modelo mejor.

Las conclusiones que se pueden sacar son las siguientes :

- El modelo es rígido a la hora de la generación de los vectores μ, los cuales se generan arbitrariamente y se mantienen fijos durante todo el desarrollo de algoritmo.
- Los valores asociados a los componentes de los vectores μ no tienen relación con la realidad. Lo recomendable sería añadir un ponderación en relación con el mínimo y máximo que pueden alcanzar.
- La incorporación de la función de evaporación ha permitido una convergencia efectiva del algoritmo en un tiempo coherente.
- Es necesario incluir en el modelo no sólo la estructura del sitio y el contenido de las páginas, sino también el comportamiento de los usuarios.
- Los vectores de preferencias por texto deben tener una componente dinámica, bajo el supuesto que los usuarios a medida que navegan por el sitio obtienen una ganancia en términos del ajuste entre sus preferencias y lo que visualizan en el sitio.
- Se debe encontrar un mejor medio de visualización de las keywords encontradas, en el sentido de generar una agrupación objetiva que permita establecer una relación que facilite la obtención de la *preferencia*.

En la siguiente sección se expondrá una nueva versión del algoritmo que incorpora nuevas funcionalidades para intentar generar una simulación más coherente. El nuevo algoritmo será discutido y revisado para su posterior implementación en un sitio real, tópico que se tratará en el siguiente capítulo.

3.2. Modelo de comportamiento de usuario con aprendizaje

La elaboración de la Prueba de Concepto permitió generar una base operativa sobre la aplicación de ACO para simular un ambiente Web. El propósito principal de la segunda versión del algoritmo, es incluir el concepto de aprendizaje en la hormigas artificiales, en el sentido de modificar progresivamente el vector de preferencia por texto μ con el fin de generar un ajuste entre las sesiones reales y las artificiales que se van construyendo. Este proceso se alimenta tanto de los registros de las sesiones reales que proporcionan los archivos web logs y su correspondiente sesionalización, como también de la estructura y contenido del sitio web en estudio. Se propone clusterizar lassesiones para reducir el número de comparaciones y facilitar la visualización de patrones de navegación. Se propone

realizar un entrenamiento del modelo y posteriormente una validación. Ambos procedimientos en base a datos reales provenientes de un sitio web. El proceso de entrenamiento tendrá como input las sesiones reales de usuario y la estructura y contenido de un sitio. Así, las hormigas serán entrenadas y el output será un conjunto de vectores de preferencia por texto μ. Éstos a su vez servirán como input para el proceso de validación, que consistirá en comparar las sesiones reales de usuario con las generadas por grupos de hormigas a las cuales les fueron entregadas los vectores señalados. De esta forma, el output del proceso de validación será un indicador del nivel de similitud entre los comportamientos artificiales y reales.

3.2.1. Clustering de sesiones reales

Motivación

El algoritmo que llevará a cabo el aprendizaje se basa en las modificaciones del vector de preferencias por texto μ de cada hormiga. Tal modificación está condicionada al resultado de las comparaciones que se van realizando entre la sesión artificial en construcción que lleva la hormiga y el conjunto de las sesiones reales. Una primera aproximación sería llevar a cabo una comparación entre la sesión artificial y todas las sesiones reales en cada iteración. La factibilidad de tal idea se pone en peligro cuando se toman en cuenta las magnitudes asociadas a los datos. Por ejemplo, inspecciones preliminares en los datos muestran un promedio de 25.000 sesiones mensuales. Adicionalmente el número de hormigas que se deben inicializar para asegurar una correcta convergencia del algoritmo como también el número de iteraciones sobre cada una de ellas, proporcionan una idea de que número de comparaciones que deberían realizarse es significativamente alta.

Por otro lado, como se busca simular el comportamiento general de los usuarios, la acción de agrupar y consolidar patrones de navegación proporciona una mejor visión sobre los resultados en términos de la detección de tendencias que esté arrojando el modelo.

Como se vio en la revisión del marco conceptual, las técnicas de clustering permiten obtener un agrupamiento de objetos con similares características [40], siguiendo la idea de la existencia de homogeneidad dentro de los grupos y heterogeneidad entre ellos. En ese sentido se propone en primer lugar realizar un clustering de las sesiones reales, tomando el elemento característico central o *centroide* como el objeto unificador del clustering y representante del mismo en el proceso de comparación con las sesiones artificiales. De esta forma el proceso de comparación entre sesiones se reduce y, dependiendo de la calidad de la elección del centroide, la pérdida de precisión no debería ser significativa.

Definición de la medida de similitud entre sesiones

La definición de la medida de similitud es un paso esencial en todo algoritmo de clustering ya que estipula el método a través del cual los elementos serán comparados, lo cual repercute en la calidad de los grupos encontrados.

La pregunta que sigue a continuación es cómo comparar sesiones de usuarios. Intuitivamente se pueden encontrar dos parámetros sobre los cuales comparar : el conjunto de elementos que componen las sesiones y el orden que siguen dentro de cada sesión. Existen diversos enfoques para determinar una medida de similitud entre las sesiones. Por ejemplo, [40] se propone un medida que involucra el uso de la distancia de Levenshtein, es decir, el número de operaciones necesarias (inserción, eliminación o sustitución) para transformar una cadena en otra. Se propone generar una medida que tenga presente no sólo una comparación entre los elementos conformantes de las sesiones, sino también el orden en que estos elementos se disponen. Una sesión tiene, tal como ha sido definida, la noción de orden secuencial entre sus componentes, luego el enfoque que se elige debe contenerlo.

Una aproximación a una métrica de comparación sobre sesiones se obtiene desde el artículo *A Framework for the Evaluation of Session Reconstruction Heuristics in Web-Usage Analysis* de Myra Spiliopoulou et al.[37] En tal documento se define el *grado de similitud* (degree of similarity) entre una sesión real y una construida artificialmente como el cuociente entre el mayor segmento de intersección continuo y el número de elementos presentes en ambas sesiones. Formalmente, sea r una sesión que pertenece al conjunto de sesiones reales R y c una sesión que pertenece al conjunto C de sesiones artificiales, entonces el *grado de similitud* se define como mediante la expresión 3.6.

$$deg_s(r, c) = \frac{|r \sqcap c|}{|r \cup c|} \tag{3.6}$$

La pregunta que subyace es cómo calcular el numerador de la expresión anterior, es decir, el segmento de intersección continuo máximo entre las sesiones. Para responder tal interrogante se debe recurrir a un problema clásico de las Ciencias de la Computación, *El problema de la subcadena común más larga*,(LCS por sus siglas en inglés). Existen diversos métodos para resolución del problema[5], siendo la técnica más utilizada la que involucra el uso de *generalized suffix tree*, cuyo tiempo de resolución es lineal[14], como también el uso de programación dinámica. Dado lo anterior, se propone una medida se similitud para la comparación de las sesiones representada por la expresión 3.7.

$$sim_d(r, c) = \frac{|LCS(r, c)|}{max\{|r|, |c|\}} \tag{3.7}$$

Elección del método de Clustering

De los métodos de clustering presentados en el Marco Conceptual, se propone implementar uno inspirado en el clustering jerárquico, básicamente debido a la naturaleza de los objetos que se desea agrupar. Las sesiones de usuario no son elementos indivisibles y no tienen un valor característico por si mismos, sino que son sus componentes y el orden en que éstos se encuentran, los que definen a cada sesión. Tanto los métodos de clustering de partición como los basados en densidad, hacen un uso intensivo de los valores de los elementos para llevar a cabo el agrupamiento, como la utilización de espacios métricos o la definición de valores medios. Tal propiedad no juega a favor de las sesiones de usuario.

Por otro lado, el clustering jerárquico sólo involucra el uso de las *distancias* o *niveles de similitud* entre los objetos, lo cual encaja de manera natural con el tratamiento de datos que no son netamente numéricos. Una de las desventajas es el hecho de que este tipo de clustering se vuelve ineficiente cuando el volumen de datos es muy grande.

Elección del *centroide*

Si bien la noción de *centroide* está asociada a los métodos de clustering particional, se hace necesario tomar prestado ese concepto con el fin de poder caracterizar a los clusters, dentro de la agrupación jerárquica, a través de una sesión particular. La *sesión centroide* será representativa del cluster respectivo, y todas las operaciones de comparación se realizarán sobre ella. En ese sentido, se propone utilizar el valor del grado de similitud definido anteriormente a través de la ecuación 3.7, pero esta vez acumulado por cada sesión perteneciente al cluster y posteriormente seleccionar la sesión que posee el grado de similitud acumulado mayor. El Algoritmo 3.2.1 muestra el procedimiento.

Algorithm 3.2.1: Elección centroide

Data: cluster $\mathcal{C}$

```
Initialize Contenedor = new Contenedor();
foreach sesión i ∈ C do
    foreach sesión j ∈ C do
        se suman las similitudes con todas las sesiones usando ecuación 3.7
        AcumSim(i) = AcumSim(i) + Sim_d(i, j);
    se inserta cada sesion y su valor acumulado de similitud
    Contenedor.add(AcumSim(i), i);
se retorna la sesion que tenga el mayor valor de similitud acumulada
return GetId(GetMax(Contenedor())) ;
```

3.2.2. Caracterización del Grafo

Dado que la recolección de la sesiones se produce en intervalo de tiempo considerable, aproximadamente 1 mes, el grafo por el cual deberán caminar las hormigas artificiales puede presentar variaciones. Esto representa un complicación en el diseño del algoritmo basado en ACO debido a que el medio en donde por el cual se mueven las hormigas varía a través del tiempo. En ese sentido se propone aproximar la dinámica del sitio a una razón diaria, es decir, un grafo por cada día asociado a la recolección de las sesiones. Dentro de cada día se considera un grafo estático.

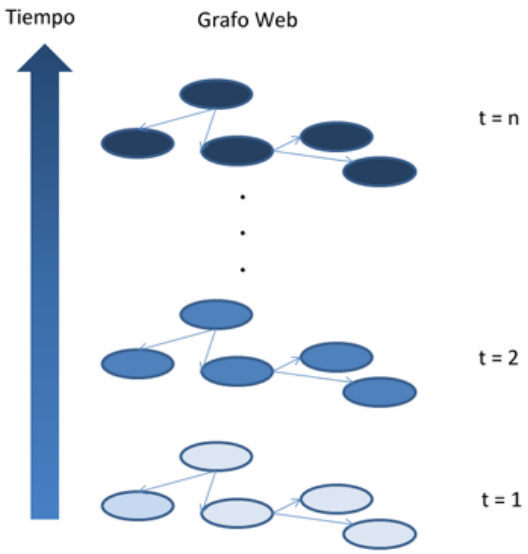

Figura 3.2: Disposición esquemática de los grafos web. Elaboración propia

Nodos del grafo

Siguiendo la configuración expuesta en la prueba de concepto, cada nodo dentro de un grafo representa una página web. Cada nodo tiene asociado un vector L que representa las palabras más importantes asociadas a la página. El método de construcción en este modelo es diferente del presentado en la prueba de concepto.

En primer lugar se debe obtener el conjunto de las palabras clave más importante dentro del grafo. Para cada grafo:

- Se obtienen todas las palabras existentes en el grafo y se almacenan en el conjunto *Keywords-Grafo*.

- Para cada palabra que pertenece a *KeywordsGrafo* se calcula su valor de TF-IDF en con respecto a cada página página. Si una página no posee la palabra clave, entonces el valor TF-IDF será 0. Luego para cada palabra clave se escoge el mayor valor de TF-IDF que haya obtenido. Este pasará a ser su valor representativo dentro del grafo. Así el valor $tfidf_i$ de la palabra i será $tfidf_i = max\{tfidf_{i1}, tfidf_{i2}, ..., tfidf_{iN}\}$ donde N representa el número de páginas (nodos) en el grafo.
- Se determinan los límites $tfidf_{min}$ y $tfidf_{max}$ como una función de los valores reales obtenidos para todas las palabras clave en el paso anterior. La idea es restringir las palabras a un conjunto que no posea valores extremos que distorsionen los posteriores análisis.
- Se obtienen todas las palabras cuyo valor de TF-IDF se encuentra entre $tfidf_{min}$ y $tfidf_{max}$ y se almacenan en el conjunto *ConjuntoMostImportantKeywords*. Formalmente: $ConjuntoMostImportantKeywords = \{k_i \in KeywordsGrafo \setminus tfidf_{min} \leq tfidf_i \leq tfidf_{max}\}$
- Finalmente se lleva a una representación vectorial, donde los elementos de *ConjuntoMostImportantKeywords* son dispuestos de manera decreciente en función su valor de TF-IDF en un vector llamado *VectorMostImportantKeywords*.

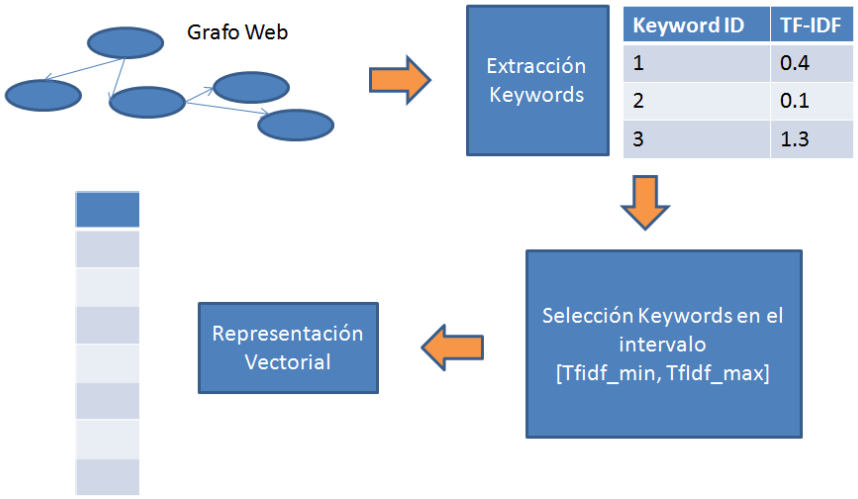

Figura 3.3: Obtención del vector con las palabras clave más importantes. Elaboración propia

En base a este vector se generarán los vectores L de cada página del grafo:

Para un grafo determinado, sea *VectorMostImportantKeywords* el vector con las palabras clave más relevantes en función de su valor de TF-IDF. Este vector contiene los *id* de las palabras clave y está ordenado de manera decreciente en función del valor de TF-IDF calculado anteriormente.

Para cada página p perteneciente al grafo se genera un vector vacío L_p.

Cada página p tiene asociado un conjunto $ConjuntoKeywordsPage_p$ que contiene todas las palabras clave existentes en tal página.

La longitud del vector L_p será igual a la del vector *VectorMostImportantKeywords*.

Existirá una correspondencia en términos de los índices de los vectores *VectorMostImportantKeywords* y L_p, es decir, el valor en el índice j del vector *VectorMostImportantKeywords* estará relacionado con el valor del vector L_p en su respectivo índice j, tal como se ilustra en la siguiente figura:

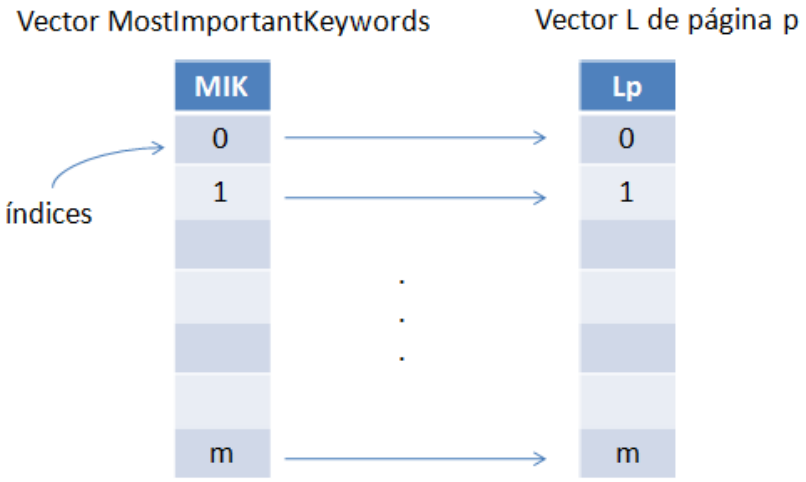

Figura 3.4: Relación entre los vectores de palabras importantes y el vector L de cada página. Elaboración propia.

Los valores l_pi que tendrá el vector L_p vendrán dados por la siguiente expresión:

$$l_pi = \begin{cases} TF - IDF_{ip} & \text{si } VectorMostImportantKeywords_i \in ConjuntoKeywords_p \\ 0 & \text{si no} \end{cases}$$

Es decir, para cada componente i del vector *VectorMostImportantKeywords*, $VectorMostImportantKeywords_i$, se busca si existe en el conjunto de palabras clave del la página p, $ConjuntoKeywordsPage_p$. Si la búsqueda fue satisfactoria, el valor que se asignará en el índice i del vector L_p, L_{pi} será igual al valor de TF-IDF de la palabra $VectorMostImportantKeywords_i$ en la página p. Si no, el valor será 0.

De esta forma cada vector L_p contendrá valores no nulos en la medida que las palabras que

existen en p estén contenidas en el conjunto de palabras más importantes. Asimismo, esto ayuda a uniformar el contenido de cada página en vectores de igual longitud, lo cual será un requisito para las posteriores comparaciones.

Arcos del grafo

Los arcos del grafo representan los enlaces existentes entre las páginas. Tal como el la prueba de concepto, se propone que sean los portadores de los trazos con feromona, cuyo nivel será proporcional a la conveniencia de pasar desde una página a otra.

El valor inicial de los trazos de feromona será análogo al utilizado en la prueba de concepto, es decir, se usarán los valores asociados a la matriz de transición calculada en función del número de enlaces que salen de cada página y de una probabilidad de simula la preferencia por seguir navegando a través de los enlaces en lugar de escribir directamente una nueva dirección.

Así, cada link entre la página i y la página j será inicializado con un valor de feromona igual a:

$$p_{ij} = \frac{p * g_{ij}}{c_j} + \frac{1-p}{n}, \tag{3.8}$$

donde cada componente sigue la misma definición expuesta en la ecuación 3.1.

3.2.3. Caracterización de las Hormigas Artificiales

Es en este apartado donde se concentran las mayores diferencias del presente modelo con el expuesto en la prueba de concepto.

Vector de preferencias por texto μ

Uno de los elementos más importantes en el modelo es el medio de aprendizaje para las hormigas. Se considera en este caso el aprendizaje como la modificación progresiva de un componente interno del agente con el cual es capaz de interactuar con el medio. Se espera que las modificaciones mencionadas influyan en el comportamiento del agente en función de un objetivo. En este caso se propone utilizar el vector de preferencias por texto μ definido en la Prueba de Concepto, pero asignándole los siguientes cambios:

- Los vectores tendrán un largo fijo igual al del vector *VectorMostImporantKeywords* (e igual a los vectores L).

- Los valores del cada vector μ serán una representación artificial de la preferencia por parte de un agente por una palabra clave determinada, perteneciente al conjunto de la palabras claves más importantes. Así, para un vector μ determinado, el valor numérico en el índice i será la preferencia por la palabra que se encuentra en el índice i en el vector *VectorMostImportantKeywords*.
- En la prueba de concepto, los valores seguían una distribución uniforme entre 0 y 100 y luego el vector resultante era normalizado. Para el nuevo modelo se propone implementar tramos de valores bajo el supuesto que de un conjunto de palabras fijo, un usuario presenta un preferencia *alta* sólo por una proporción pequeña[42]. En ese sentido si el valor 0 representa nulo interés en una palabra determinada y 1 representa total interés, se propone la distribución:
  - Alta preferencia: 20 % de las palabras. Sus valores oscilarán entre 0.8 y 1.
  - Mediana preferencia: 60 % de las palabras. Sus valores oscilarán entre 0.2 y 0.8.
  - Baja preferencia: 20 % de las palabras. Sus valores oscilarán entre 0 y 0.2.

Función de utilidad

El objetivo de cada hormiga es recorrer el grafo con el fin de satisfacer completamente su *necesidad de información*. En ese sentido se define la función de utilidad $utilidad_h$ que tendrá un valor numérico asociado a cuanta información la hormiga h ha consumido. El nivel de utilidad alcanzado deberá ser proporcional a la similitud entre las páginas que la hormiga vaya recorriendo y su preferencia especifica por información. Asimismo se define un valor de *saciedad*, es decir, un límite $limUtilidad_h$ para la utilidad obtenida. Si la hormiga alcanza tal límite, su recorrido dentro del grafo termina.

Inicialización de las hormigas

A cada hormiga k se le asigna un vector μ_k, el cual definirá su preferencia inicial por texto dentro del grafo. Adicionalmente se le asocia uno de los clusters de sesiones encontrados anteriormente. De esta forma, cada hormiga deberá *aprender* de un conjunto de sesiones determinadas, modificando su propio vector μ_k. Luego, el número de vectores μ necesarios debe ser igual al número de clusters de sesiones reales que se hayan encontrados.

Elección del nodo de partida

Dado que cada hormiga tiene un cluster de sesiones asociado, se extraen los identificadores de todas las páginas que aparecen como *primera página* dentro de las sesiones pertenecientes al cluster mencionado. Adicionalmente se calcula, para cada página, la frecuencia que posee como *primera página*, la cual finalmente se expresa como una proporción. Por ejemplo, sea el

cluster $C = \{(2,1,4),(2,1,3),(1,2,2),(2,3,2,4),(3,1,2)\}$, donde cada conjunto representa una sesión en términos de sus identificadores únicos. Se puede apreciar que el conjunto de páginas iniciales está dado por $PaginasIniciales = \{1,2,3\}$ donde la proporción estaría dada por $ProporcionPaginasIniciales = \{1/5, 3/5, 1/5\}$, es decir, una distribución de probabilidad discreta.

La probabilidad acumulada se representará como segmentos que separan a las páginas iniciales, utilizando el método de Monte Carlo, tal como se muestra en la Figura 3.5 siguiendo el ejemplo anterior (los valores fraccionarios han sido llevados a su expresión decimal).

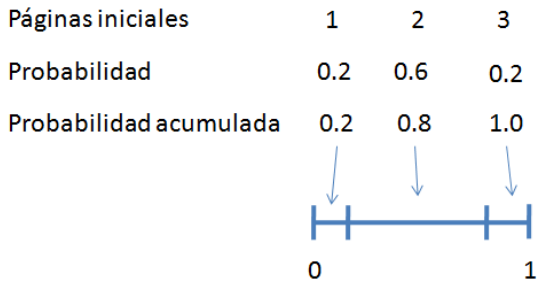

Figura 3.5: Disposición de las probabilidades acumuladas de las páginas iniciales. Elaboración propia.

Posteriormente se genera un número aleatorio entre 0 y 1. Se seleccionará la página asociada al intervalo que contiene al número aleatorio, tal como se ve en la Figura 3.6.

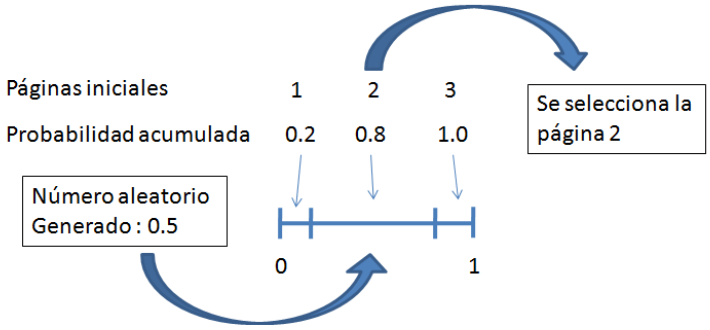

Figura 3.6: Elección de la página de inicio. Elaboración propia.

3.2.4. Algoritmo de aprendizaje

La idea detrás del aprendizaje es la progresiva modificación del vector μ asociado a cada hormiga con el objetivo de que las sesiones que la hormiga vaya generando en el grafo sean lo más

parecidas a las presentes en el cluster de sesiones que le fue asociado.

El algoritmo se traduce en los pasos siguientes:

1. Antes de iniciar la sesión se genera una copia del vector μ y se altera arbitrariamente en ésta uno de sus componentes. Este nuevo vector será el utilizado para generar la sesión, la que estará asociada a esta alteración.
2. A cada hormiga se le asocia un cluster de sesiones reales.
3. Al comenzar su viaje a través del grafo se elige una página inicial en función de las páginas iniciales de las sesiones reales del cluster.
4. Al llegar a la página inicial, se calcula la utilidad como la similitud entre el vector de palabras importantes de la página en cuestión y el vector μ propio de la hormiga.
5. Se define una serie de condiciones de término de sesión.
6. Se toma el conjunto de páginas vecinas a la actual y decide probabilísticamente el destino en función de los niveles de feromona en los enlaces asociados a cada página vecina.
7. Al llegar a la página destino se calcula la utilidad y proporcionalmente a ésta se actualiza el nivel de feromona asociado al enlace entre la página anterior y la actual.
8. Cuando se cumple una de las condiciones de salida, la sesión termina y es almacenada en un contenedor ad-hoc.

A continuación de detalla el proceso definiendo los pasos más relevantes.

Modificación a priori del vector μ

Para cada hormiga, antes de comenzar un viaje a través del grafo, se aumenta el valor de uno de los componentes de su vector μ, es decir, se incrementa la preferencia de la hormiga por una determinada palabra. Este proceso se lleva acabo duplicando el vector μ original y realizando la modificación en esa copia, la cual es estregada a la hormiga para que comience su viaje. El índice del elemento modificado es guardado, ya que se relacionará posteriormente con la sesión generada, tal como lo muestra la figura 3.7.

La modificación se lleva a cabo a través de la adición de una constante a un componente del vector μ. El supuesto detrás de esta acción tiene que ver con que al modificar (en este caso aumentar) la preferencia que tiene una determinada hormiga por una de las palabras que caracterizan el contenido del grafo, las sesiones que genere esta hormiga estarán influenciadas por tal modificación, es decir, conforme pase el tiempo, convergerán a sesiones cuyas páginas contengan la palabra, cuya preferencia a sido aumentada, en una mayor proporción.

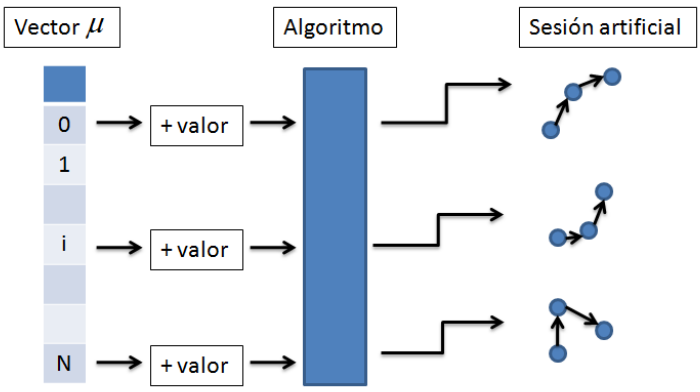

Figura 3.7: Relación entre la modificación de la preferencia y la ruta generada. Elaboración propia.

Llegada de la hormiga al grafo

Posteriormente la hormiga es dejada en la página inicial, calculada según el método visto anteriormente. En este paso, la hormiga debe comparar su vector de preferencias con el vector de palabras L de la página. En esta acción se pretende seguir la analogía de cuando un usuario llega por primera vez a un sitio, evalúa su página inicial y en función de la utilidad que le genera el contenido, decide seguir o no a través de un enlace.

La forma de comparación será a través de la similitud del coseno. Este valor entregará un nivel de sincronización entre la preferencia del usuario y el contenido de la página. A este valor se le relacionará con la utilidad obtenida por llegar a esa página.

$$utilidad = \frac{\langle \mu \bullet L_p \rangle}{||\mu|| ||L_p||} \tag{3.9}$$

Inicialmente el valor de la utilidad es 0.

Posteriormente a este cálculo se debe inspeccionar si el valor obtenido es menor que el límite de utilidad. Si lo es, la hormiga se detiene y sale del grafo, si no , continúa su viaje a través del grafo mediante el método que se presenta a continuación.

Método de selección del nodo a seguir

Dada una hormiga en su página inicial de sesión, debe continuar su camino a través de los enlaces que esta página posee. Cada uno de los enlaces contiene un nivel de feromona inicial, tal como se señaló en la Sección 3.2.2. Como los valores de feromona vienen de la aplicación de una matriz de

transición, en el fondo son probabilidades discretas. Luego el método para elegir qué enlace seguir (y consecuentemente a qué pagina moverse) será a través de la simulación utilizando la probabilidad acumulada, es decir, el mismo método utilizado para elegir la página inicial.

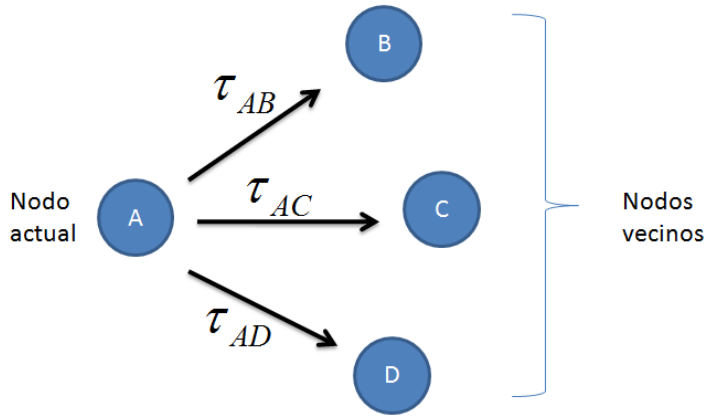

Figura 3.8: Disposición de los niveles de feromona τ_{ij}. Elaboración propia.

En la Figura , los valores de τ_{ij} representan los niveles de feromona existentes en los enlaces que nacen en la página A. Para ejemplificar el método, sean $\tau_{AB} = 0{,}7; \tau_{AC} = 0{,}2 y \tau_{AD} = 0{,}1$, luego si se genera un número aleatorio entre 0 y 1 y se compara con los intervalos de feromona acumulada, el ejemplo quedaría de la forma representada en la Figura 3.9.

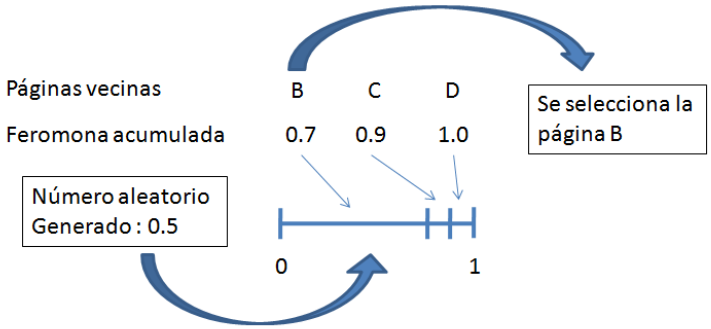

Figura 3.9: Selección nodo vecino. Elaboración propia.

Llegada al nodo destino

En este paso se vuelve a calcular la utilidad de haber llegado al nodo destino a través de la medida de similitud del coseno. Posteriormente la utilidad obtenida en el paso actual debe ser

sumada a la utilidad que llevaba hasta el momento, tal como lo ilustra la Ecuación 3.10.

$$utilidad(t) \quad = \quad utilidad(t) + \sum_{i=0}^{t-1} utilidad(i) \tag{3.10}$$

Actualización de los niveles de feromona

La actualización de los niveles de feromona está divida en dos partes. La primera tiene relación con la modificación en los enlaces que nacen de la página actual, en especial del que se eligió como camino a seguir. En segundo lugar se lleva a cabo un disminución leve o *evaporación* de los niveles de feromona en todos los enlaces pertenecientes al grafo.

En términos de la modificación del nivel de feromona en el enlace elegido, en la prueba de concepto se utilizó la expresión proveniente del Modelo Logit de preferencias discretas, el cual ha sido ampliamente utilizado y aplicado para modelar la forma en la cual una persona busca información[26, 33]. Uno de los problemas encontrados en la prueba de concepto fue que los valores obtenidos mediante la ecuación 3.4 eran demasiado bajos, lo cual repercutía en los tiempos de convergencia del algoritmo global, ya que el refuerzo de una elección era demasiado bajo en relación con los cambios posteriores que se realizaban en el grafo, como la evaporación.

En ese sentido se propone la siguiente expresión , esta vez incluyendo el valor $\varepsilon > 1$ con el cual se pretende potenciar el reforzamiento en el nivel de feromona. De esta forma, sea i la página actual y j la página elegida como nodo destino, luego la expresión que será sumada al nivel actual de feromona τ_{ij} vendrá dada por la Ecuación 3.11.

$$\tau_{ij} \quad = \quad \tau_{ij} + \frac{e_{ij}^{V}}{\sum_{k \in Vecinos_i} e_{ik}^{V}} * \varepsilon, \tag{3.11}$$

donde V es una abreviación de la utilidad definida entre un vector de preferencias μ y el vector L de cada página a través la medida se similitud del coseno.

Posteriormente se normalizan los valores asociados a la feromona en todos los enlaces del nodo de inicio con el fin de mantener su suma acumulada en 1.

Adicionalmente, con el fin de acercar las sesiones artificiales generadas a las existentes dentro del cluster asociado a cada hormiga, se propone añadir un término más a la expresión anterior si se cumple que el enlace elegido pertenece al conjunto de enlaces o transiciones presentes en el cluster.

Así, para los casos donde se cumple la condición anterior, la actualización de los niveles de

feromona estará dada por el Ecuación 3.12.

$$\tau_{ij} = \tau_{ij} + \frac{e_{ij}^{V}}{\sum_{k \in Vecinos_i} e_{ik}^{V}} * \varepsilon + boost, \tag{3.12}$$

donde *boost* es el parámetro positivo fijo que se suma y que potencia aún más el nivel de feromona en el enlace elegido.

Condiciones de término de sesión

Las condiciones de término de la sesión en curso son las siguientes:

- Se sobrepasa el límite de utilidad: Tal como se vio anteriormente, cada hormiga tiene asociado un límite de utilidad. Si esté se alcanza o sobrepasa, la sesión se detiene.
- La hormiga cayó en un loop: Si la hormiga comienza a moverse en forma circular, implica un comportamiento alejado de las prácticas de los usuarios web reales. En tal caso, la sesión se termina. Para identificar el posible loop, cada vez que se añade un nuevo nodo a la sesión actual se inspeccionan sus apariciones pasadas y si las secuencias se repiten dos veces, se toma como un loop inválido.
- Visita persistente de una página : Si se visita , dentro de la misma sesión, una página más de 4 veces consecutivas, la sesión se detiene. Si bien este comportamiento se puede dar en la realidad, por ejemplo, al visitar galerías fotográficas, se asume el supuesto de que no representa un comportamiento normal.

Almacenamiento de la sesión generada

Una vez se ha cumplido alguna de las condiciones de salida, la sesión termina y es almacenada en un contenendor *contenedorSesiones*, en el cual se ingresa tanto la sesión generada, como también el índice en el que el vector μ fue modificado.

De esta forma, al final de esta primera parte se tendrá un número de sesiones igual al número de componentes del vector μ.

Recapitulación proceso de generación de sesiones

Con el fin de facilitar la comprensión del proceso global de aprendizaje, en este apartado se muestra el comportamiento del método de generación de sesiones a través del Algoritmo 3.2.2.

Algorithm 3.2.2: Algoritmo AntSurfer

```
Data: Cluster c , Vector μ
Initialize contenedorSesiones(), utilidad = 0;
μ_Copia ← μ;
for i ← 0 to μ.length() do
    sesion() ;
    μ_Copia[i] ← μ_Copia[i] + Modificaμ;
    Normalizar(μ_Copia);
    nodoActual ← getNodoInicial(c);
    sesion.add(nodoActual);
    L ← getVectorL(nodoActual);
    utilidad ← utilidad + CosSim(μ_Copia, L);
    while utilidad < max AND !InvalidLoops AND !InvalidRep do
        Vecinos ← getVecinos(nodoActual) ;
        nodoDestino ← getNodoDestino(Vecinos) ;
        sesion.add(nodoDestino);
        L ← getVectorL(nodoDestino);
        utilidad ← utilidad + CosSim(μ_Copia, L);
        if link(nodoActual → nodoDestino) ∈ ConjuntoLinks(c) then
            τ_nodoActual→nodoDestino ←
            τ_nodoActual→nodoDestino + LogitIncrement(nodoActual, nodoDestino) + Boost;
        else
            τ_nodoActual→nodoDestino ←
            τ_nodoActual→nodoDestino + LogitIncrement(nodoActual, nodoDestino);
        evaporacionGrafo();
        nodoActual ← nodoDestino;
    contenedorSesiones.add(sesion, i);
    sesion.clear();
```

3.2.5. Modificación de los vectores de preferencias μ

Los constantes viajes de cada hormiga terminan con el llenado de un contenedor *contenedorSesiones* que posee tanto las sesiones generadas como también el índice del vector μ que fue modificado para generarlas. Cabe recalcar que durante la generación de sesiones se modificó una copia temporal de los vector μ, no el vector original, lo cual se explica a continuación. El

proceso de modificación real de vector μ se basa en la comparación entre las sesiones generadas y el centroide del cluster asociado a la hormiga. Cada sesión generada se compara con el centroide a través de la función de similitud definida por la Ecuación 3.7. Posteriormente, de todos los valores calculados, se rescatan los índices asociados a las máximas similitudes encontradas, tal como se puede ver en la Figura 3.10.

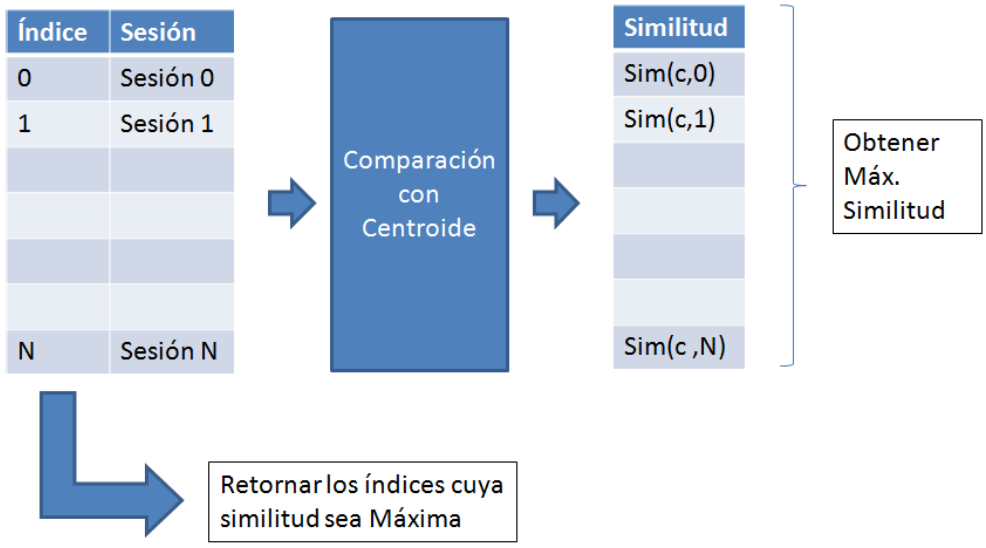

Figura 3.10: Selección de los índices de μ a modificar. Elaboración propia.

El razonamiento tras este método es encontrar la modificación en el vector μ que genere las sesión más similar al valor representativo del cluster. Una vez obtenidos los índices se procede a modificar el vector μ original. Posteriormente, cada vector es normalizado con el fin de mantener las proporciones de las preferencias por cada componente. Este proceso se repite un número determinado de veces, lo cual, para cada vector, progresivamente incrementa la preferencia por ciertos componentes y la disminuye para otros. Así, finalmente, el resultado del proceso de aprendizaje es un conjunto de vectores de preferencia μ entrenados, cada uno asociado a un cluster de sesiones reales.

3.2.6. Validación del modelo

Para facilitar la notación, el intervalo de tiempo del cual se extrajo los datos para el entrenamiento se denominará $T_{entrenamiento}$, el cual a priori se toma como un periodo que comprende 1 mes. De la misma manera, el periodo del cual se extraerán los datos para la validación se designará por $T_{validacion}$, el cual también se toma como un mes.

La idea detrás del proceso de validación del modelo es realizar una comparación entre sesiones generadas por hormigas artificiales entrenadas y las sesiones reales en un nuevo intervalo de tiempo. En sentido el input para este proceso son los vectores de preferencia μ entrenados. En términos simples, los pasos que comprende este proceso son los siguientes :

1. Se asocia un grupo de hormigas a cada vector de preferencias μ entrenado.
2. Se extrae el contenido y estructura del sitio web real con el fin generar el medio para las hormigas.
3. Se procesa comportamiento de usuario correspondiente a $T_{validacion}$, se obtienen las sesiones y se clusterizan. Adicionalmente se obtienen los centroides.
4. Cada grupo de hormigas es liberado en un grafo web y sus sesiones son almacenadas y procesadas. Se calcula un sesión representativa para cada grupo.
5. Se realiza una comparación entre los clusteres de sesiones reales y las sesiones representativas generadas a través de las hormigas entrenadas, con lo cual se concluye sobre el nivel de similitud que tiene el modelo en términos de la simulación del comportamiento real del usuario en la Web.

A continuación se presentan las secciones que detallan en mayor profundidad el proceso de validación.

Extracción de datos para la validación

Se deben llevar a cabo los mismos procesos de extracción y procesamiento tanto de las sesiones reales de usuario como de la estructura y contenido del sitio web, realizadas en la fase de entrenamiento, pero esta vez en $T_{validacion}$. De esta forma se debe obtener lo siguiente:

- Conjunto de sesiones reales durante el nuevo intervalo de tiempo. Se pueden extraer a través de algunos de los métodos anteriormente mencionados (estraterias reactivas o proactivas).
- Representación como grafo del sitio web y sus diferentes versiones durante el nuevo intervalo de tiempo. Esto implica almacenar tanto la estructura como el contenido.

Preparación grafo web

Se deben calcular los niveles de feromona iniciales en el nuevo grafo de acuerdo al método que utiliza la matriz de transición visto anteriormente.

También se debe representar el contenido de cada página como un vector L que incluya las palabras más importantes en función de su valor TF-IDF. En este caso, el conjunto de palabras sobre las que se construirá el vector debe ser igual al encontrado en proceso de entrenamiento. Con esto se asegura que las comparaciones serán coherentes en términos de los componentes sobre los que se busca similitud. Además, se garantiza que las longitudes de los vectores L serán iguales a las de los vectores μ entrenados, requisito para calcular el producto punto dentro de la medida de similitud.

Clustering de sesiones reales

El conjunto de sesiones reales obtenido en el intervalo de validación $T_{validacion}$ debe ser clusterizado utilizando el mismo método presentado en el proceso de entrenamiento con el fin de generar una equivalencia entre ambos conjuntos. Cabe señalar que dada la naturaleza de los datos no se puede asegurar a priori que el número de clusters que se encuentren en el intervalo de validación sea igual al encontrado en el de entrenamiento dado que ni el número de sesiones ni la estructura de éstas será igual para ambos meses.

Inicialización de las hormigas

Para cada uno de los vectores μ entrenados, se genera un conjunto de hormigas artificiales a las cuales se les asocia dicho vector, es decir, serán hormigas con una *preferencia por texto* definida. El número de hormigas debe ser ser considerable debido a que en esta etapa el concepto de comunicación indirecta o *estigmergia* será relevante.

Adicionalmente cada grupo de hormigas tendrá un grafo web para recorrer, tal como se ilustra en la figura[7] 3.11.

En ese sentido, cada grupo de hormigas realiza un proceso independiente del otro.

Comportamiento de las hormigas

Cada grupo de hormigas se libera en su grafo asociado y todas comienzan a recorrerlo al mismo tiempo. Como dentro de un grupo todas contienen el mismo vector de preferencias μ, todas buscarán la misma información a través del sitio. Haciendo una analogía con la biología, todas las hormigas de un grupo buscarán el mismo tipo de *alimento* (en este caso, buscarán la misma información). De esta forma, en esta etapa se está en presencia de la implementación convencional de la metaheurística, donde el conjunto de agentes artificiales recorre el grafo y en función de la conveniencia de los caminos generados, se actualizan los niveles de feromona existentes, lo cual, a

[7]El dibujo de la hormiga fue sacado de iStockphoto.com - Tomasz Zachariasz

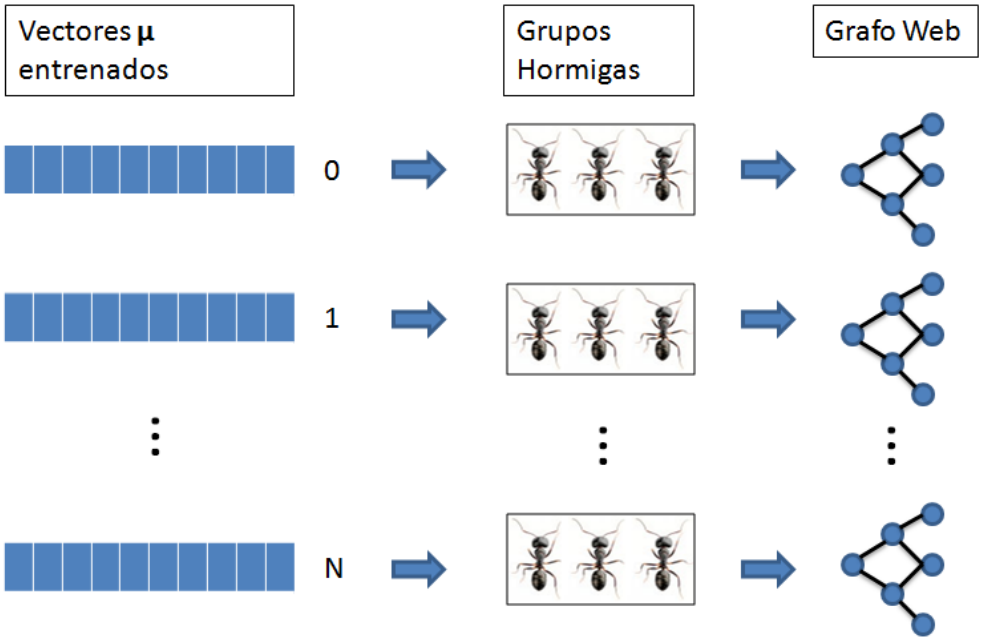

Figura 3.11: Inicialización proceso de validación. Elaboración propia.

través de un proceso de comunicación indirecta y refuerzo auto catalítico, va haciendo converger el algoritmo a una solución, que en este caso se representa como una sesión o un conjunto pequeño de sesiones.

En ese sentido el algoritmo que gobierna el comportamiento de las hormigas es análogo al utilizado para en la fase de entrenamiento exceptuando la componente de aprendizaje. Luego, para una hormiga perteneciente al grupo H_j (es decir, asociado al vector de preferencias entrenado μ_j),

su comportamiento estaría representado por el Algoritmo 3.2.3.

Algorithm 3.2.3: Algoritmo en fase de validación

```
1  Initialize contenedorSesiones(), utilidad = 0, MAXIteraciones;
2  for i ← 0 to MAXIteraciones do
3      sesion() ;
4      nodoActual ← getNodoInicial();
5      sesion.add(nodoActual);
6      L ← getVectorL(nodoActual);
7      utilidad ← utilidad + CosSim(μ, L);
8      while utilidad < max AND !InvalidLoops AND !InvalidRep do
9          Vecinos ← getVecinos(nodoActual) ;
10         nodoDestino ← getNodoDestino(Vecinos) ;
11         sesion.add(nodoDestino);
12         L ← getVectorL(nodoDestino);
13         utilidad ← utilidad + CosSim(μ, L);
14         τ_{nodoActual→nodoDestino} ←
           τ_{nodoActual→nodoDestino} + LogitIncrement(nodoActual, nodoDestino);
15         evaporacionGrafo();
16         nodoActual ← nodoDestino;
17     contenedorSesiones.add(sesion, i);
18     sesion.clear();
```

De esta forma, cada hormiga perteneciente a un grupo determinado va progresivamente generando sesiones, las cuales son almacenadas. Finalmente se tiene un contenedor por cada grupo con las sesiones artificiales.

Caracterización de las sesiones generadas

Para cada grupo de hormigas se generó un contenedor en el cual fueron insertadas secuencialmente las sesiones que éstos generaban. Cabe recordar que existen tantos grupos de hormigas como vectores de preferencias μ entrenados.

Con respecto a la morfología de las sesiones almacenadas, se espera que inicialmente haya heterogeneidad, debido a las decisiones aleatorias en un grafo donde los niveles de feromona no permiten reconocer rutas definidas. A medida que transcurre el tiempo, la actualización y reforzamiento de ciertos trazos por parte del conjunto de hormigas, como también la evaporación paulatina de los niveles de feromona, delinearán ciertos caminos dentro del grafo que llevarán progresivamente a la generación de sesiones más uniformes entre sí.

Luego, se espera que exista un límite o *umbral* que separe las sesiones que se podrían denom-

inar como *ruido* de las que poseen cierto nivel de convergencia y que son útiles como herramienta para caracterizar al grupo de hormigas. Para encontrar este conjunto de sesiones se propone extraer un porcentaje de las últimas sesiones ingresadas al contenedor, es decir, de las que con mayor probabilidad han alcanzado la convergencia. A este conjunto se sesiones se le llamará *Conjunto Factible*.

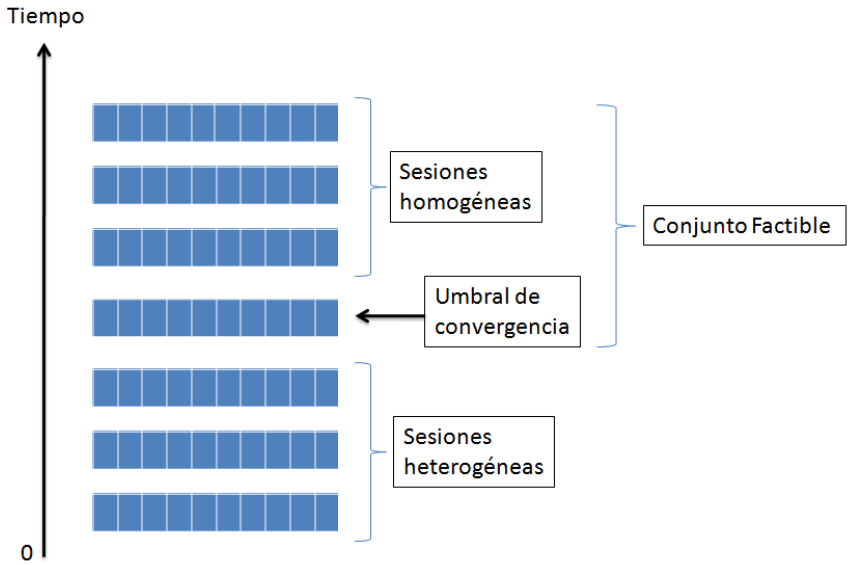

Figura 3.12: Comportamiento de las sesiones artificiales almacenadas. Elaboración propia.

Metodología de comparación entre sesiones

La idea principal de la validación se basa en la comparación agregada de las sesiones con el fin de encontrar niveles de similitud y con ellos poder tomar una decisión sobre el rendimiento del modelo.

En primer lugar se debe realizar clustering entre las sesiones reales provenientes del intervalo de validación $T_{validacion}$. A cada cluster se le debe identificar un centroide como sesión representativa utilizando el mismo método visto en la fase de entrenamiento.

Posteriormente, a las sesiones artificiales generadas por cada grupo de hormigas, se les debe encontrar un valor central. Una vez más, se debe utilizar el método visto en la fase de entrenamiento.

Por último, se debe realizar un cruce entre los conjuntos anteriores comparando sus sesiones centrales. Cada valor central de las sesiones generadas por los distintos grupos de hormigas se debe comparar con todos los centroides de los clusters de sesiones reales calculando la medida de similitud.

Se debe fijar un *umbral de similitud* tal que si es alcanzado, permite decir que el valor central

de las sesiones generadas por un grupo de hormigas determinado explica el comportamiento del cluster con cuyo centroide fue comparado.

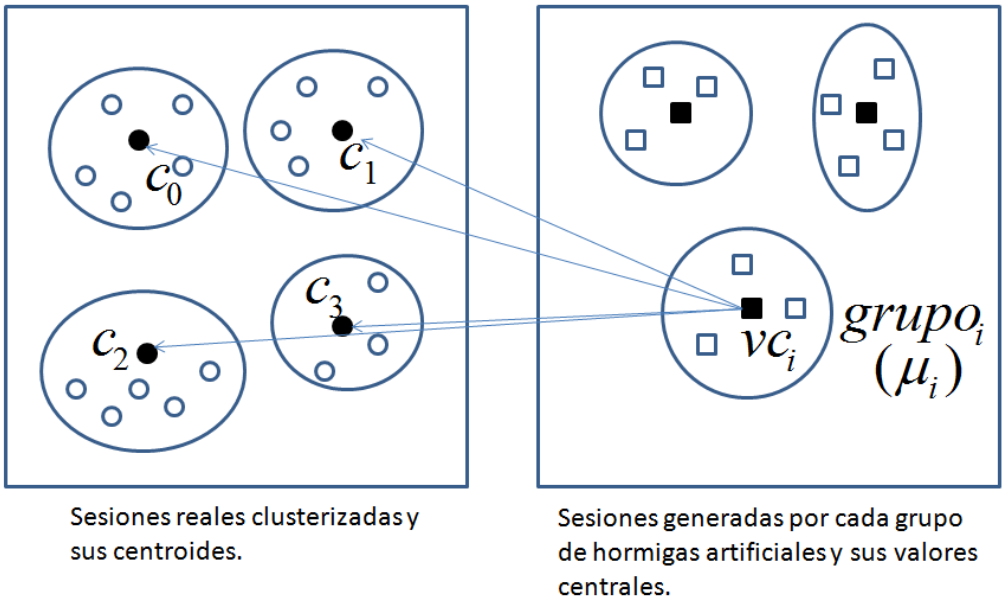

Figura 3.13: Inicialización proceso de validación. Elaboración propia.

En la Figura 3.13 se muestra el esquema de comparación, en el cual, dado un conjunto de sesiones generado por el grupo de hormigas i, se compara el valor central de ese conjunto vc_i con todos los centroides de los clusters de las sesiones reales. Cabe señalar que en la Figura 3.13 se ha cometido un abuso de notación al representar los cluster como un modelo de particional, siendo que el utilizado es jerárquico. La razón es simplemente que tal representación visual es más fácil de comprender.

Medidas cuantitativas de validación

La primera medida de validación que se propone tiene que ver con cuantificar el número de clusters de sesiones reales que pueden ser explicados por la simulación entrenada de hormigas, valor que puede ser obtenido a través de la Ecuación 3.13.

$$\%\ \text{Clusteres explicados} \quad = \quad \frac{\text{Nº Clusteres explicados}}{\text{Nº Total Clusteres}} \tag{3.13}$$

Una segunda medida que se propone se deriva de la anterior. Se tiene el supuesto de que los conjuntos de sesiones artificiales deberían ser capaces de explicar un rango amplio de clusters

de sesiones reales y no concentrarse en una minoría o que se de el fenómeno que una cantidad considerable de grupos expliquen el mismo cluster. Para visualizar esta situación se propone calcular la distribución del número de clusters explicados por grupo. Lo que se espera es que se obtenga una distribución similar a la Gausiana inversa, es decir, la existencia de pocos grupos que expliquen muchos clusters y muchos grupos expliquen pocos clusters.

En tercer lugar propone una comparación a nivel de contenido. Si se observa que un grupo de hormigas determinado g_i (cuyos integrantes tienen asociado el vector μ_i) es capaz de explicar el cluster de sesiones reales c_j, entonces debería existir una correlación entre las palabras más importantes presentes en las páginas de las sesiones que conforman el cluster, y los componentes que determinan el vector de preferencias μ_i. Como las palabras claves que componen los vectores μ y las que están presentes en las páginas provienen de un conjunto común, $ConjuntoMostImporantKeywords$, se puede llevar a cabo una comparación y ver si las palabras más importantes en el cluster están contenidas en las que el entrenamiento realizado al vector μ_i entrega mayor peso.

Capítulo 4

Resultados Experimentales

En este capítulo de lleva a cabo la implementación del modelo propuesto en un sitio web real y se analizan los resultados en términos de el proceso de aprendizaje como también en las sesiones que generen el conjunto de hormigas artificiales entrenadas en términos de su similitud con el comportamiento real del usuario web.

4.1. Descripción del sitio web a utilizar

La implementación del modelo fue realizada en el sitio web perteneciente al Departamento de Ingeniería Industrial de la Universidad de Chile[1]. Este sitio representa la mayor fuente de información acerca del departamento y sus actividades. Los contenidos que engloba son los siguientes:

- Noticias
- Eventos académicos
- Información sobre el Pregrado
- Información sobre los programas de Postgrado
- Medios de Comunicación
- Información general
- Cuerpo académico

[1] http://www.dii.uchile.cl

Figura 4.1: Página inicial del Departamento de Ingeniería Industrial.

Adicionalmente, este sitio sirve de plataforma de transición para un número no despreciable de *subsitios* correspondientes a los programas de postgrado y los centros de investigación pertenecientes al departamento.

Al llevar a cabo un análisis inicial desde la perspectiva del usuario, se puede ver que los sitios contenidos poseen una estructura estándar, en donde prima el contenido en forma de texto. En términos de la construcción y mantención de las páginas, se realizó una inspección más a fondo y se pudo concluir que existe heterogeneidad en relación a los sistemas y métodos utilizados en la elaboración de los sitios. Se utilizan desde CMS[2] como Wordpress[3], páginas estáticas construidas sobre HTML puro, páginas estáticas sobre HTML utilizando *frames*[4], como también sistemas que son actualizados a través de la plataforma que proporciona Microsoft Office.

Esta heterogeneidad es relevante en el sentido de las estructuras tanto de links como las que almacenan el contenido, no son uniformes lo cual puede representar un problema para las herramientas de extracción que se utilicen, las cuales deben ser lo suficientemente flexibles como para lograr su cometido.

4.2. Procesamiento del Contenido y Estructura

En este paso, los datos provenientes del sitio web en estudio son transformados y almacenados en una estructura que sirva de input para el modelo propuesto.

[2]Content Management System
[3]http://www.wordpress.org
[4]Sistema a través del cual una página es dividida en segmentos que pueden ser tratados de forma independiente.

4.2.1. Extracción de los datos

La extracción de los datos se realizó utilizando una versión modificada de la librería WebSphinx[5]. Si bien este crawler no está diseñado para gestionar grandes cantidades de datos [32], su desempeño es suficiente para llevar a cabo el experimento.

El proceso de extracción considera las variaciones posibles que se pueden dar tanto a nivel de estructura de links como de contenido. Luego, el crawler fue ejecutado diariamente y se llevaron a cabo modificaciones que permitieron darle la posibilidad de identificar los cambios que se produjeran en el sitio, como también la capacidad de consignarlos.

La implementación del crawler fue realizada en Java y su ejecución se realizó desde los servidores del Web Intelligence Group[6] perteneciente al Departamento de Ingeniería Industrial. En relación a los tiempos involucrados, se decidió que se utilizarían en total dos meses de extracción, el primer mes sería utilizado en el proceso de entrenamiento del modelo y el segundo mes en el proceso de validación. El modelo de datos asociado al proceso de extracción se muestra en la figura 4.2.

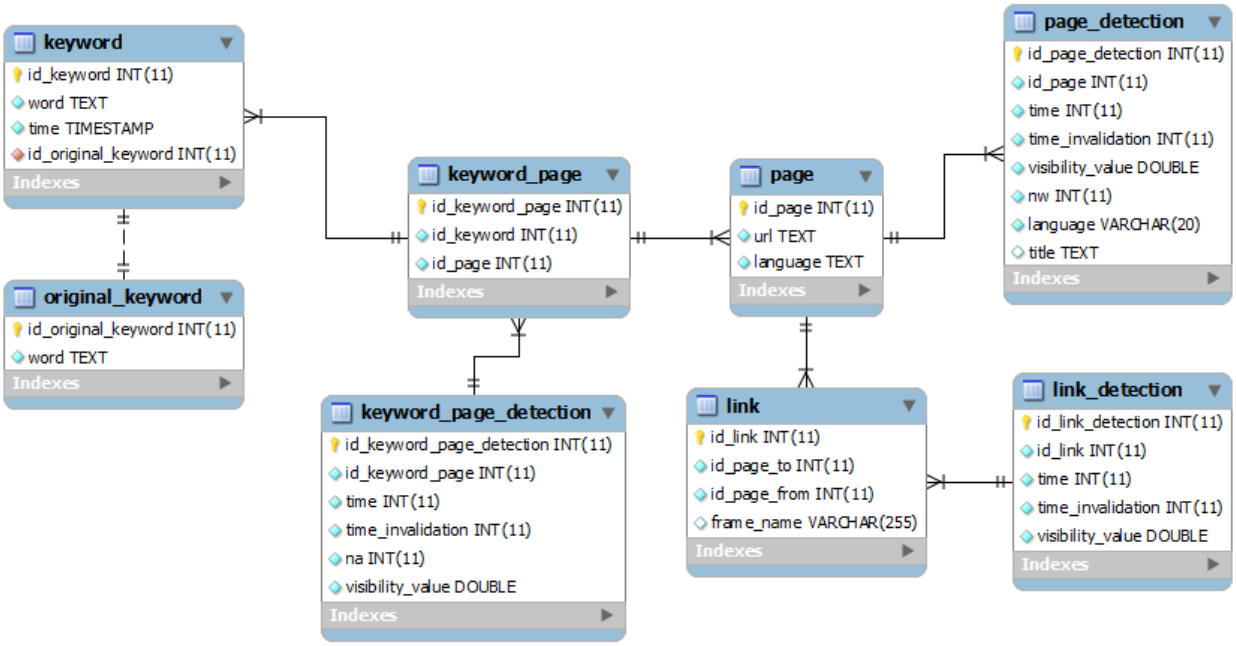

Figura 4.2: Modelo de datos utilizado por el web crawler. Elaboración propia.

- **Tabla : original_keyword**
  Esta tabla almacena las palabras clave en su forma original, es decir, del modo que aparecen en la web.

[5]http://www.cs.cmu.edu/ rcm/websphinx/
[6]http://wi.dii.uchile.cl

  - id_original_keyword: Identificador único de palabra almacenada.
  - word: La palabra propiamente tal.

- **Tabla : keyword**
  Esta tabla almacena las palabras clave una vez procesadas por el crawler a través del método de stemming.

  - id_keyword: Identificador único de palabra almacenada.
  - word: La palabra propiamente tal.
  - time: La marca de tiempo correspondiente a la detección de la palabra
  - id_original_keyword: Llave que hace referencia al identificador de la palabras original.

- **Tabla : page**
  Esta tabla almacena las URL de las páginas web pertenecientes al sitio.

  - id_page: Identificador único de la página almacenada.
  - url: La URL propia de la página.
  - language: Registro del lenguaje en forma abreviada, por ejemplo EN : inglés, ES : español.

- **Tabla : keyword_page**
  Esta tabla almacena la relación entre las palabras y las páginas, es decir, permite conocer qué palabras contiene cada página.

  - id_keyword_page: Identificador único de la asociación página-palabra.
  - id_keyword: Identificador de la palabra insertada. Hace referencia a la tabla **keyword**.
  - id_page: Identificador de la página insertada. Hace referencia a la tabla **page**.

- **Tabla : keyword_page_detection**
  Esta tabla almacena una marca de tiempo asociada a la aparición o desaparición de la asociación entre páginas y palabras. En otras palabras, permite conocer cuando una palabra apareció y/o fue eliminada de una página determinada.

  - id_keyword_page_detection: Identificador único.
  - id_keyword_page: Identificador de la asociación entre página y palabra. Hace referencia a la tabla **keyword_page**.
  - time: Marca de tiempo relativa a la primera vez que se identificó la asociación.
  - time_invalidation: Marca de tiempo relativa al instante en que la asociación palabra-página desapareció.
  - na: N^{o} de apariciones de la palabra en la página.
  - visivility_value: Valor asociado a la forma en la que la palabra es desplegada en la página. Por ejemplo, si está contenida dentro de tags *title*, *b* o *h1* se le asocia una importancia mayor que si no lo está.

- **Tabla : page_detection**
  Esta tabla almacena una marca de tiempo asociada a la aparición o desaparición de una determinada página.
  - id_page_detection: Identificador único.
  - id_page: Identificador página. Hace referencia a la tabla **page**.
  - time: Marca de tiempo relativa al instante en que se identificó la página.
  - time_invalidation: Marca de tiempo relativa al instante en que la página desapareció.
  - nw: Nº de palabras contenidas en la página.
  - language: Registro del lenguaje en forma abreviada, por ejemplo EN : inglés, ES : español.
  - title: Título de la página. Se obtiene procesando el tag *title*.
- **Tabla : link**
  Esta tabla almacena los enlaces existentes entre las páginas.
  - id_link: Identificador único.
  - id_page_from: Identificador de la página desde la que nace el enlace.
  - id_page_to: Identificador de la página destino.
  - frame_name: En caso de que el enlace sea produzca dentro de una página que contenga frames, se identifica el nombre del ésta.
- **Tabla : link_detection**
  Esta tabla almacena una marca de tiempo asociada a la aparición o desaparición de un determinado enlace entre páginas.
  - id_link_detection: Identificador único.
  - id_link: Identificador del enlace. Hace referencia a la tabla **link**.
  - time: Marca de tiempo relativa al instante en que se identificó el enlace.
  - time_invalidation: Marca de tiempo relativa al instante en que el enlace desapareció.
  - visibility_value: Peso asociado a cómo está dispuesto el enlace en la página.

4.2.2. Procesamiento y adaptación de los datos

Si bien al iniciar el proceso de extracción se configura el crawler con el objetivo de que sólo rescate documentos web (archivos html o php), se pudo apreciar que los registros obtenidos contenían, en una baja proporción, referencias a archivos de imágenes (extensiones .jpg y .gif principalmente) como también algunos archivos javascript (.js) y hojas de estilo (.css). Todos estos elementos fueron eliminados ya que no añaden valor al análisis. Por otro lado, tanto el proceso de stemming como el de eliminación de stop words fue realizado por el crawler en la fase de extracción.

Posteriormente se llevó a cabo una transformación de la estructura de tablas generadas con el fin de adaptarlas a los requerimientos del algoritmo. El principal motivo es facilitar el acceso a

los datos en función de la noción de *grafo por día*[7], es decir, dado un mes de extracción, se requiere saber tanto la estructura de links como el contenido exacto que tenía el sitio en cada día. A la par con esta transformación, se deben llevar a cabo los siguientes sub-procesos:

- Transformación del contenido a través de la utilización del Vector Space Model. Esto implica el cálculo del valor de TF-IDF.
- Dada una estructura de links, calcular las probabilidades de transición iniciales que servirán como valor de los trazos de feromona.

De esta forma se genera el modelo de datos presentado en la Figura 4.3.

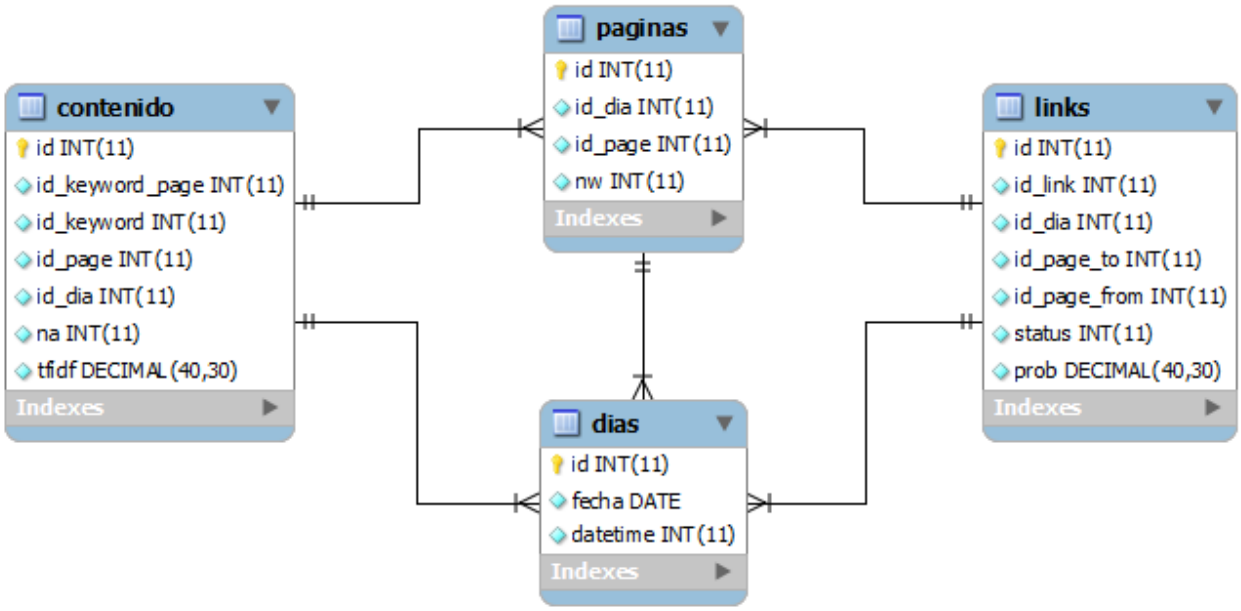

Figura 4.3: Modelo de datos auxiliar. Elaboración propia.

- **Tabla : dias**
  Esta tabla almacena una referencia a los días contenidos dentro del periodo de extracción. Resume los valores de time y time_invalidation.
  - id : Identificador único.
  - fecha: Marca de tiempo del inicio del día en formato *date*.
  - datetime: Marca de tiempo del inicio del día en formato *timestamp*.

[7] Ver Sección 3.2.2.

- **Tabla : paginas**
  Esta tabla almacena el conjunto de páginas presentan en cada día.
  - id : Identificador único.
  - id_dia : Identificador del día.
  - id_página : Identificador de la página.
  - nw : Número de palabras que contiene la página en el día determinado.
- **Tabla : contenido**
  Esta tabla almacena el conjunto de palabras que conforman el grafo para un día determinado.
  - id : Identificador único.
  - id_keyword_page: Identificador de la asociación palabra-página.
  - id_keyowrd: Identificador de la palabra.
  - id_page: Identificador de la página.
  - id_dia: Identificador del día
  - na: Número de apariciones de la palabra en la página asociada.
  - tfidf: Valor de TF-IDF de la palabra en la página.
- **Tabla : links**
  Esta tabla almacena los links presentes para cada día determinado.
  - id : Identificador único.
  - id_link: Identificador del enlace.
  - id_dia: Identificador de día.
  - id_page_from: Identificador de la página desde la que nace el enlace.
  - id_page_to: Identificador de la página destino.
  - status: Parámetro auxiliar que toma valor 1 si el link es válido, 0 si no.
  - prob: Valor de la probabilidad de transición. Sirve como parámetro inicial del nivel de feromona.

De esta forma, se tiene una representación completa tanto de la estructura de links como del contenido de la páginas a través del tiempo.

Para generar el conjunto de datos de entrenamiento se procesó durante el mes de septiembre de 2010, mientras que para el conjunto de validación se proceso durante el mes de octubre del mismo año.

4.3. Extracción de las Sesiones de Usuario

Al igual que para el proceso de extracción de contenido, se utilizaron los meses de Septiembre y Octubre de 2010. El primero será utilizado para el entrenamiento y el segundo para la validación del modelo.

El proceso de extracción y procesamiento del comportamiento del usuario se llevó a cabo mediante una estrategia proactiva que consiste en la utilización de un archivo javascript insertado en la mayoría de las páginas a analizar. Este archivo javascript (Track.js) es la pieza principal de un proceso de seguimiento y registro de las páginas que un usuario visita. El método se basa en la utilización de una cookie como medio de almacenamiento temporal de los datos de navegación con el fin de generar la identificación completa de cada sesión, la cual posteriormente es enviada a un servidor donde en almacenada definitivamente en una base de datos.

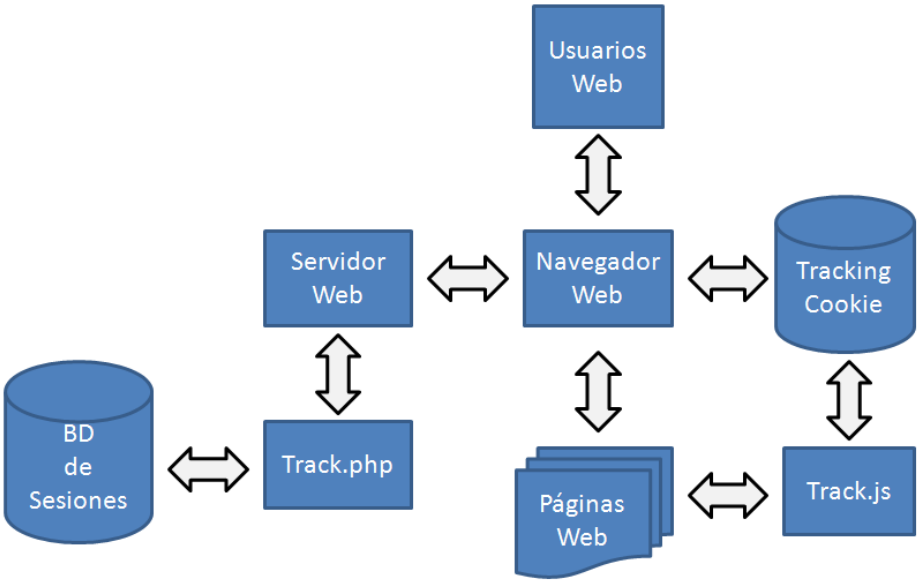

Figura 4.4: Esquema del almacenamiento de sesiones de usuario. Elaboración propia basado en [33].

Este método fue implementado por el profesor Pablo Román, y una explicación más profunda como también el código fuente puede ser encontrado en [33]. La principal ventaja de este método está en la generación *automática* del proceso de sesionalización. Esto tiene un valor elevado en relación con total de tareas a realizar, ya que reduce los tiempos de manera considerable. Cabe recordar que la prueba de concepto se utilizó el proceso de sesionalización tradicional desde los archivos web logs, lo cual implicó mucho más trabajo.

Las sesiones obtenidas son almacenadas de forma secuencial en la base de datos, donde se les asocia un identificador único. Con respecto a las dificultades que presenta el método, se pudieron observar, en los registros obtenidos, ciertas referencias a archivos de imágenes u hojas de estilo.

Otro de los problemas detectados tiene que ver con su dependencia de las funciones *onload* y *onbeforeunload* pertenecientes a Javascript, las cuales son usadas para identificar cuando un usuario entra y sale de una determinada página. Tales funciones presentan un comportamiento disímil dependiendo del navegador utilizado, por que fue necesario implementar un método heurístico corregir los valores defectuosos.

El modelo de datos asociado al almacenamiento de sesiones se muestra en la Figura 4.5.

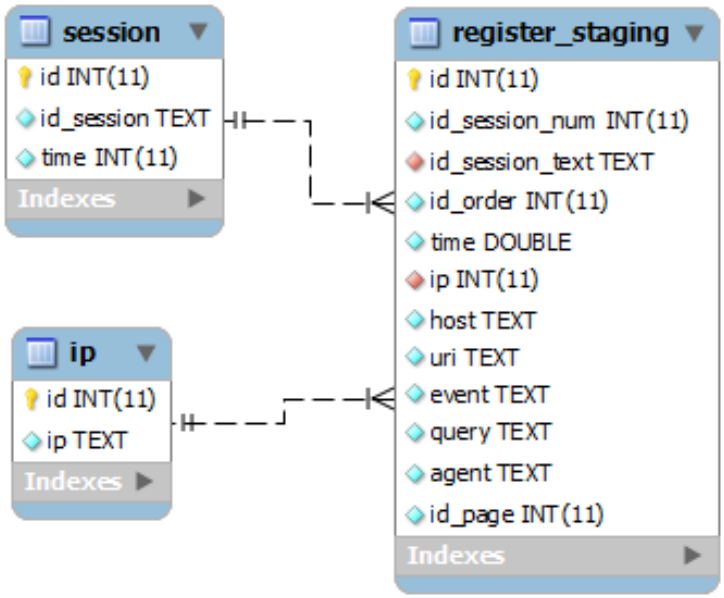

Figura 4.5: Modelo de datos almacenamiento sesiones. Elaboración propia.

- **Tabla : ip**
  Esta tabla almacena una la dirección IP del visitante y la asocia a un identificador único.
  - id : Identificador único.
  - ip: Dirección IP.
- **Tabla : session**
  Esta tabla almacena cada sesión identificada por el método de extracción utilizado y su respectiva marca de tiempo de comienzo.
  - id : Identificador único.
  - id_session : Identificador de la sesión.
  - time : Marca de tiempo del comienzo de la sesión.
- **Tabla : register_staging**
  Esta tabla almacena cada página visitada, la cual tiene asociado un identificador de sesión y el tiempo de visita.

- id: Identificador único del registro.
- id_session_num: Identificador único de la sesión.
- id_session_text: Identificador único de la sesión en formato *string*.
- id_order: Orden en que se encuentra la página dentro de la sesión.
- time: Marca de tiempo en la que fue requerida la página.
- ip: Dirección IP del visitante.
- host: Host asociado a la página visitada.
- uri: Dirección del documento (página) solicitado.
- event: Toma el valor IN si el usuario llega a la página, OUT si sale de ella.
- query: Almacena los parámetros de la url si se trata de un página dinámica.
- agent: Almacena el User-Agent propio del visitante.
- id_page: Asocia la página actual con algún identificador proveniente la tabla de páginas obtenidas con el web crawler.

En términos de la limpieza, se realizó una inspección de los registros y se pudo ver que si bien los datos en mayor porcentaje no presentaban problemas, una porción poseía ruido de las siguientes fuentes:

- Robots y web crawlers pertenecientes a buscadores. El método utilizado no fue capaz de identificar, a través del User-Agent, las visitas provenientes de bots.
- Inclusión de archivos auxiliares como imágenes, lo cual representó una fracción mínima.

Todos los registros defectuosos fueron eliminados a través de una estrategia que combinó el uso de expresiones regulares y búsqueda manual. En primer lugar, se llevó a cabo una inspección de las url presentes y a través de las funciones de agregación de SQL, se pudieron identificar direcciones no válidas (por ejemplo, referencias a buscadores) como también ciertas direcciones IP que dirigían al dominio principal dentro del DII. Es este caso, se tuvo que llevar a cabo un proceso de unificación. En segundo lugar, el atributo User Agent fue analizado con el fin de identificar usuarios artificiales. Cada registro se comparó con un conjunto de cadenas de texto que representan que contienen referencias a los bots o web crawlers más utilizados por los buscadores (por ejemplo : "googlebot", ƶahoo", .ªlexa",etc.).

4.3.1. Sincronización y Homologación de las fuentes de datos

Como se pudo ver, los procesos de extracción del contenido del sitio y del comportamiento del usuario se llevaron a cabo de manera independiente, aunque durante el mismo periodo de tiempo.

Una de las tareas más complejas del procesamiento de los datos resultó ser el encontrar la correspondencia entre los conjuntos de páginas provenientes de ambas fuentes. Se vio que el conjunto de páginas proveniente del crawler no contenía todas las páginas registradas en las sesiones. En promedio, considerando los dos meses, se registró que una ausencia del orden del 11 %. Esto se debe a que el crawler sigue estrictamente los caminos asociados a la estructura del links condicionado a un parámetro de *profundidad* que se configura, es decir, dado un nodo raíz, cuántos pasos debe avanzar dentro del sitio. Otra de las causas es la presencia de páginas islas, es decir, pertenecen al sitio pero no contienen links. Estas páginas pueden ser visitadas por los usuarios reales (si tienen una referencia explícita, por ejemplo páginas provenientes de servicios de *newsletter* o boletines), pero no pueden ser alcanzadas por el crawler. Se pudo observar un conjunto de páginas pertenecientes al sitio del MBA que poseían esta característica.

Lo que se busca es que todas las páginas presentes en las sesiones reales estén contenidas dentro del conjunto de las provenientes del crawler, debido a la necesidad de generar la correspondencia necesaria que exige el modelo de entrenamiento. La solución encontrada fue la siguiente:

1. Obtener todas las páginas que existen en las sesiones reales pero no en las obtenidas por el crawler a través de una consulta SQL.
2. Entregar este conjunto como nodos raíces al crawler a través de una estructura de arreglo.
3. Configurar el crawler para partir desde el conjunto entregado con el parámetro de profundidad en 1.

De esta forma se vuelve a recorrer el sitio y se rescatan las páginas faltantes. Se asume que no sufrieron cambios en todo el intervalo de tiempo.

Finalmente se lleva a cabo el proceso a través del cual se le asigna un identificador único a cada página existente el las sesiones reales. Para esto se utiliza el identificador que poseen las páginas en las tablas propias del crawler. Se realiza un matching a nivel de url entre ambos conjuntos siguiendo el esquema representado en la Figura 4.6.

De esta forma, para cada día de cada uno de los meses de extracción se tiene una sincronización entre las fuentes de datos.

De los repositorios finales se pueden extraer los siguientes indicadores:

- Estructura y contenido del sitio: Se almacenaron los datos para cada día dentro de los dos meses en estudio. En promedio, cada grafo contiene 2.510 páginas. Este número permanece casi constante, es decir, no se crearon ni eliminaron páginas. El número de links existentes va desde los 35.585 a los 35.594. Con respecto al contenido, se tienen en promedio 348.029 palabras por grafo. Las variaciones también son escasas a lo largo del tiempo, aproximadamente un 3 %. La página donde se observa un porcentaje mayor, es la página de inicio del sitio del DII, en la

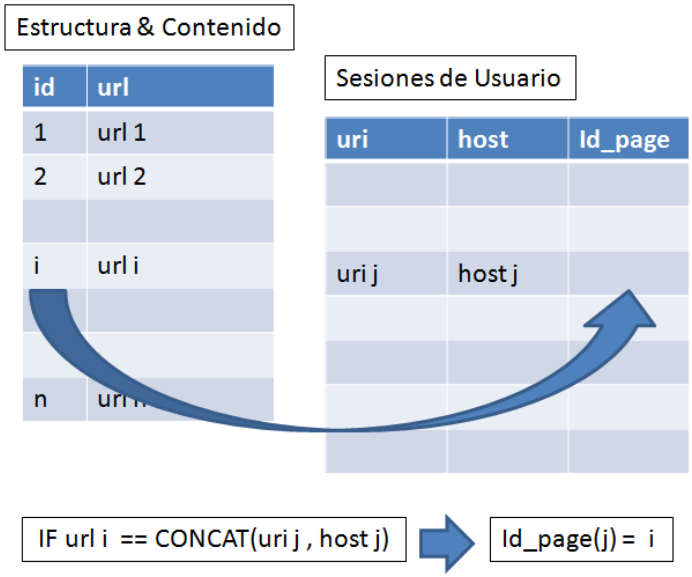

Figura 4.6: Homologación de identificadores de páginas. Elaboración propia.

cual se despliegan las noticias, con una variación promedio de aproximadamente 20 %. En esta página, el contenido está dividido en dos formatos. Por una parte, se muestra la introducción de cada noticia en formato html y el resto del contenido está en un archivo con extensión .pdf, formato no considerado por el web crawler.

- Sesiones de usuario: En el primer mes de estudio se recolectaron un total de 18.876 sesiones, cuyos comportamientos siguen el patrón estándar predicho por la *Ley de Navegación*, mencionada en la sección del Marco Conceptual, tal como se puede apreciar en la figura 4.7.

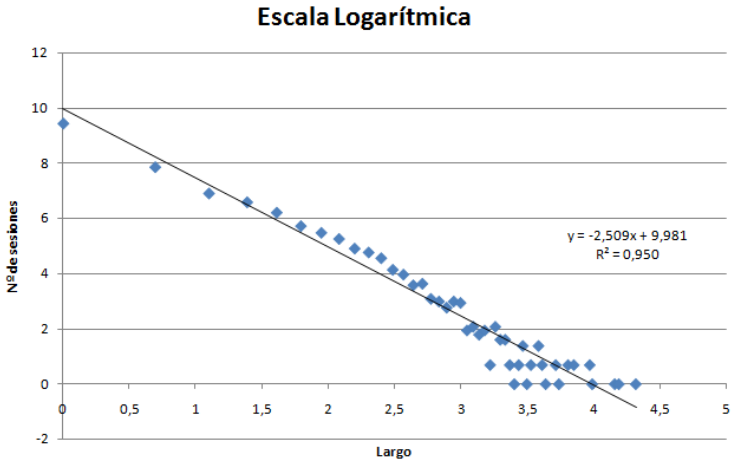

Figura 4.7: Distribución en escala logarítmicas de los largos de sesión. Elaboración propia.

El segundo mes contiene un total de 14.296 sesiones y su comportamiento sigue el mismo patrón anterior.

4.4. Entrenamiento del Modelo

La labor de entrenamiento se llevó a cabo utilizando los datos recolectados del primer mes en estudio, es decir, Septiembre de 2010.

4.4.1. Clustering de sesiones reales

En primer lugar se eliminaron todas las sesiones de largo 1, debido a que no representan interés en el sentido de no contener la noción de camino secuencial. Luego, quedaron 6.282 sesiones.

Posteriormente se ejecuta el algoritmo de clustering jerárquico. Una de las características de este tipo de clustering es que se puede visualizar en cada iteración cuales son los clusters que se van generando a través del particionamiento. Luego, surge la pregunta : ¿Cómo determinar la cantidad de clusters óptima?

Hay que notar que en el diseño del algoritmo se definió en lugar de una *distancia*, una *medida de similitud* entre las sesiones. Luego, al implementar el algoritmo se vio que era necesario y más intuitivo considerar una distancia entre las sesiones, cuya definición se presenta en la Ecuación 4.1.

$$distancia(i,j) \quad = \quad 1 - sim_d(i,j), \tag{4.1}$$

donde i, j son dos sesiones y $sim_d(i,j)$ es el grado de similitud entre ellas definido por la Ecuación 3.7 en 3.2.1.

Así, para calcular el número de clusters óptimos, se utilizó la razón entre la distancia intra-clusters y la distancia extra-clusters. Se fue iterando hasta identificar el mínimo valor de la razón anterior, lo cual se traduce en la elección de 806 clusters para las 6282 sesiones.

Si bien se puede percibir que el número es elevado, no se debe olvidar que la utilización del clustering tiene como objetivo reducir la dimensionalidad del problema del comparación, pero sin sacrificar la calidad de la similitud entre las sesiones. En ese sentido, dada la alta exigencia de la medida de similitud utilizada (el la cual no sólo se compara los elementos conformantes de las sesiones sino también el orden en que se encuentran), es comprensible el algoritmo sea reticente a realizar un número elevado de uniones. Posteriormente, dentro de cada cluster identificado, se calculó la sesión

representativa, o *centroide*, como el elemento que posee la máxima similitud acumulada dentro de cada grupo.

El algoritmo de clustering fue escrito en lenguaje Java, en donde las primeras pruebas se realizaron utilizando los objetos Vector[8] y ArrayList[9] para gestionar el almacenamiento temporal. La ventaja de este tipo de clases es la versatilidad y facilidad de implementación, por ejemplo, el no tener que definir la estructura de dato con un largo fijo. Pero el principal problema que se experimentó tiene que ver con los tiempos relativos a la ejecución.

De este modo se llevó el algoritmo a una estructura de arreglos comunes, los cuales, si bien son más rígidos, permiten una mayor velocidad.

El tiempo total que tomó el proceso de clustering de las sesiones del mes de entrenamiento fue de aproximadamente 30 minutos.

4.4.2. Ejecución del Algoritmo de Aprendizaje

A cada uno de los 806 clusters encontrados se le asoció una hormiga artificial,la cual a su vez fue inicializada con con un vector μ generado artificialmente. Los valores de TF-IDF de las keywords del mes de entrenamiento fueron calculados y posteriormente normalizados con el fin de que estuvieran contenidos entre 0 y 1. Luego se seleccionaron todas las keywords cuyo valor de TF-IDF está entre 0.6 y 0.9 y éstas fueron almacenadas dentro del conjunto de las palabras más importantes, el *ConjuntoMostImportantKeywords* definido en el capitulo anterior. Este conjunto comprendió 165 elementos. De esta se generaron los vectores L de cada página como también los vectores μ, de tamaño 165.

Se llevaron a cabo múltiples pruebas del algoritmo con diferentes parámetros, con el fin explorar su comportamiento, la calidad de sus soluciones y los tiempos asociados. En términos de la convergencia, se pudo apreciar la relevancia del factor de evaporación, el cual inicialmente había sido subestimado asignándole un valor inferior a 0.5, finalmente se tuvo que dejar casi en 1. Del mismo modo, el valor del valor modificador de feromona tuvo que ser progresivamente incrementado con el fin de permitir una diferenciación real entre los caminos más frecuentes y los que no eran utilizados.

Finalmente se llegó a la siguiente configuración:

- ε (reforzador) = 2.2
- *factor de evaporación* = 0.8
- *límite utilidad de las hormigas* = 0.19

[8] http://download.oracle.com/javase/1.4.2/docs/api/java/util/Vector.html
[9] http://download.oracle.com/javase/1.4.2/docs/api/java/util/ArrayList.html

- *modificador feromona del camino elegido* = 5.5
- *boost (premio por elección de link contenido en sesiones)* = 0.2

Cada una de las 806 hormigas iteró 1000 veces sobre sus propios grafos. Este proceso se configuró de tal forma que corrieran paralelamente un cierto número de hormigas con el fin de acortar los tiempos. Se intentó, a través de a librería MASON[10], generar las 806 instancias simultáneas, pero las características del equipo donde se corrió no lo permitieron. Luego, se ejecutó el algoritmo en segmentos de 100 hormigas cada uno.

El equipo utilizado se compuso de un procesador Intel Core 2 Duo 2.1 Ghz, 3Gb RAM bajo Windows 7 Professional Edition de 32 bits y el algoritmo fue escrito en Java.

El tiempo promedio de ejecución para entrenar las 806 hormigas fue de aproximadamente 4 horas, haciendo uso casi total de la memoria virtual del sistema. El output del algoritmo es el conjunto de vectores μ entrenados. Los resultados que se obtuvieron muestran cambios significativos en los valores que toman los vectores entrenados en comparación con los que inicialmente tenían.

Cabe recordar que los vectores μ iniciales fueron generados en base valores aleatorios que seguían una proporción determinada y que simulaban la preferencia del usuario por cada palabra. Así, el 20 % de las palabras tenía asociado un valor entre 0.8 y 1, lo que representaba alto interés. El 60 % de las palabras tenía valores entre 0.2 y 0.8, representando interés medio y finalmente el 20 % restante tenía valores entre 0 y 0.2, representando bajo interés. Los resultados obtenidos muestran que, para cada vector, a medida que transcurre el tiempo, la mayoría de los valores sufren caídas drásticas, mientras que una minoría sube niveles.

Las Figuras 4.8 y 4.9 muestran el estado inicial y final el vector μ_{10}, es decir, el asociado a la hormiga (y al cluster) 10, respectivamente.

Como se puede apreciar, la heterogeneidad inicial en los valores desaparece luego de 1.000 interaciones de entrenamiento. Los valores altos se asocian a las palabras clave que caracterizarían a las sesiones del cluster. Este comportamiento se repite para todos los vectores μ. Cabe señalar que en la mayoría de los vectores, los valores mayores se tienden a concentrarse en los primeros componentes. Una explicación para esto, es que las palabras en los vectores texto L de las páginas están ordenadas en forma decreciente en función de sus valores de TF-IDF, luego el entrenamiento tiende a reforzar los valores en esa zona.

[10]http://cs.gmu.edu/ eclab/projects/mason/

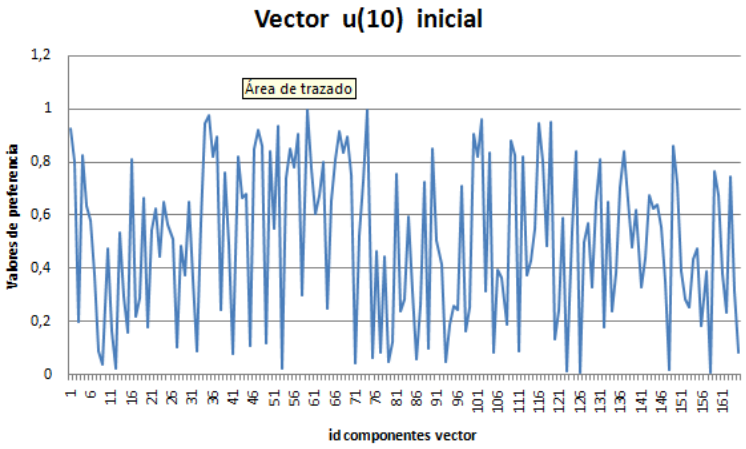

Figura 4.8: Estado inicial del vector μ_{10}.

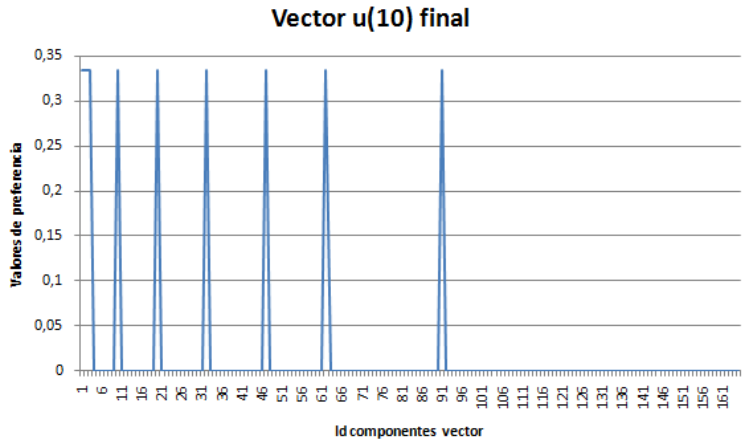

Figura 4.9: Estado final del vector μ_{10}.

4.5. Validación del Modelo de Aprendizaje

Para validar el modelo se utilizaron los datos provenientes del segundo mes. En éstos se realizó el mismo procedimiento de clustering de las sesiones reales. Los resultados obtenidos entregaron 733 clusters para las 5701 sesiones con largo mayor a 1. Posteriormente se calcularon los *centroides asociados a cada cluster*. Para cada uno de los vectores entrenados μ, se generó un conjunto de 200 hormigas (valor arbitrario).

Para cada conjunto de hormigas se generó un grafo con los datos pertenecientes al segundo mes. En este caso se tomó el grafo correspondiente a un día arbitrario dado los casi nulos cambios tanto en estructura como en contenido. Adicionalmente se calcularon los niveles iniciales de feromona en cada link. Posteriormente, cada grupo de hormigas fue liberado en su respectivo grafo y se

ejecutó el algoritmo de validación. Tal como se mostró en el Capítulo 3, para cada grafo, las hormigas, ahora con una preferencia por texto definida, comenzaron a moverse generando rutas y alterando los niveles de feromona en los links, reforzando caminos a través del uso de la estigmergia.

Este algoritmo fue escrito en Java y es el que utilizó la librería multiagente MASON en sus mayores capacidades. Para cada uno de los conjuntos de hormigas, las sesiones generadas fueron almacenadas en una tabla temporal.

Para evaluar la evolución de las sesiones generadas se definió el indicador:

$$comparador(t) \quad = \quad sim_d(sesion(t), sesion(t-1)), \tag{4.2}$$

es decir, compara la similitud de cada sesión generada con la que fue generada justo en el instante anterior, independiente de qué hormiga lo hizo. Esto porque se asume que el conjunto de agentes funciona como un sólo organismo.

A continuación se muestra la evolución de las similitudes entre sesiones para el conjunto de hormigas asociadas a μ_{10}:

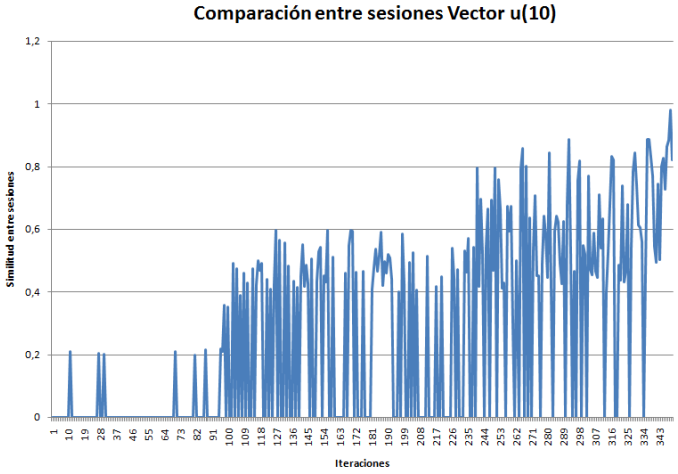

Figura 4.10: Comparación secuencial de sesiones para μ_{10}

Como se puede ver, los valores van aumentando a medida que se tienen más iteraciones. Inicialmente las similitudes de las sesiones eran casi nulas, pero luego comienza a *afirmarse* en un valor medio, para posteriormente, cercano a la iteración 350, llegar a una similitud casi total (cercano a 1).

Inicialmente se deseaba establecer un umbral de similitud, sobre el cual todas las sesiones generadas serían catalogadas como *homogéneas*. Pero, una exploración a los contenedores con sesiones generadas por diversos grupos muestra que los comportamientos son demasiado variables, por lo que no tendría sentido establecer un único valor. De esta forma se decidió, para cada grupo

de hormigas, tomar las últimas 50 sesiones generadas, teniendo como supuesto de que al final del algoritmo es donde se alcanza la convergencia en términos de las rutas.

De esta forma, cada uno de los 806 grupos de hormigas tiene asociado un conjunto de 50 sesiones. Estos conjuntos reciben el nombre de *Conjuntos Factibles*. En cada *conjunto factible* se identifica un valor central, que consiste en la sesión que posee el nivel de similitud acumulado mayor.

Luego, se pasa a la fase final que consiste en realizar una comparación entre los valores centrales de los *conjuntos factibles* y los *centroides* de las sesiones reales en el mes de validación.

En primer lugar se fijó una similitud mínima sobre la cual se puede decir que un valor central explica un centroide, y por lo tanto el cluster. Tal límite, dado que la medida de similitud va entre 0 y 1, se fijó en 0.6. Posteriormente, se tomó cada valor central proveniente de las sesiones generadas por las hormigas y se iteró sobre el conjunto de centroides de las sesiones reales, calculando el valor de la similitud.

Los resultados obtenidos muestran que de los 806 valores centrales, 611 poseen al menos una similitud mayor o igual a 0.6 con alguno de los centroides de las sesiones reales.

Si se deja el límite de similitud en 0.7, la cifra disminuye a 484, mientras que si se deja en 0.5, la cifra sube a 639. Es decir, subiéndole la exigencia a la comparación se sufre una variación mayor a lo que sucede bajándola.

Ahora, es necesario visualizar la proporción de clusters que fueron explicados. Cabe señalar que se tienen 733 clusters de sesiones reales. Realizando la comparación desde los clusters a los valores centrales, se tiene que, para un límite de 0.6, 598 centroides están relacionados con los valores centrales de las sesiones artificiales. Si el límite se eleva a 0,7, se obtienen 452 centroides que cumplen la condición, mientras que si se rebaja a 0.5, se obtienen 601.

Como se puede apreciar, los valores no son análogos en ambos sentidos, esto se debe a que hay un grupo de valores centrales que apunta a un sólo centroide, es decir, hay una proporción de los clusters de sesiones reales que está siendo explicado por más de un grupo de hormigas. Luego, el nivel de explicación, tomando como umbral de similitud igual a 0.5, llega aproximadamente al 81 %. Es decir, las hormigas artificiales entrenadas, son capaces de, a través de un proceso dinámico de construcción de sesiones, poder simular de una manera bastante cercana el comportamiento de los usuarios reales. El tiempo asociado a esta comparación fue de aproximadamente 30 minutos.

Capítulo 5

Conclusiones

El uso de la librería multiagente MASON representó una gran utilidad en relación con la gestión eficiente del comportamiento de un número considerable de agentes que interactúan entre sí haciendo un uso intensivo de los recursos del sistema. En ese sentido, se puede decir que al momento de implementar metodologías de modelamiento a considerable escala, es necesario tener en cuenta tanto las características de hardware como de software involucradas, ya que esta información permite llevar a cabo cambios que si bien pueden parecer pequeños, pueden representar mejores relevantes, lo cual repercute en la calidad de los resultados. Uno de los puntos a destacar en términos del procesamiento general de datos dentro de Web Mining, es que los volúmenes asociados superan con creces lo que se visualiza en los problemas de Minería de Datos tradicional. Esto representa un desafío bastante interesante, ya que la capacidad de los paquetes y librerías que son utilizados usualmente se ve sobrepasada. Luego, los investigadores y profesionales del área deben idear estrategias que involucren tanto adaptaciones como trabajo en conjunto de distintas herramientas con el fin de lograr obtener el conocimiento deseado. Lo anterior tiene especial importancia en las fases de preprocesamiento de los datos, ya que la Web, al ser un ambiente eminentemente heterogéneo, no posee una estandarización natural tanto del contenido como de la estructura.

Pasando a analizar el proceso de modelamiento utilizado, se puede decir que una de las características principales de la metaheurística de ACO es la libertad que le proporciona al diseñador para atacar los problemas. Si bien esto tiene un lado positivo, asociado a la ausencia de rigideces excesivas y de limitaciones conceptuales, al mismo tiempo presenta la desventaja de que no existe una *guía* o procedimiento estandarizado para abordar los problemas. Sólo existen ciertos elementos intrínsecos derivados de su naturaleza biológica que se deben respetar, tales como la noción de *estigmergia*. Esta simpleza obliga a experimentar tratando de evaluar las opciones e ideas que permitan adaptar este método al problema en estudio, dando un valor especial a la experiencia y *know how*. La elaboración de una Prueba de Concepto resultó ser de gran utilidad en términos de facilitar la exploración y experimentación con diversas propiedades y herramientas que posteriormente fueron utilizadas en el diseño del modelo.

Uno de los aspectos más relevantes al momento de la construcción del modelo es la naturaleza multivariada de Web Mining. Se puede concluir que en el proceso del estudio y análisis del comportamiento del usuario en la Web, convergen las dimensiones asociadas al contenido, estructura y uso cuya interacción permite explicar los fenómenos que se presentan. En ese sentido, entre más completas sean las fuentes de datos, mejor análisis se podrá realizar.

Entre la complejidad de Web Intelligence y la relativa simpleza metodológica y operacional de ACO se genera una relación que debe manejarse con precaución, en el sentido de no esperar poder implementar un procedimiento de simulación de comportamiento de usuario utilizando solamente la formulación convencional multiagente. Es ahí donde el término *metaheurística* cobra sentido en relación con la generación de un *framework* que sea adaptable a los problemas. Es así como fue necesario realizar múltiples adiciones y cambios en la formulación estándar que proporciona ACO para lograr incluir todas las variables que el problema de simulación requería.

En relación con el modelo, la principal característica radica en que recibe como input el contenido, la estructura y el uso de un sitio, lo cual le proporciona una versatilidad y la capacidad de incorporar cualquier elemento dentro del ambiente Web al proceso de simulación. Como desventaja asociada, se requiere tanto capturar como procesar una cantidad considerable de datos, los cuales no siempre están disponibles.

Por otro lado, se tiene que el proceso de aprendizaje basado en la construcción de vectores de preferencia presenta un buen desempeño en términos de condicionar las sesiones a generar. Es decir, diferentes preferencias por texto (representadas a través de diferentes vectores), generan, a largo plazo, sesiones diferentes utilizando un mismo grafo web. Los tiempos asociados podrían verse como un problema, pero se estima que una mejor implementación a nivel operacional podría reducirlos.

Uno de los aspectos relevantes del modelo es la forma en que se llevan a cabo las comparaciones entre las sesiones de usuario. En este trabajo se propuso no sólo comparar en términos de qué páginas las componen, sino también en analizar el orden. Este enfoque permitió incorporar la noción de camino secuencial, sobre la cual subyace el concepto de búsqueda. La implementación de esta medida derivó en la no utilización de las sesiones de largo igual a 1, lo cual, si bien influye en términos de identificar las distribuciones de largo de sesión, no castiga al objetivo principal del modelo, el cual es recrear los caminos y las decisiones que toman los usuarios al contrastar su preferencia de texto con el contenido propio de las páginas.

Los resultados obtenidos muestran que las sesiones artificiales generadas por el modelo pueden explicar aproximadamente el 81 % del uso real en un intervalo de un mes. El proceso de comparación se realizó en base a una agregación de los conjuntos de sesiones, tanto artificiales como reales, midiendo el nivel de similitud entre sesiones características de los conjuntos. Esta metodología permite por una lado agilizar el proceso, debido a la disminución de comparaciones, y por otro, establecer los resultados como tendencias de navegación en lugar de una predicción individual de comportamiento. Esto se condice con la filosofía detrás de ACO en relación a la construcción colaborativa por parte de múltiples agentes que trabajan a nivel microscópico para generar un patrón a nivel macroscópico.

La utilidad de este modelo está relacionada con el nivel de apoyo que pueda representar para las estrategias de marketing asociadas a determinados sitios y la necesidad de conocer y tener una estimación de las conductas de los usuarios. De esta forma, se puede conocer a priori los patrones de navegación y se pueden llevar a cabo acciones para mejorar el nivel de fidelización. Otra aplicación práctica está relacionada con el poder probar ciertos cambios en la estructura de un sitio o contenido del sitio. Por ejemplo, se pueden realizar modificaciones en una versión *off line* y ejecutar el modelo con el fin de simular y evaluar el comportamiento que seguirían los usuarios o cómo sería su respuesta frente a dichos cambios. Por último, la relación entre el comportamiento y los vectores de preferencias por texto podría ser utilizado como input para algoritmos que elijan y desplieguen dinámicamente publicidad.

5.1. Trabajo Futuro

Uno de los puntos característicos del sitio web en el cual se implementó el modelo es su poco dinamismo. La extracción continua de la estructura y el contenido mostró que los cambios que se realizaron durante el intervalo en estudio fueron muy bajos. Esto es entendible dada la misión eminentemente informativa que posee y la naturaleza de la información que muestra, la cual, al tener un carácter *corporativo* y de divulgación, no varía frecuentemente. Dado lo anterior, se propone como tarea futura probar el modelo sobre una plataforma web que si tenga un nivel de cambios considerables, tanto a nivel de contenido como de la propia estructura de links presentes. De esta forma se podría explorar la capacidad de ajuste del proceso de entrenamiento en función de su versatilidad frente a las modificaciones más drásticas.

Con respecto al modelo desarrollado, se podrían realizar cambios relativos a mejorar la simulación del usuario en el sentido de permitir más grados de libertad. En primer lugar, tratar de simular el límite de utilidad asociado a cada hormiga. En la implementación actual está fijo y de igual valor para todos los agentes. Lo ideal sería que tuviera un componente dinámico siguiendo el supuesto de que cada persona tiene una valoración y un límite de utilidad asociado a su navegación. En segundo lugar, se propone la construcción de nuevas medidas de similitud entre sesiones y su estudio comparado en función de encontrar mejores ajustes.

Una de las etapas más complejas dentro del presente trabajo resultó ser el preprocesamiento de los datos, tanto en términos de la estructura y contenido del sitio como también del comportamiento de los usuarios. Dada la poca estructuración de las fuentes de datos en la Web, la preparación y estandarización toma un papel preponderante dentro del proceso KDD aplicado a la Web. En ese sentido se propone continuar con el desarrollo de métodos que progresivamente sean capaces de unificar las fuentes en una sola plataforma que sea capaz de consolidar los datos y que garantice la *integridad referencial* tan necesaria para obtener resultados coherentes. Derivado del punto anterior se propone trabajar en relación con generar un *dataset* para Web Mining, es decir, una colección de datos que sea útil para realizar pruebas sobre cualquier modelo predictivo que se aplique a la Web, tal como los existentes para los modelos de Minería de Datos tradicional, lo que facilitaría el trabajo de estudiantes e investigadores.

REFERENCIAS

[1] Ajith Abraham and Vitorino Ramos. Web usage mining using artificial ant colony clustering and linear genetic programming. In *Genetic Programming, Congress on Evolutionary Computation (CEC), IEEE*, pages 1384–1391. Press, Australia, 2003.

[2] Evelyn Andaur. Análisis del comportamiento del usuario en la web para optimizar la estructura de navegación de un sitio usando algoritmos genéticos. *Memoria para optar al título de Ingeniera Civil Industrial, Universidad de Chile.*

[3] John R. Anderson. The adaptive nature of human categorization. *Psychological Review*, 98:409–429, 1991.

[4] Luca Becchetti, Carlos Castillo, Debora Donato, and Adriano Fazzone. A comparison of sampling techniques for web graph characterization.

[5] L. Bergroth, H. Hakonen, and T. Raita. A survey of longest common subsequence algorithms. In *Proceedings of the Seventh International Symposium on String Processing Information Retrieval (SPIRE'00)*, pages 39–, Washington, DC, USA, 2000. IEEE Computer Society.

[6] Eric Bonabeau, Marco Dorigo, and Guy Theraulaz. *Swarm intelligence: from natural to artificial systems.* Oxford University Press, Inc., New York, NY, USA, 1999.

[7] Cristián Bravo, Sebastián Maldonado, and Richard Weber. Experiencias prácticas en la medición de riesgo crediticio de microempresarios utilizando modelos de credit scoring. *Revista de Ingeniería de Sistemas*, 24(1):69–88, June 2010.

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

[9] J. L. Deneubourg, S. Goss, N. Franks, A. Sendova Franks, C. Detrain, and L. Chrétien. The dynamics of collective sorting robot-like ants and ant-like robots. In *Proceedings of the first international conference on simulation of adaptive behavior on From animals to animats*, pages 356–363, Cambridge, MA, USA, 1990. MIT Press.

[10] Marco Dorigo and Luca Maria Gambardella. Ant colonies for the traveling salesman problem. *Biosystems*, 1997.

[11] Marco Dorigo and Thomas Stützle. *Ant Colony Optimization*. MIT Press, 2004.

[12] Nadav Eiron and Kevin S. Mccurley. Locality, hierarchy, and bidirectionality in the web. In *In Workshop on Algorithms and Models for the Web Graph*, 2003.

[13] Gartner. Visitado en diciembre 2010. `http://www.gartner.com/it/page.jsp?id=1454221`.

[14] Dan Gusfield. *Algorithms on String, Trees and Sequences. Computer Science and Computational Biology*. Cambridge University Press, 1999.

[15] Bernardo A. Huberman, Peter L. T. Pirolli, James E. Pitkow, and Rajan M. Lukose. Strong regularities in world wide web surfing, 1997.

[16] Avrim Blum Hubert, T h. Hubert, Chan Mugizi, and Robert Rwebangira. A random-surfer web-graph model. In *In ANALCO 2006: Proceedings of the eigth Workshop on Algorithm Engineering and Experiments and the third Workshop on Analytic Algorithmics and Combinatorics*, page 238246, 2006.

[17] R. Kimball and R. Merz. *The Data Webhouse Toolkit: Building the Web-Enabled Data Warehouse*. Wiley and Sons, 2000.

[18] Chang-Chun Lin. Optimal web site reorganization considering information overload and search depth. *European Journal of Operational Research*, 173(173), November 2006.

[19] Chang-Chun Lin and Lu-Chuan Tseng. Website reorganization using an ant colony system. *Expert Systems with Applications*, 37(12):7598 – 7605, 2010.

[20] Haifeng Ling, Yezheng Liu, and Shanlin Yang. An ant colony approach for discovery of users preferred navigation paths. In *PACIS 2004 Proceedings*, 2004.

[21] Bing Liu. *Web DataMining: Exploring Hyperlinks,Contents and Usage Data*. Springer-Verlag Berlin Heidelberg, 2007.

[22] Sean Luke, Claudio Cioffi-revilla, Liviu Panait, Keith Sullivan, and Gabriel Balan. Mason: A multi-agent simulation environment, 2005.

[23] Erik D. Lumer and Baldo Faieta. Diversity and adaptation in populations of clustering ants. In *Proceedings of the third international conference on Simulation of adaptive behavior : from animals to animats 3: from animals to animats 3*, pages 501–508, Cambridge, MA, USA, 1994. MIT Press.

[24] G. J. Myatt. *Making Sense of Data: A Practical Guide to Exploratory Data Analysis and Data Mining*. Wiley and Sons, 2007.

[25] M. Perkowitz and O. Eizioni. Adaptive web sites: An ai challenge. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI-97)*, pages 39–21, Nagoya, Japan, 1997. Morgan Kaufmann.

[26] Peter L. T. Pirolli. *Information Foraging Theory: Adaptive Interaction with Information (Oxford Series in Human-Technology Interaction)*. Oxford University Press, USA, 1 edition, April 2007.

[27] Sriram Raghavan and Garcia-Molina H. Representing web graphs. In *Proceedings of the 19th International Conference on Data Engineering*, pages 405–416, 2003.

[28] Vitorino Ramos, Fernando Muge, and Pedro Pina. Self-organized data and image retrieval as a consequence of inter-dynamic synergistic relationships in artificial ant colonies. In *HIS*, pages 500–512, 2002.

[29] Vitorino Ramos, Fernando Muge, and Pedro Pina. Self-organized data and image retrieval as a consequence of inter-dynamic synergistic relationships in artificial ant colonies. In *HIS*, pages 500–512, 2002.

[30] Victor Rebolledo. Plataforma para la extracción y almacenamiento del conocimiento extraído de los web data. *Master Thesis, Universidad de Chile*, 2008.

[31] Pablo E. Román and Juan D. Velásquez. The time course of the web user. In *Second Workshop on Time Use Observatory (TUO2)*, San Felipe, Chile, 2010.

[32] Pablo Román. Apuntes clases auxiliares del curso web mining. *Departamento de Ingeniería Industrial, Universidad de Chile*, 2009.

[33] Pablo Román. Web user behavior analysis. *PhD Thesis, Universidad de Chile*, 2011.

[34] Herbert A. Simon. The architecture of complexity. In *Proceedings of the American Philosophical Society*, pages 467–482, 1962.

[35] Herbert A. Simon. *Designing Organizations for an Information-Rich World*, pages 37–72. Press, 1971.

[36] Chakrabarti Soumen. *Mining the Web. Discovering knowledge from hipertext data.* Morgan Kaufman Publishers, 2003.

[37] Myra Spiliopoulou, Bamshad Mobasher, and Bettina Berendt Miki Nakagawa. A framework for the evaluation of session reconstruction heuristics in web-usage analysis. *Journal of Computing*, 15(2):171–190, 2003.

[38] J. D. Velásquez and L. Donoso. Aplicación de técnicas de web mining sobre los datos originados por usuarios de páginas web. visión crítica desde las garantías fundamentales, especialmente la libertad, la privacidad y el honor de las personas. *Revista de Ingeniería de Sistemas*, 24(1):47–68, June 2010.

[39] Juan D. Velásquez and Lakhmi C. Jain. *Advanced Techniques in Web Intelligence.* Springer-Verlag Berlin Heidelberg, 2010.

[40] J. D. Velásquez and V. Palade. *Adaptive Web sites: A Knowledge Extraction from Web Data Approach.* IOS Press, 2008.

[41] Juan D. Velásquez. Aplicación de técnicas de web mining sobre los datos originados por usuarios de páginas web. visión crítica desde las garantías fundamentales, especialmente la libertad, la privacidad y el honor de las personas. *Tesis para optar al grado de Magíster en Derecho de la Informática y de las Telecomunicaciones, Facultad de Derecho, Departamento de Derecho Procesal. Universidad de Chile.*

[42] Tony White, Amirali Salehi-Abari, and Braden Box. On how ants put advertisements on the web. In *IEA-AIE 2010: Proceedings of the 23rd International Conference on Industrial, Engineering & Other Applications of Applied Intelligent Systems*, Cordoba, Spain, May 2010.