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

DISEÑO Y CONSTRUCCIÓN DE UN DATA MART PARA EL FILTRO DE OPINIONES EN LA WEB A PARTIR DE DATOS ORIGINADOS EN EL PORTAL EDUCAR CHILE

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL INDUSTRIAL.

JAIME ARIAS CUEVAS.

PROFESOR GUÍA:

JUAN D. VELÁSQUEZ SILVA

MIEMBROS DE LA COMISIÓN:

SEBASTIÁN RÍOS PÉREZ

ÁNGEL JIMÉNEZ MOLINA

SANTIAGO DE CHILE

JUNIO 2012

RESUMEN DE LA MEMORIA
PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL INDUSTRIAL
POR: JAIME ARIAS CUEVAS
LUNES 27/06/2012
PROF. GUÍA: SR. JUAN VELÁSQUEZ SILVA

DISEÑO Y CONSTRUCCIÓN DE UN DATA MART PARA EL FILTRO DE OPINIONES EN LA WEB, A PARTIR DE DATOS ORIGINADOS EN EL PORTAL EDUCAR CHILE

El objetivo general del trabajo de título es diseñar y construir un *Data Mart* que permita obtener indicadores de uso de los escritorios del portal educarchile.

Educarchile es un portal administrado por la Fundación Chile, y que fue creado por esta última en conjunto con el Ministerio de Educación. Su misión es contribuir al mejoramiento de la calidad de la educación en todos sus niveles, para lo cual cuenta con un sitio dirigido a todos los miembros de la comunidad educativa nacional. Para esto, el diseño del sitio se basa en un *Home* y 4 escritorios enfocados en cada segmento de usuarios, los que constan de secciones que sirven de enlace al contenido del portal.

Educarchile, con el objetivo de obtener información acerca del comportamiento de sus usuarios, trabaja con dos herramientas de pago, Certifica y Google Analytics. Sin embargo, debido al tamaño del portal y el dinamismo del contenido publicado en sus escritorios, no obtiene de estas herramientas información acerca de las preferencias que tienen los usuarios respecto a las secciones de aquellas páginas, y las llamadas *viñetas* que las componen, que son recursos que permiten la publicación de contenido bajo la restricción que algunas sean visualizadas solo si se hace un click sobre ellas. Adicionalmente, el sitio permite la emisión de opiniones en los artículos, sin embargo, no existe ningún tipo de alerta o filtro para las publicaciones que no se ajustan al clima y objetivo del portal.

La hipótesis del trabajo plantea que a través de la creación de indicadores limpios y consolidados respecto del uso de las secciones y viñetas que componen el portal, y que se almacenarán en un Data Mart, el equipo de administración del sitio podrá acceder a información detallada acerca del comportamiento de sus visitantes, la que no ha sido obtenida hasta hoy.

Para llevar a cabo el trabajo, se diseñó una arquitectura que permite la extracción y el procesamiento de los datos, además de su posterior carga en un repositorio multidimensional, el que funciona como fuente de datos para consultas OLAP. La arquitectura consta de 3 elementos principales: los modelos de datos; el proceso de extracción, transformación y carga de los datos; y un modelo para clasificar y filtrar opiniones, basado en el algoritmo *Naive Bayes.* Para cada elemento se optó por la utilización de herramientas gratuitas.

Los indicadores obtenidos a través del procesamiento de los archivos *weblog* entregaron información desconocida y valiosa al equipo del portal. Uno de los principales resultados fue: comprobar que las viñetas que componen las secciones de los escritorios producen un alto sesgo en el comportamiento de los usuarios, principalmente en aquellas secciones que contienen información de actualidad. En ellas los usuarios no visualizan los recursos que son publicados en las viñetas que se encuentran ocultas por defecto, lo que se traduce en una política ineficiente de edición y publicación de artículos. Por su parte, el algoritmo *Naive Bayes* obtuvo un alto índice de recall para aquellas clases que se deseaba predecir (*ayuda* y *planificación)*, que en ambos casos supera el 85%. Sin embargo, la clase que representa el resto de los comentarios tiene un menor recall, habiendo un 30% de las opiniones clasificadas erróneamente.

Como conclusión, el modelo propuesto es capaz de satisfacer las necesidades de información de la organización, entregando conocimiento útil a la hora de evaluar y definir nuevas políticas de publicación de contenidos que se ajusten a las reales preferencias de los usuarios. A pesar de aquello, se recomienda realizar una nueva medición de los indicadores una vez efectuados cambios en el diseño de las páginas, para así obtener resultados contundentes que permitan identificar las preferencias de diseño y contenido por parte de los usuarios. Además, se recomienda implementar en el sitio el modelo obtenido para las opiniones, y así detener la publicación de comentarios que no aportan valor al sitio.

Agradecimientos

En primer lugar, quisiera agradecer a mis padres por el enorme apoyo y cuidado que me dieron durante toda mi formación académica, entregándome valores de los que me siento orgulloso, y dándome la oportunidad de acceder a una buena educación.

También agradecer a mis 6 hermanos, quienes tuvieron mucho que ver en este largo proceso de aprendizaje.

Agradecer también a quien me acompañó estos 6 años de universidad, siendo un pilar emocional en este importante periodo.

A Juan Antonio, con quien fue un gusto desarrollar mi trabajo de título.

Y por último, agradecer al profesor Juan Velásquez por darme la oportunidad y confianza de desarrollar con él mi trabajo de título, entregando consejos que no solo me guiaron en este proceso, sino que también son de ayuda en el día a día.

Dedicado a mi tío Carlos Cuevas Moya, a quien espero honrar con mi título en esta facultad.

Índice de contenidos:

CAPÍTULO 1 – INTRODUCCIÓN

La Web es un canal que crece continuamente, tanto en el número de usuarios como en la cantidad de sitios. En este contexto de fuerte competencia, como medio de información, publicidad y ventas; las organizaciones se han visto en la obligación de destinar recursos para poder captar y retener clientes a través de sus portales. Por esto, las diversas entidades se ven en la necesidad de saber cómo se comportan sus clientes, qué es lo que buscan, lo que valorizan, cuán difícil es lograr encontrar lo que buscan, entre otros.

El análisis de un sitio web permite predecir las necesidades de los usuarios, evaluar el impacto del portal, y guiar las mejoras, tanto en su contenido, diseño y estructura. Esto suele ser desarrollado a través del análisis de los *clickstreams,* los que son transformados en una gran cantidad de datos almacenados en los archivos *weblogs*, y que permiten conocer qué es lo que visitó el usuario en cada sesión, información que es utilizada por técnicas ligadas al Data Warehousing y al Web Mining, con la finalidad de encontrar nuevo conocimiento valioso sobre el comportamiento de navegación del usuario [35].

Por otro lado, con el desarrollo de la Web 2.0, que se caracteriza por dar la posibilidad de que los usuarios colaboren en la publicación de contenido, se han incrementado los portales que contienen opiniones [14], las que son entendidas como estados que no son abiertos a la observación o verificación objetiva [28]. Con ello ha nacido el interés por conocer las opiniones generales de los usuarios sobre determinados temas, y así poder clasificarlas a partir de su contenido.

1.1. DESCRIPCIÓN DEL PROYECTO

Educarchile trabaja con herramientas de pago que le permiten obtener indicadores acerca del comportamiento de sus usuarios. Sin embargo, estas herramientas no le permiten conocer las preferencias en los servicios que son ofrecidos en las páginas principales, lo que se traduce en la imposibilidad de conocer a cabalidad cómo el usuario se comporta con respecto a los elementos que están a su disposición.

El equipo de análisis de educarchile no ha podido responder preguntas acerca del interés que despiertan en los usuarios los contenidos publicados en las portadas de cada escritorio[1], los que están a cargo de un equipo de edición. Debido a esto, no han logrado evaluar, tanto la estrategia de publicaciones que llevan a cabo, como tampoco el gasto de recursos monetarios y humanos producto de la actualización constante del contenido.

Por otro lado, el portal cuenta con secciones dentro de los escritorios donde los usuarios pueden dejar su opinión. Sin embargo, hoy no existe ningún

[1] El sitio cuenta con 4 páginas principales llamadas escritorios, las que están diseñadas para publicar contenido, tanto a docentes, estudiantes, directivos, y familiares.

procesamiento ni análisis automático de los comentarios que se publican en los distintos artículos. Esto produce que en muchas ocasiones existan opiniones que no se ajustan al clima de respeto que se da y que incentiva educarchile. De hecho, es posible encontrar insultos hacia instituciones y personas públicas, como también hacia usuarios, mayoritariamente en la sección de debate del portal. Adicionalmente, es usual encontrar a usuarios solicitando ayuda en artículos de contenido educativo, tanto acerca de la búsqueda de material, como de información acerca de la Prueba de Selección Universitaria (PSU). E incluso, en páginas relacionadas a planificaciones de clases, se observa la existencia de un *mercado*, donde usuarios solicitan, compran, y venden planificaciones para distintas asignaturas.

El trabajo de título constará de 4 elementos que permitirán abordar el problema descrito: una investigación en el área de repositorios de información web [37], en donde se diseñarán los modelos de almacenamiento de datos; una investigación respecto a las metodologías actuales que permiten realizar filtros de opiniones, con la finalidad de aplicarlos sobre alguna sección de interés del sitio; la aplicación sobre el sitio de educarchile de la metodología diseñada; y el desarrollo de un prototipo de interfaz de consulta para el repositorio de información web.

1.2. OBJETIVOS

1.2.1. OBJETIVO GENERAL

Diseñar y construir un Data Mart para la obtención de indicadores de uso del portal educarchile, como también para la clasificación de opiniones publicadas en el sitio, basada en el algoritmo *Naive Bayes*.

1.2.2. OBJETIVOS ESPECÍFICOS

1. Revisar el estado del arte en repositorios de información y clasificación de opiniones en un sitio web.
2. Definir los indicadores claves que la organización desea que sean medidos.
3. Adaptar y aplicar una metodología que permita obtener indicadores de desempeño del portal y filtros de las opiniones de los usuarios.
4. Construir un prototipo de interfaz de consulta para los indicadores.
5. Obtener un modelo que permita identificar los comentarios que soliciten ayuda, o que son parte del mercado de las planificaciones dentro del sitio.

1.3. RESULTADOS ESPERADOS

Se espera que el trabajo al menos obtenga los siguientes resultados:

- Obtener un prototipo de interfaz de consulta al usuario final, que preste indicadores de interés que hoy no puede obtener a través de las herramientas de pago que se ocupan.
- Elaborar un modelo que permita clasificar las opiniones que los usuarios publican en el sitio.

1.4. METODOLOGÍA

El trabajo pretende cubrir dos problemáticas que se dan en el portal, por lo que la metodología debe ser capaz de permitir alcanzar los objetivos propuestos para cada problema. Para conseguirlo se combinarán las metodologías de diseño y construcción de repositorios de información web [37], con las metodologías que permiten clasificar opiniones vertidas en la Web [26].

1.4.1. REQUERIMIENTOS DE INFORMACIÓN

1. Indicadores.

El Data mart deberá responder a necesidades de información del usuario final, por lo que el primer paso debe ser conocer estas necesidades. Esto será llevado a cabo a través de entrevistas con los usuarios del data mart.

2. Seleccionar las fuentes de datos.

El Data mart trabajará con las siguientes fuentes:

- Los archivos *weblog*, que permitirán almacenar los datos que representan el comportamiento de los usuarios en el portal.
- El código HTML de las páginas, el que permitirá almacenar el contenido y la estructura de hyperlinks del portal. Además, se deben extraer las opiniones vertidas en las distintas paginas del sitio.

1.4.2. DISEÑO DE LOS MODELOS DE DATOS Y DEL PROCESO ETL

3. Desarrollar el modelo lógico de datos.

Para esto, se deberá diseñar un modelo lógico que almacene los datos provenientes de las fuentes de información. Este modelo es conocido como el *Data Staging Area*. Posteriormente se deberá diseñar un modelo de información basado en la arquitectura Data Mart. Para esto utilizara como base el modelo estrella.

4. Diseñar el proceso de extracción, transformación y carga de los datos.

Se utilizarán algoritmos que sean capaces de extraer los datos desde las fuentes, llevar a cabo una limpieza para filtrar la información que no resulte relevante para el estudio, y transformar los datos, tanto en sesiones de navegación, como en

token's que representen el contenido de las opiniones, para finalmente realizar la carga de los datos en el repositorio de información.

5. Elegir el Data Base Manager System.

Este paso es importante, ya que incidirá directamente en el rendimiento del repositorio de información desarrollado, desde el punto de vista de la capacidad medida en tiempos para responder una consulta.

1.4.3. IMPLEMENTACIÓN DE LOS MODELOS

6. Implementar el modelo lógico en una base de datos.

Corresponde a la implementación física del punto 3 de la metodología.

7. Cargar la información y evaluar el modelo.

En este paso se lleva a cabo la implementación del proceso ETL sobre los datos.

1.4.4. WEB OPINION MINING

8. Aplicar algoritmos de minería de datos sobre las opiniones.

Finalmente, a partir de los datos almacenados en el repositorio de información web, se aplicará el algoritmo *Naive Bayes,* el que permitirá obtener un filtro de las opiniones vertidas por los usuarios en el portal.

1.5. ALCANCES DEL TRABAJO

El presente trabajo pretende obtener un prototipo para el usuario final que sólo muestre indicadores de los 4 escritorios del portal, más un set de 10 páginas relevantes para la administración. Esto debido a que el sitio cuenta con una cantidad considerable de secciones que no permiten el levantamiento de requerimientos sobre ellas durante el tiempo destinado para desarrollar el trabajo.

Además no está dentro de los alcances obtener una plataforma que trabaje online. Esto debido a los plazos involucrados, y las capacidades técnicas de las herramientas que se utilizarán.

Por último, debido que el análisis de las opiniones está restringido por la semántica de los temas, el análisis sólo será realizado para las clases *Ayuda* y *Planificación.*

1.6. CONTRIBUCIONES

El trabajo expuesto a continuación obtuvo como resultado un prototipo funcional instalado en las dependencias de la Fundación Chile, basado en el software Jasper Server. El prototipo permite la consulta de todos los indicadores

relacionados con el uso del portal que fueron medidos a través del procesamiento de los archivos weblog, y almacenados en un Data Mart. La información levantada permitió, tanto a la dirección de educarchile como al equipo de análisis de usuarios, tomar decisiones acerca del diseño de las páginas más relevantes del sitio, y que fueron parte del estudio, como también establecer nuevas políticas de publicación de contenido. Adicionalmente, se firmó un convenio entre el Departamento de Ingeniería Industrial de la Universidad de Chile y educarchile, el que permitirá desarrollar nuevos estudios relacionados con el portal, enmarcados en trabajos de título de estudiantes del departamento.

CAPÍTULO 2 – MARCO CONCEPTUAL

En este capítulo se abordarán los temas y conceptos relacionados con la extracción de conocimiento a partir de datos originados en la Web. Para esto, se comienza con un estudio de la Web y su funcionamiento. Luego se introduce la tecnología del *Data Warehousing*, desde donde es posible extraer información a partir de datos. Posteriormente se ahondará en los *Data Warehouse's* aplicados a datos de la Web, cuya área de estudio es el *Web Warehousing*. Y finalmente se estudiará el *Web Mining*, área de especialización del *Data Mining* que tiene por objetivo encontrar patrones a partir de datos de la Web.

2.1. WORLD WIDE WEB

Desde 1989, el británico Tim Berners Lee lideró en el CERN[2] el desarrollo de lo que denominó World Wide Web. Su motivación era entregar un universo global de información utilizando la tecnología existente hasta ese entonces. La investigación que realizó incluyó dos grandes tópicos del conocimiento humano que abarcan los computadores: el hipertexto y la recuperación de texto [5].

En su origen, el modelo de la Web era un grafo dirigido, que usaba tanto el paradigma de los links de hipertexto, como el de la búsqueda de texto, donde ninguno podía reemplazar al otro [5]. En el modelo, los nodos representan documentos, tales como noticias, investigaciones o notas, y los arcos son hyperlinks, que representan las referencias entre los documentos. Esto queda ilustrado en la Figura 1.

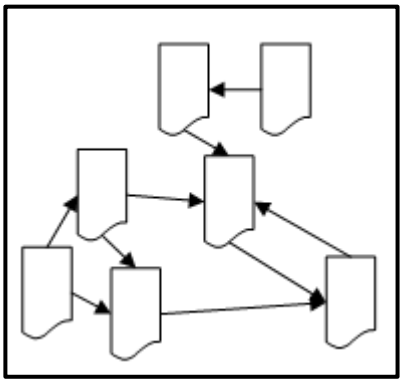

Figura 1: La Web como un grafo dirigido. Elaboración propia.

A pesar de sus interesantes características, fue el desarrollo del Internet lo que permitió la masificación de la Web [37]. Con el Internet, los documentos

[2] CERN: European Organization for Nuclear Research

dejaron de ser compartidos tan sólo por computadores, ya que también podían traspasarse a través de distintas redes.

2.1.1. FUNCIONAMIENTO

El funcionamiento de la Web está basado en el modelo cliente/servidor [3]. Consiste en que el cliente solicita objetos a un servidor web, el que es identificado a través de su dirección URL[3]. Para lograr comunicarse con el servidor, el cliente debe realizar la petición a través del protocolo que soporte el servidor web, siendo HTTP[4] el más común.

Luego que el cliente realiza la petición del objeto al servidor, lo que suele realizarse a través de un navegador web, recibe de vuelta una respuesta en código HTML [5] . Este lenguaje interpretado permite visualizar en forma estandarizada el contenido del objeto, y está basado en etiquetas.

Un componente importante en la interacción de los usuarios con los servidores web son los proveedores de internet (ISP), que son quienes permiten conectar el sistema que utiliza el usuario web con el Internet. Para esto, los ISP utilizan sus servidores *proxi*, los que se conectan a *carriers* tradicionales, como las redes telefónicas, redes de televisión por cable, o redes satelitales. Así, el usuario envía peticiones a los servidores del ISP, quien se ocupa de enviarlos a través de internet a los servidores web correspondientes [34].

Como ya se mencionó, los servidores web del portal son los que se ocupan de recibir las peticiones de los usuarios, y luego satisfacerlas. Sin embargo, cuando se reciben una gran cantidad de peticiones, un solo servidor web no es capaz de satisfacer las peticiones de los usuarios. Por esta razón, muchos portales deben utilizar servidores web replicados, los que funcionan en paralelo, y permiten satisfacer una alta demanda de peticiones [34].

[3] URL: Uniform Resource Locator, usado para localizar recursos en internet.
[4] HTTP: Hypertext Transfer Protocol. Protocolo usado para las transacciones en la Web.
[5] HTML: Hypertext Markup Language

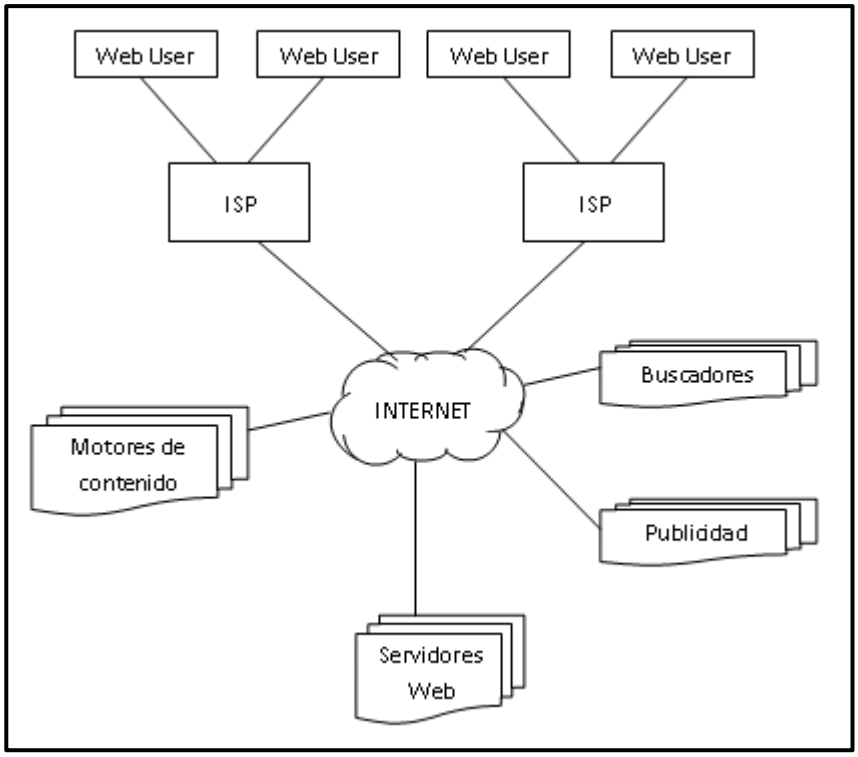

Figura 2: Funcionamiento de la Web. Fuente: [34]. Elaboración propia.

2.1.2. DATOS ORIGINADOS EN LA WEB

Las peticiones hechas por los usuarios al servidor web del sitio, son almacenadas en los archivos *weblog*. Estos archivos de texto pueden ser interpretados como una matriz, donde cada fila corresponde a una petición hecha al servidor, y cada columna representa a un atributo.

Los weblog siguen un estándar dictado por la W3C[6], en donde se define la estructura de los datos almacenados. Estos datos suelen ser:

- Fecha y hora de la petición.
- Dirección IP del cliente.
- Tipo de requerimiento.
- Archivo solicitado (host dentro del servidor).
- Status de la solicitud, representado a través de un código.
- Referrer, que corresponde al documento desde el cual se realizó la petición.
- User-agent, correspondiente al software utilizado por el cliente para realizar la petición (comúnmente un navegador web).

[6] W3C: World Wide Web Consortium. Consorcio internacional que realiza recomendaciones acerca de la Web

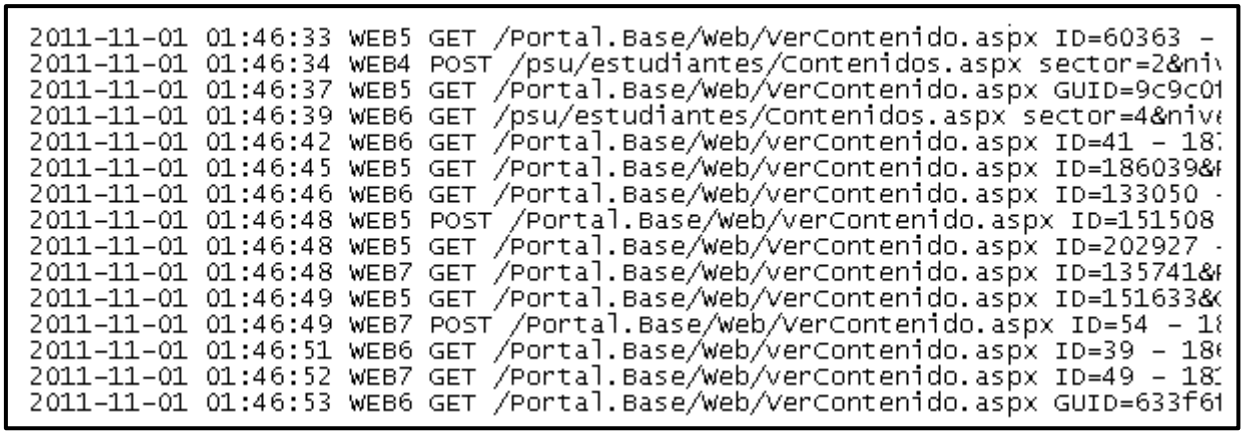

Figura 3: Ejemplo de un archivo weblog. Elaboración propia.

Los weblogs representan datos valiosos para la organización, ya que corresponden a la mejor encuesta sobre el uso que sus clientes dan al sitio web [37].

2.2. REPOSITORIOS DE INFORMACIÓN

Los repositorios de información nacieron como sistemas que aprovechaban los datos desde los distintos sistemas operacionales, con el objetivo de ayudar con información confiable y oportuna al proceso de toma de decisiones. A continuación, se presentan las principales temáticas relacionadas con los repositorios de información, para introducir el concepto de Data Warehousing.

2.2.1. MODELO RELACIONAL

En 1970, el matemático inglés Edgar Codd presentó una nueva teoría que permitía el almacenamiento masivo de datos, la que dio paso al modelo relacional, que se basa en las operaciones del álgebra matemática. El modelo permitió separar la capa de datos de las aplicaciones que los manejaban, además de posibilitar al usuario definir de forma estructurada la forma en que se organizan los datos [8].

El modelo relacional, propuesto por Codd, está basado en relaciones n-arias entre distintas entidades, las que a su vez poseen atributos. Dentro de sus características principales se encuentran sus variadas formas de evitar la redundancia entre las relaciones, y la posibilidad de asegurar la consistencia en la base de datos. Además, permite responder prácticamente cualquier tipo de consulta.

2.2.2. DATA WAREHOUSE

En la década de los 90', los usuarios se enfrentaban a múltiples problemas derivados de querer obtener información desde sus bases de datos operacionales.

Entre estos problemas se encontraba la heterogeneidad de los datos desde las distintas fuentes; que las bases de datos se encontraban diseñadas para consultas cortas y predecibles, y que era costoso agregar datos desde distintas tablas relacionales [6].

Ante aquellos problemas, se dio paso a la arquitectura *Data Warehousing,* que es un conjunto de tecnologías que tienen por objetivo permitir al analista tomar decisiones mejores y rápidas [7]. Algunas de las ventajas de esta arquitectura son la mejora en el rendimiento, una mejor calidad de los datos, y la posibilidad de consolidar y resumir datos desde distintos sistemas [23].

Una de las principales funcionalidades de un Data Warehouse, es el actuar como fuente de datos, tanto para el analytical online processing (OLAP), como también para aplicar técnicas de Data Mining [16]. Así, la consolidación de datos operacionales de interés, le permitirá a la empresa obtener indicadores sobre predeterminadas consultas, como también patrones sobre los datos

2.2.2.1. Objetivos del Data Warehousing

En las organizaciones es común que en distintas áreas ligadas a la toma de decisiones se encuentren problemas relacionados con el manejo de grandes volúmenes de datos. Los ejecutivos suelen toparse con temas como “tenemos montañas de datos, pero no podemos acceder a ellos”, debido a la gran cantidad de tiempo que consume obtener un reporte basado en bases de datos operacionales. También suelen mencionar “sólo muéstrame lo importante”, en el sentido que los reportes muchas veces cuentan con datos o información que no resulta útil para la toma de decisiones [18].

Teniendo en cuenta los problemas universales de las organizaciones derivados de obtener información a partir de datos operacionales, en Spiliopoulou y otros[17] se propone que un Data Warehouse persigue los siguientes objetivos:

- **El Data Warehouse debe hacer fácilmente accesible la información de la organización.**
  La información contenida en el Data Warehouse debe ser entendible e intuitiva para el usuario de negocios, y no sólo para el desarrollador. Esto debe traducirse en que los datos estén bien etiquetados, que las herramientas que utilicen el repositorio sean de fácil acceso, y que los tiempos que toman las consultas sean breves.
- **El Data Warehouse debe presentar información consistente de la organización.**
  La información contenida debe ser de alta calidad y confiable. Para esto, el procesamiento de los datos desde las distintas fuentes operacionales debe ser llevado a cabo con cuidado, y se necesita asegurar que toda la información sea bien etiquetada para evitar que se produzcan problemas de interpretación.
- **El Data Warehouse debe ser adaptativo y resistente al cambio.**
  Debido a que en el tiempo son constantes los nuevos requerimientos de información en las organizaciones, el Data Warehouse debe ser diseñado

para poder incorporar cambios en el futuro. Una vez que se incluyen, la información existente no debe ser alterada, sin importar los nuevos datos y aplicaciones en el repositorio.

- **El Data Warehouse debe proteger de forma segura la información.**
  El repositorio contiene información importante del negocio de la organización, por lo que se debe controlar de manera efectiva el acceso a la información, para así evitar que caiga en manos de personas erradas.
- **El Data Warehouse debe servir como fundamento en la toma de decisiones.**
  El repositorio debe contener la información correcta para la toma de decisiones, debido a que será utilizada como evidencia en aquel proceso.
- **El Data Warehouse debe ser aceptado por la comunidad para que sea exitoso.**
  Sin importar si la solución desarrollada cumple con todo lo esperado, si el repositorio no es utilizado por los usuarios de negocios, entonces el Data Warehouse no será exitoso.

2.2.2.2. Modelos de análisis multidimensional

Usualmente quienes toman decisiones en las empresas intentan analizar los datos desde distintas dimensiones. Por ejemplo, una consulta recurrente es "cuántas unidades del producto X se vendieron durante los días Y a través del distribuidor Z". En esta pregunta el encargado de tomar decisiones requiere saber sobre las ventas en las dimensiones producto, tiempo y distribuidor. Esta consulta tan común muestra la multidimensionalidad de los datos que se necesitan para facilitar la toma de decisiones.

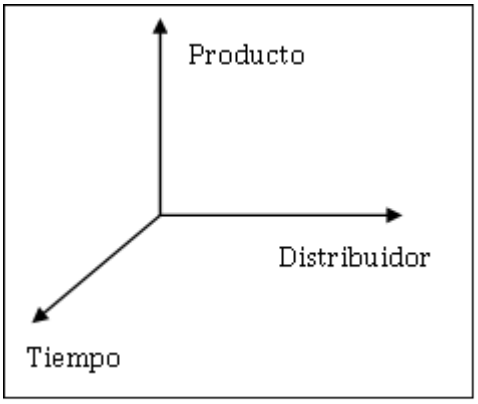

Figura 4: Multidimensionalidad de las consultas. Elaboración Propia.

El Modelo Multidimensional de Datos (MDM) [37] provee una vía para poder realizar consultas que agreguen datos desde distintas dimensiones de manera más eficiente que el modelo relacional. En concreto, es una técnica que

conceptualiza los modelos de datos como un conjunto de medidas que describen los aspectos comunes de la empresa [4]**.**

El MDM es útil a la hora de resumir y ordenar los datos para apoyar el análisis de información. Para lograr esto, se focaliza en datos numéricos, como es el valor de un elemento; o en funciones numéricas, como contar elementos, el peso porcentual, la suma de los valores, entre otros.

El MDM tiene tres conceptos básicos: los hechos, las dimensiones, y las medidas.

Hechos (Fact)

Un hecho corresponde a una colección de elementos relacionados, que consiste en las medidas, y en datos que contextualizan las dimensiones. Los hechos permiten representar elementos del negocio, como pueden ser una venta o una compra, o cualquier evento que pueda ser utilizado en el análisis de procesos de negocio [4].

En un Data Warehouse, los hechos son registrados en la *Fact Table*, que corresponde a la tabla central del modelo de datos. La lista de las dimensiones que se toman en cuenta en esta tabla define el grano utilizado, el cual debe ser igual para todas las medidas de una Fact Table [18].

En términos de la utilidad de las *Fact Table's,* son las númericas y aditivas las que permiten obtener resultados más interesantes. Esto porque las consultas hechas sobre un Data Warehouse consideran miles o millones de registros, y usualmente lo más útil es sumar los valores de sus medidas. Por ejemplo, cuánto dinero vendió la sucursal 1 de la región 4. Sin embargo, no siempre esto es posible, ya que existen medidas que son semi-aditivas o no-aditivas. Las semi-aditivas permiten sumarse sólo en algunas dimensiones, y las no-aditivas no permiten hacerlo en ningún caso. Así, la mayor utilidad de estas últimas medidas es el contar los registros que cumplen con cierta condición [18].

Dimensiones

Una dimensión corresponde a una colección de miembros del mismo tipo. En el modelo multidimensional, cada registro de la *Fact Table* es asociado a un y sólo un miembro de cada dimensión considerada en la tabla. Así, son las dimensiones las que permiten contextualizar el hecho que se registra en la *Fact Table* [4].

Las dimensiones son los parámetros sobre los que se realiza el *Online Analytical Processing* (OLAP). Por ejemplo, para el caso de la Figura 4, las consultas sobre las ventas serán hechas sobre las dimensiones *Tiempo, Producto* y *Distribuidor.*

En el *Data Warehouse*, cada dimensión está representada por una tabla, la cual posee varios atributos que caracterizan de manera única cada miembro de la dimensión. Usualmente a cada dimensión se le asocian tantos atributos útiles como sea posible, lo que provoca que en muchas ocasiones existan tablas con más de 50 atributos [18].

Las dimensiones son el elemento vital de un *Data Warehouse*, ya que permiten comprenderlo y hacerlo usable. *"El poder de un Data Warehouse es directamente proporcional a la calidad y profundidad de los atributos de las dimensiones"* [18].

A diferencia de los hechos, para las dimensiones los mejores atributos son los textuales y discretos. Esto porque nombres reales permiten al usuario de negocios comprender mejor las dimensiones, en comparación con las abreviaciones o valores numéricos.

Medidas

Una medida es un atributo numérico que caracteriza a un hecho, representando el rendimiento o comportamiento de la empresa en relación a las dimensiones [4]. Por ejemplo, una medida para un foro en la Web es la cantidad de post, o el número de temas creados. Las medidas se encuentran en la *Fact Table*.

Por otro lado, existen dos enfoques para implementar el MDM: el modelo cubo y el modelo estrella, cuya principal diferencia está en el tipo de motor de bases de datos utilizado.

Modelo cubo

Corresponde a una implementación del MDM en bases de datos multidimensionales (MDBMS), las que fueron creadas para este enfoque. Su nombre se debe a que su diseño se puede interpretar como un cubo de n-dimensiones, lo que permite hacer consultas eficientes a través de sus caras [37].

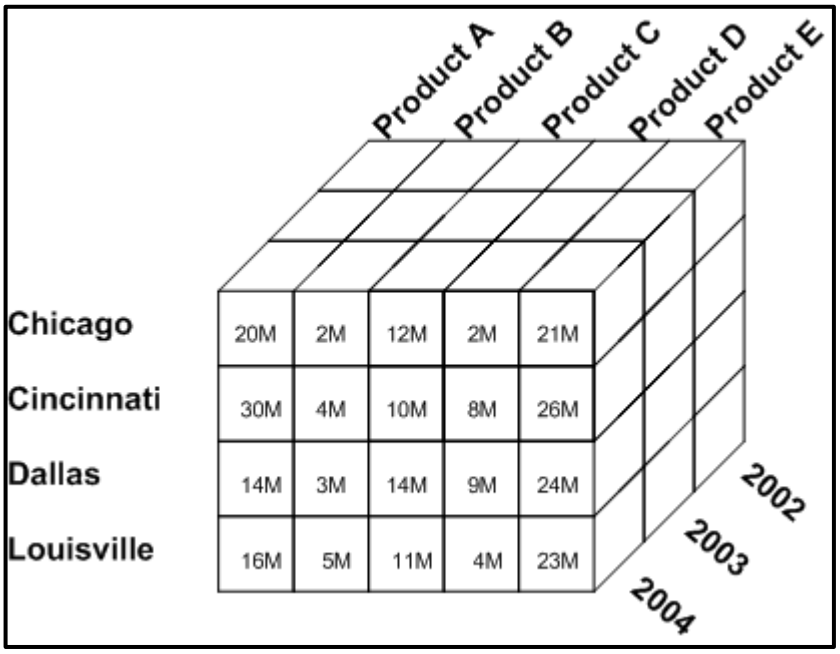

Figura 5: Ejemplo de un cubo multidimensional. Fuente: Oracle.

Un cubo de información permite realizar las siguientes operaciones:

- Pivoting: rotar el cubo y mostrar la cara deseada.
- Slicing: seleccionar una dimensión particular del cubo.
- Dicing: seleccionar una o más dimensiones del cubo.
- Drill-down: disminuye la jerarquía en un nivel.
- Roll-up: aumenta la jerarquía en un nivel.

La Figura 5 muestra un ejemplo de un cubo multidimensional, aplicado sobre datos de una empresa de retail. Usualmente, los cubos asociados a las ventas cuentan con dimensiones como fecha, lugar, sucursal, y producto, entre otras que pueden resultar de interés. A la vez, las medidas que se suelen considerar son la cantidad de ventas, y el monto en dinero asociado a ellas. En este caso, la vista del cubo muestra 3 dimensiones: producto, a nivel de nombre; fecha, a nivel de año; y lugar, a nivel de ciudad. A la vez, la medida considerada es la cantidad de ventas. Sin embargo, a través de las operaciones *drill-down* y *roll-up*, se pueden obtener visualizaciones para otros niveles de las dimensiones.

Computacionalmente, la representación de un cubo es a través de arreglos multidimensionales, lo que trae consigo bastantes ventajas conceptuales. Sin embargo, un MDBMS requiere de importantes recursos, para implementar una alta dimensionalidad, además del uso de memoria virtual, lo que se debe sumar a posibles problemas técnicos del sistema operativo que soporte el MDBMS [37].

Otra importante desventaja del modelo cubo, es que necesita utilizar bases de datos multidimensionales, las que son de alto costo monetario en comparación a las bases de datos relacionales, que son las utilizadas para las distintas operaciones en las empresas. Esto, sumado a las desventajas mostradas en el párrafo anterior, hace la implementación del modelo cubo impracticable [37].

Modelo estrella

Corresponde a la implementación del MDM en bases de datos relacionales. Consiste en representar las dimensiones y medidas en distintas tablas relacionadas entre sí. El modelo consta de una tabla central llamada *Fact Table*, donde se deben almacenar medidas del negocio, con un nivel de detalle pre-definido, que es conocido como granularidad. Además, se deben incluir atributos que representen a las dimensiones, a través de llaves foráneas que las apunten [37].

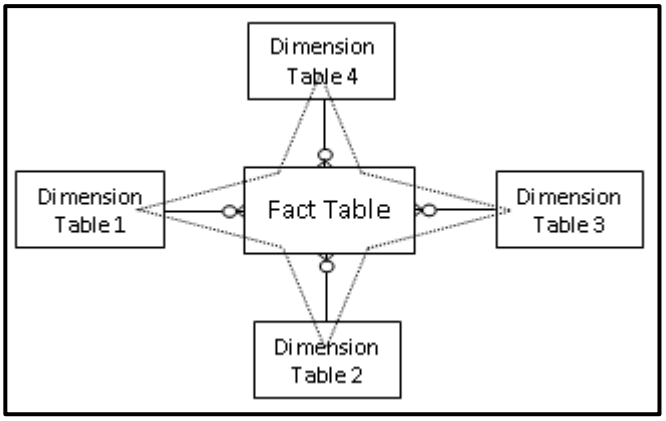

Figura 6: Modelo estrella genérico. Fuente: [24]. Elaboración propia.

El modelo también necesita la existencia de tablas que representen a las dimensiones, donde los atributos permiten caracterizarlas a través de valores numéricos y textuales. Por ejemplo, para la dimensión *Date* es usual registrar el número de la semana del año, como también el día de la semana en formato texto. La Figura 7 muestra un ejemplo concreto para el modelo estrella, donde los datos provenientes de las ventas son analizados a través de las dimensiones cliente, producto, tiempo, y sucursal.

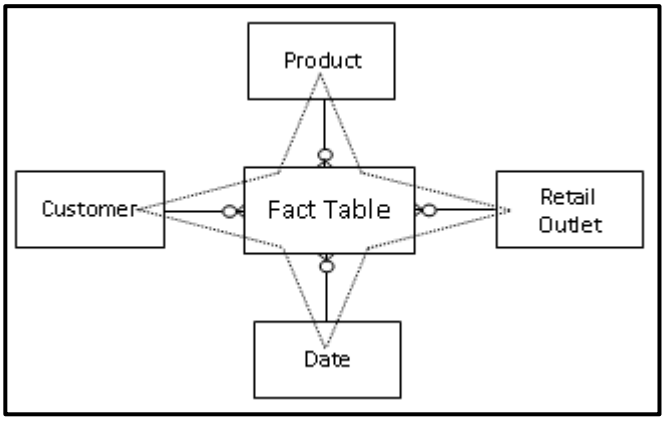

Figura 7: Ejemplo de un modelo estrella. Fuente: [24]. Elaboración propia.

A pesar que el sistema utiliza el modelo relacional, durante el diseño de un esquema estrella las restricciones de normalización se flexibilizan [37], ya que se tiene por objetivo crear un modelo dimensional, que no siempre es compatible con la tercera forma normal de las bases de datos. Por esta razón, es posible asociar más de una jerarquía a cada dimensión. Por ejemplo, la Figura 8 muestra los atributos de la dimensión *customer* del modelo ilustrado en la Figura 7. En esta dimensión es posible identificar 3 jerarquías independientes que no han sido normalizadas: la industria, el lugar, y la segmentación de mercado, cada una con sus respectivos niveles.

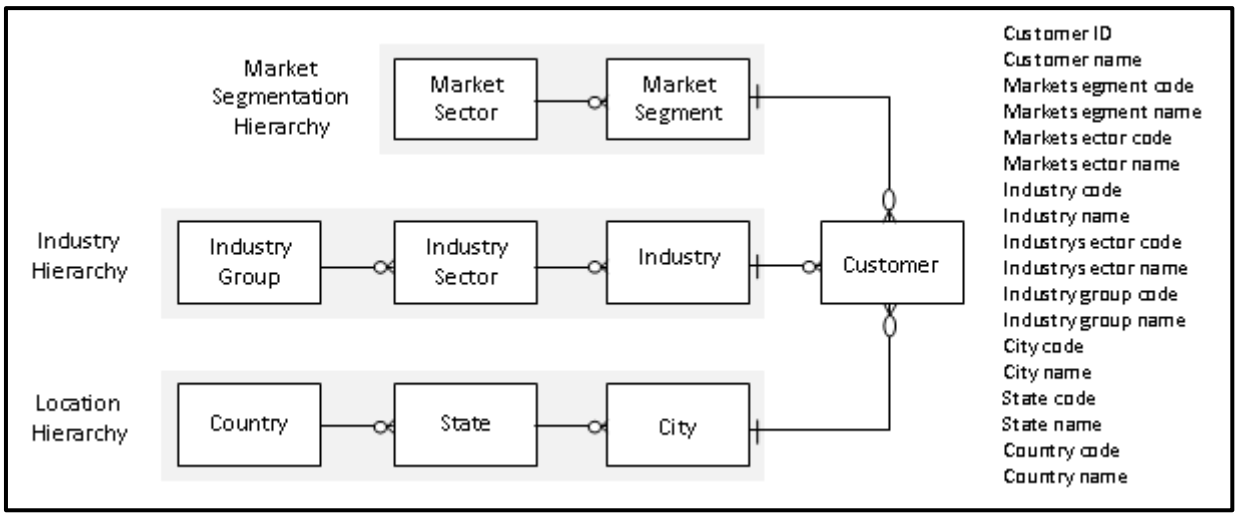

Figura 8: Múltiples jerarquías en una dimensión. Fuente: [24]. Elaboración propia.

Al implementarse en bases de datos relacionales, los reportes se pueden llevar a cabo a través de consultas SQL. Sin embargo, para obtener indicadores que consideren varias dimensiones, es necesario realizar un *star join*, el que reduce considerablemente el rendimiento a medida que aumenta la cantidad de dimensiones incluidas en la consulta [37].

Una variación al modelo estrella es el modelo constelación, cuya característica es que posee varias *Fact Table's,* lo que permite realizar consultas agregadas para distintas medidas que no necesariamente comparten las mismas dimensiones. Su nombre se debe a que sigue la misma lógica del modelo estrella, y es posible observar varios de ellos dentro de la constelación.

Otra alternativa al modelo estrella está basado en lo que algunos autores proponen, que sería normalizar las tablas que representan las dimensiones, lo que permite reducir la redundancia de datos. Esta solución es conocida como *snowflake* [21], y en ella existen tablas dimensionales que apuntan a otras de ellas.

El siguiente es un ejemplo de una consulta MDX sobre el modelo descrito en la Figura 7, que selecciona la cantidad de unidades vendidas en el mes de noviembre, de los productos computador y celular.

```
SELECT
{Measures.Unidades} on columns,
{Product.Computador,Product.Celular} on rows
FROM CuboVentas
WHERE (Date.Mes.Noviembre)
```

2.2.2.3. Componentes de un Data Warehouse

Antes de comenzar el desarrollo de un Data Warehouse, es importante entender cada uno de sus componentes, debido a que cada uno de ellos cumple una función específica. KIMBALL [18] indica que una de las principales amenazas para el éxito de un *Data Warehouse* es confundir los roles y las funciones de sus componentes.

Tal como se aprecia en la Figura 9, un *Data Warehouse* está compuesto por 4 elementos básicos: los sistemas operaciones, la data staging area, la data presentation area, y las herramientas de acceso a los datos [18].

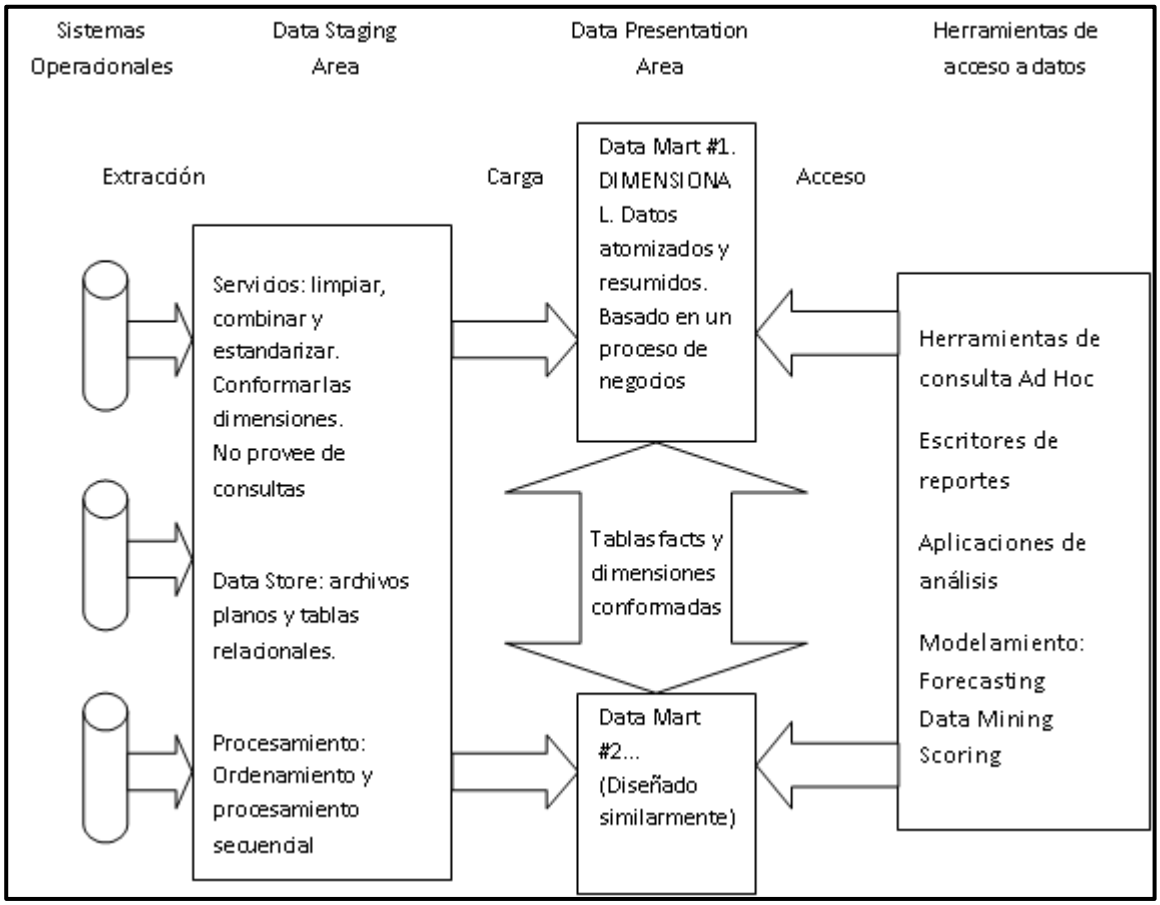

Figura 9: Elementos básicos de un Data Warehouse. Fuente: [18]. Elaboración Propia.

Sistemas operacionales

Corresponden a los sistemas que registran las transacciones del negocio. Sus principales prioridades son la disponibilidad y el rendimiento en el procesamiento. A diferencia de los Data Warehouse's, sus consultas no suelen considerar una gran cantidad de filas. En este sentido, la existencia de un repositorio de datos permite que los sistemas operacionales no sean los responsables de representar el pasado.

Usualmente el desarrollador del repositorio de datos no suele tener control sobre los sistemas operacionales, por lo que debe tomarlos como un elemento externo.

Data Staging Area

La *Data Staging Area* (DSA) corresponde al lugar donde los datos son consolidados y transformados en información [37]. En otras palabras, es tanto un lugar donde los datos son almacenados, como también es una serie de procesos conocidos como extracción, transformación y carga de los datos (ETL)**.** Así, su importancia radica en que es lo único que separa a los sistemas operacionales con el área de presentación de los datos.

Al ser un lugar de transición de los datos, es clave que la DSA esté fuera de los alcances del usuario de negocios, y que no esté diseñada para responder a servicios de consulta y presentación**.**

En cuanto a la normalización de los datos en la DSA, a pesar de que es una práctica aceptable, no es recomendable. Esto debido a que, al tener estructuras normalizadas de datos para la DSA, y estructuras dimensionales para el área de presentación, el proceso ETL se debe realizar en dos oportunidades. Esto conlleva a que en muchos proyectos de *Data Warehouse*, los desarrolladores destinen demasiado tiempo en la normalización de los datos, y no concentren sus esfuerzos en desarrollar un área de presentación que permita mejorar el proceso de toma de decisiones.

Data Presentation Area

Corresponde al elemento del *Data Warehouse* donde los datos son organizados, almacenados, y puestos a disposición para ser consultados por el usuario de negocios, o por aplicaciones de análisis.

A diferencia de la DSA, donde los datos no están al alcance de los usuarios de negocios, el área de presentación de los datos es todo lo que el usuario percibe, y que es el Data Warehouse**.** Es por esta razón que, centrar recursos en su desarrollo es parte importante del éxito de los proyectos de *Data Warehouse*, ya que sólo si los usuarios finales interactúan con la capa de presentación, el proyecto podrá ser declarado exitoso.

La capa de presentación suele ser una zona donde se integran distintos *Data Marts*, siendo estos últimos representaciones de los datos de un único proceso de negocios. Para esto, los distintos Data Marts deben ser construidos utilizando dimensiones y tablas *fact* comunes, lo que es la base de la arquitectura de *Data Warehouse's* propuesta por Kimball y Ross [18].

La idea detrás de construir un *Data Warehouse* basado en distintos *Data Marts*, proviene de la complejidad involucrada, en términos presupuestarios, políticos y de tiempo, en desarrollar un *Data Warehouse* centralizado.

Herramientas de acceso de datos

Finalmente, el cuarto elemento de un *Data Warehouse* corresponde a las herramientas de acceso de datos, las que permiten a los usuarios de negocios utilizar el área de presentación de los datos, tanto para el análisis de información, como para la toma de decisiones**.**

Las herramientas de acceso de datos pueden ser de distintos tipos, de acuerdo a los requerimientos del usuario que la utilizará. Las más simples permiten extraer indicadores a través de consultas multidimensionales. Por otro lado, otras más complejas permiten realizar minería de datos, tomando como fuente los datos almacenados en el *Data Warehose*.

En "The Data Warehouse Toolkit" [18] se menciona que alrededor de un 80% a 90% de los usuarios utilizan las herramientas de acceso de datos de acuerdo a parámetros pre-establecidos, habiendo sólo alrededor de un 10% de los usuarios del *Data Warehouse* capacitados para utilizar toda la potencialidad de las herramientas, a través del diseño de consultas multidimensionales y el posterior procesamiento de los resultados.

2.2.2.4. Proceso ETL

Al diseñarse un sistema de información que reúne datos desde distintas fuentes, es necesario utilizar técnicas que extraigan los datos, los transformen a formatos específicos y útiles, y los carguen en la base de datos. Este conjunto de técnicas es conocido como el proceso ETL [37].

Extracción

Consiste en extraer datos operacionales desde distintas fuentes de la empresa, como son sistemas informáticos, archivos de datos, o bases de datos relacionales, entre otros [37].

Durante su diseño es necesario tener en cuenta cómo se realizará en el tiempo la extracción de los datos, considerando escenarios cambiantes, como pueden ser nuevas fuentes o formatos.

Transformación

Esta etapa consiste en transformar los datos extraídos a valores y formatos específicos. Para esto, primero se debe realizar la estandarización de los datos, que consiste en llevarlos a la misma unidad de medida. Luego se deben generar los valores que serán almacenados en el repositorio, que representarán las distintas jerarquías dentro de cada dimensión. Estos dos complejos pasos conllevan a que esta etapa del proceso ETL sea la que tarda más tiempo en desarrollarse [37].

Para llevar a cabo esta etapa existen 3 alternativas:

- Programar algoritmos en los lenguajes de cada fuente de datos.
- Utilizar herramientas de pago.
- Utilizar una base de datos transitoria para posteriormente utilizar un único lenguaje para la transformación.

Debido a la complejidad de la primera alternativa, y el costo de la segunda, la base de datos transitoria resulta ser la más utilizada. Este repositorio es la ya mencionada DSA.

Carga

Suele ser la etapa más sencilla del proceso ETL. Consiste en cargar los datos de la DSA en el repositorio final, utilizando las herramientas y el lenguaje de la base de datos [37].

2.3. ANÁLISIS DE UN SITIO WEB

Es la extracción de información y conocimiento a partir de los datos emanados desde un sitio web. Existen dos enfoques para llevarlo a cabo:

- Análisis de indicadores de desempeño: consiste en el análisis multidimensional de los datos.
- *Web Mining*: consiste en aplicar técnicas del *Data Mining* con la finalidad de extraer patrones acerca del contenido, uso y estructura del sitio.

2.3.1. WEB WAREHOUSING

La arquitectura *Data Warehousing* suele ser utilizada sobre datos operacionales de la empresa, desde donde es posible extraer indicadores de desempeño. Sin embargo, existen otras aplicaciones menos tradicionales de esta arquitectura. Una de ellas corresponde al *web warehousing* [37], que utiliza datos provenientes desde un sitio web, almacenando tanto el contenido de las páginas, como lo generado en la navegación de los usuarios (weblogs).

Los datos provenientes desde los archivos weblogs representan toda la interacción de los usuarios con los servidores web del portal. A esta interacción se le conoce como *clickstream*, que corresponde a cada elemento web registrado en los servidores web [18].

Dada la naturaleza propia de los datos utilizados en este tipo de *Data Warehouse's,* existen dimensiones no tradicionales que permiten caracterizar el contexto en el que suceden las peticiones al servidor. Entre estas se encuentran *página, session,* o *referrer.*

Otra particularidad del *Web Warehousing* corresponde a que se suele utilizar un volumen muy grande de datos, tanto de carácter textual como numérico. Incluso, en KIMBALL [18] se afirma que es la cantidad de datos más grande que

se utilice en proyectos de *Data Warehousing*. Esto hace que sean considerados como desafiantes, pero también dificultosos por el procesamiento involucrado.

En cuanto a los datos que son utilizados, a pesar que a través de los archivos weblog se posee información acerca de las preferencias y comportamiento del usuario, suele ser útil tener mayor información del usuario, lo que es posible a través de la alimentación del *Webhouse* por medio de otros datos, los que pueden ser del tipo operacionales. Un ejemplo de esto corresponde a los datos que los usuarios entregan al registrarse en un sitio web, de donde se suele obtener información del tipo demográfica, como lo son el sexo, edad y ubicación geográfica.

Existen distintos modelos propuestos en la literatura para el diseño de un *Webhouse*, donde la diferencia suele estar en el grano escogido.

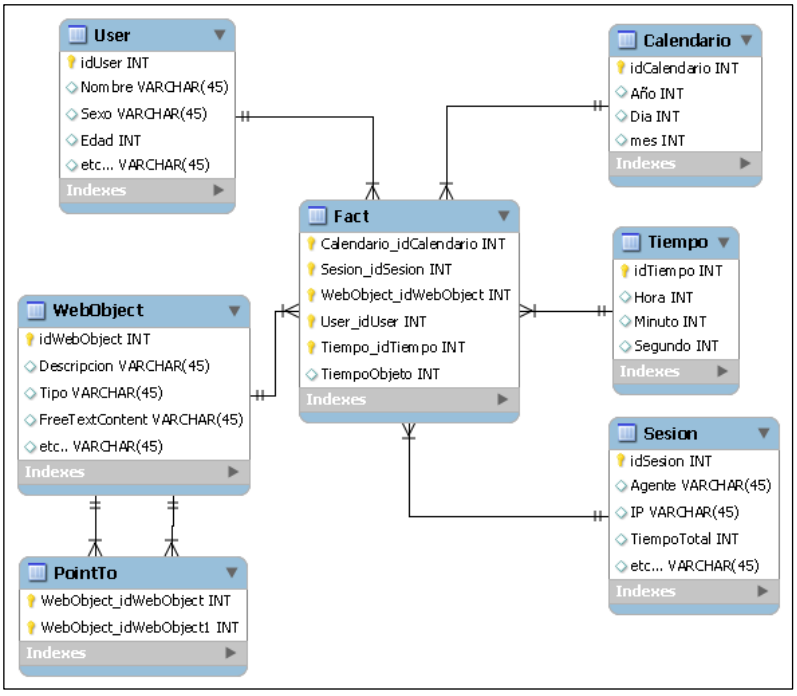

Figura 10: Modelo de datos para un *web warehouse*. Fuente: [37]. Elaboración propia.

En la Figura 10 es posible observar un ejemplo de un Webhouse. La medición hecha en la tabla *Fact* corresponde al tiempo destinado al objeto, que es posible cuantificar a través de la metodología de Velásquez y Dujovne [10]. Por otro lado, entre las dimensiones se encuentran:

- Session : identifica la sesión a la que está asociada la petición del objeto.
- Web Object : incluye las características del objeto requerido.
- User : información del usuario, que se obtiene desde fuentes distintas al web log.

- Tiempo : corresponde a la fecha y hora en que fue solicitado el objeto.

2.3.2. DIFICULTADES DEL WEB WAREHOUSING

Los datos contenidos en los archivos weblogs entregan información acerca de cada petición hecha por los usuarios al servidor web. Sin embargo, a pesar de tener esta valiosa data, existen problemas sujetos al procesamiento que se debe realizar para obtener la información que se desea. Estos problemas están ligados a la reconstrucción de las sesiones de los usuarios, la identificación del origen de los usuarios, y la existencia de distintos usuarios que visitan el portal desde una misma red local.

Identificar el origen de la visita

En el análisis de los *clickstreams*, un atributo importante es el referrer, que corresponde a la página web desde donde el usuario llega al portal, a través de un click en un enlace. Sin embargo, existen situaciones que impiden trazar cómo el usuario llegó al portal [18].

En algunos casos, una de las páginas del portal –usualmente el *home*- es la página de inicio del navegador utilizado por el usuario. Esto es de gran valor a la hora de generar visitas, sin embargo, no es posible identificar en qué casos el usuario visitó la página por esta razón.

Lo mismo ocurre cuando el usuario ha guardado el portal en sus *bookmarks,* teniendo acceso directo a alguna de las páginas del portal, lo que impide la identificación de la naturaleza de la visita.

Así, distintas situaciones provocan el mismo efecto en los archivos weblogs: no poseer referrer. El desarrollador del *Webhouse* no podrá distinguir si el usuario llegó debido a la página de inicio de su navegador, o por iniciativa propia, tanto al utilizar los *bookmarks,* como por escribir la página en la barra de direcciones.

Identificar la sesión

En el desarrollo de un *Webhouse*, es importante reconstruir las sesiones de los usuarios. El archivo *weblog*, a pesar de contener datos que caracterizan a los visitantes, como son la IP o el agente utilizado, no posee un atributo que indique a qué sesión corresponde la petición. Esto es así debido a que el protocolo que usualmente es utilizado en la Web es el HTTP, que no utiliza una acción de inicio de sesiones. Por esta razón, es necesario utilizar alguna técnica que permita identificar con seguridad las sesiones de los usuarios, y así tener conocimiento de cuál fue el set de páginas que el usuario visitó, y en qué orden [18].

Las técnicas que son utilizadas para identificar las sesiones no están libres de errores**.** Se considera que deben satisfacer dos criterios para que las sesiones satisfagan la condición de reconstruir el comportamiento de usuarios reales en el portal: que las páginas pueden ser agrupadas, y que ellas pertenezcan a la misma visita y al mismo grupo [37].

Existen dos clases de estrategias que los desarrolladores utilizan para reconstruir las sesiones de los usuarios [37]:

- Activas
  En ella se utilizan cookies en los computadores de los usuarios cuando visitan por primera vez el portal, guardando en ellas la información de las sesiones. Sin embargo, esta técnica presenta dos problemas importantes: que irrumpe sobre la privacidad de los usuarios, incluso no satisfaciendo las leyes de protección de datos de algunos países; y que las cookies pueden ser eliminadas por los usuarios de sus computadores.
- Reactivas
  Son las más utilizadas, y utilizan la información contenida en los archivos *weblogs* para reconstruir las sesiones de los usuarios. Los desarrolladores pueden optar por dos alternativas: definir una sesión en base a una secuencia lógica de páginas dentro del portal, tomando en cuenta las páginas que se pueden visitar desde otras desde los *hyperlinks*. O bien, definir una sesión en base al set de páginas visitadas por un usuario en un tiempo pre-definido, que suele tomarse 30 minutos.

Identificar al visitante

Una de las mayores dificultades para el *Webmaster* o el desarrollador del *Web Data Warehouse*, es identificar al usuario que utilizó el portal. Básicamente se presentan 5 problemas para llevar esta tarea a cabo [18]:

- Los usuarios desean ser anónimos.
  Usualmente no existen razones suficientes para que los usuarios entreguen sus datos personales a los administradores del portal, lo que dificulta las tareas de registro de usuarios.
- Al registrarse, los usuarios suelen utilizar datos falsos.
  Según datos mostrados por KIMBALL [18][17]**,** los usuarios entregan seudónimos falsos al momento de registrarse en más del 50% de los casos, siendo una práctica que utilizan más las usuarias mujeres.
- No es posible identificar dos usuarios en un mismo computador.
  Incluso utilizando cookies, no se puede detectar cuando dos usuarios diferentes visitaron el portal desde el mismo computador en el período determinado como máximo para una sesión.
- Varios usuarios pueden visitar el portal desde la misma red local.
  Cuando varios miembros de una red local visitan el portal, el archivo *weblog* mostrará visitas con la misma IP. Este problema puede ser minimizado utilizando en las estrategias reactivas una combinación IP-Agente al momento de individualizar un usuario. Sin embargo, cuando los miembros de la red utilizan el mismo agente, no es posible identificar distintos usuarios.
- No se puede asumir que el usuario proviene siempre desde el mismo computador.
  Esto sucede cuando un usuario debe utilizar el portal desde dos

computadores diferentes a la vez (o en el mismo intervalo de tiempo). Si se utilizan *cookies*, se tendrán en cuenta dos usuarios distintos. Y si se usan heurísticas basadas en los archivos *weblogs*, es posible que el usuario utilice en los computadores distintos navegadores, haciendo imposible su identificación.

2.3.3. PROCESO ETL

Como todo desarrollo de un *Data Warehouse*, para los repositorios de información web es necesario aplicar sobre los datos el proceso de extracción, transformación y carga. Sin embargo, dada la naturaleza de los datos utilizados, y las dimensiones no tradicionales que son consideradas, el proceso es llevado a cabo con características propias.

Extracción de datos

Esta etapa consiste en extraer los datos relevantes desde las distintas fuentes de de información. Éstas últimas generalmente son los archivos weblogs, el contenido HTML de las páginas del portal, y cualquier otra fuente que contenga información necesaria por los usuarios del repositorio [34]**.**

A menudo no se presta atención a los cambios en las fuentes de datos, tanto en su estructura como en su formato, y no existen mecanismos que notifiquen cuando se producen. Esto provoca que los algoritmos utilizados en la extracción fallen e interrumpan la etapa sin previo aviso [34].

En esta etapa, los archivos *weblog* suelen ser sometidos a un primer filtro, que elimina aquellos registros que no son útiles para el repositorio [22]. Ejemplo de esto último corresponde a la eliminación de las peticiones asociadas a *web crawlers*[7]*.*

Transformación de los datos

En esta etapa los datos son preparados para la carga en el repositorio que será utilizado por los usuarios, lo que conlleva a que sea la tarea que más tiempo consume durante el proceso ETL. Los programas desarrollados deben ser capaces de cruzar los datos extraídos desde las distintas fuentes, y llevarlos a los formatos que son requeridos para la correcta lectura de usuarios humanos [34]**.**

Algunos programas que son desarrollados son el de la sesionización de los registros; la construcción del *vector space model* para el contenido del portal, y la construcción del *user behaviour* para la navegación de los usuarios [22].

[7] Web crawlers: programa que inspecciona las páginas de la Web. Generalmente se utiliza para obtener copias de las páginas.

Carga de los datos

Esta tarea, a pesar de parecer sencilla, no lo es. Antes de proceder con la carga de los datos en la *Fact Table*, es necesario actualizar las dimensiones, o incluso agregar algunas nuevas si los requerimientos lo indican. Esto implica que los datos contenidos en las nuevas filas de la *Fact Table* también deben ser actualizados. Sólo después de haber hecho estas tareas se puede proceder a la carga de datos nuevos en la *Fact Table*, con una previa validación de las llaves de las dimensiones que serán utilizadas [34].

2.4. WEB MINING

El análisis de un sitio *web* también puede ser hecho con la finalidad de extraer conocimiento desde los datos emanados por la navegación de los usuarios y el contenido de las páginas, lo que es posible a través de la aplicación de técnicas del *Data Mining*, las que permiten extraer patrones acerca del contenido, estructura y uso de un sitio *web* [22]. Así, se da paso a la disciplina del *Web Mining*, que corresponde a la adaptación del *Data Mining* sobre los datos de la Web.

2.4.1. TÉCNICAS DEL WEB MINING

El *Web Mining* es la aplicación de técnicas del área de la *Minería de* Datos sobre datos originados en la Web, con el objetivo de encontrar patrones en ellos. En esta sección se presentan 3 de las técnicas más utilizadas durante la etapa de minería de datos [37][13]**:**

1. **Clasificación:**
   Corresponde al proceso de encontrar una serie de modelos o funciones que describan las distintas categorías o clases de los datos, con el objetivo de asignar posteriormente a cada objeto una clase que lo describa. Para esto, el modelo es derivado a partir de un conjunto de datos de prueba, cuya clasificación ya se conoce, para posteriormente aplicar el modelo a otros objetos de clase desconocida. Para la obtención del modelo, se pueden utilizar diversos algoritmos, como son las reglas de clasificación (If – Then), árboles de decisión, redes neuronales, o fórmulas matemáticas. A la clasificación se le denomina como *aprendizaje supervisado*, pues se inicia el proceso con datos ya clasificados, para posteriormente utilizar el aprendizaje obtenido en otros datos.
   A pesar que a través de la clasificación se pueden obtener las clases para los objetos, en algunos casos se requiere encontrar un valor desconocido para los datos, especialmente numéricos, lo que se realiza a través de la predicción, que aunque sirve para encontrar categorías de los objetos, es usualmente utilizada para la predicción de valores, lo que la diferencia de la clasificación.

Tanto la clasificación como la predicción suelen contar en algunos casos con un *análisis de relevancia*, con el objetivo de identificar aquellos atributos que no contribuyen en la obtención del modelo.

2. Clustering

A diferencia de la clasificación, en donde se utilizan datos cuyas clases ya son conocidas, en el *clustering* se analizan objetos sin consultar su categorización, usualmente debido a que el conjunto de datos de prueba no cuentan con ella. Así, el *clustering* es muchas veces utilizado para generar las distintas clases de objetos, y por esta razón se le denomina *aprendizaje no supervisado*.

El objetivo es agrupar los datos bajo la premisa de "*maximizar la similitud dentro de una mismo grupo, y minimizar la similitud entre grupos"*, comparando los distintos objetos a través de una medida de similitud.

Las técnicas de *clustering* más utilizadas son:

- Clustering particionado
  Esta técnica divide los *n* elementos del conjunto total de datos en un número de *p* particiones, definido a priori, cumpliendo que cada partición contenga al menos un elemento, y que la intersección entre las particiones sea vacía.

- Clustering jerárquico
  En esta técnica se construye una descomposición jerárquica de los datos a través de dos métodos distintos. Por un lado, el *método aglomerativo* define en un principio cada dato como un grupo, para luego agrupar distintos clusters de acuerdo a su similitud, hasta llegar a una condición de término pre-definida. Por su parte, el *método de división* comienza con todos los datos en un mismo grupo, para luego separarlos de acuerdo a su medida de similitud, hasta llegar a la condición de término.

- Clustering basado en la densidad
  Esta técnica se basa en la noción física de la densidad. La idea principal es encontrar los clusters a través de un límite de densidad pre-definido, que no es más que el número máximo de elementos que se permitirá en cada cluster. Para esto, en cada iteración se asocia un objeto a un cluster *C,* si su distancia al centroide de *C* es menor al radio máximo pre-definido para los clusters. La técnica detendrá su iteración para cuando la cardinalidad de cada cluster sea menor al límite de densidad que se definió.

3. Análisis de asociación

Corresponde a la búsqueda de *reglas de asociación,* las que son condiciones sobre los valores de los atributos, que ocurren frecuentemente

juntas en un set de datos. Así, el objetivo es encontrar una correlación significativa entre condiciones aplicadas a los datos.

Las reglas de asociación son de la forma $X \rightarrow Y$, donde *X* e *Y* son subconjuntos del set de datos, y su interpretación viene dada por *"los datos que cumplen las condiciones en X suelen cumplir las condiciones en Y"*.

Una regla de asociación posee un porcentaje de *soporte*, que indica los datos que cumplen tanto X como Y. A la vez, posee un porcentaje de *confianza*, que es la probabilidad de que un dato que cumple X, también cumpla Y. Por ejemplo, la regla de asociación:

$$contiene\ (T, \text{"cama"}) \rightarrow contiene\ (T, \text{"colchón"}) \quad [soporte = 1\%, confianza = 60\%)$$

Significa que en el 1% de las transacciones, el cliente compro tanto una cama como un colchón, y que existe una probabilidad de un 60% que un cliente que compra una cama, también compre un colchón.

2.4.2. APLICACIONES DEL WEB MINING

En esta sección se detallarán las distintas categorías del *Web mining*. De acuerdo con Kosala [19]**,** el *Web mining* se puede clasificar en 3 áreas, definidas a partir de la sección de la Web que se minará: Web content mining, Web usage mining, y Web structure mining. Un resumen de estas categorías y sus principales aplicaciones es mostrado en la Tabla 1.

2.4.3. WEB CONTENT MINING (WCM)

Corresponde a la rama del *Web Mining* que tiene por objetivo encontrar patrones acerca del interés de los usuarios sobre las palabras, oraciones y párrafos del sitio [36]. Así, el WCM provee información al webmaster que le permitirá tomar decisiones acerca de cambios en el contenido de las páginas de su sitio web.

El WCM consta de dos aplicaciones generalmente utilizadas: web page content mining, que mina el contenido de cada página del sitio; y el search result mining, que se utiliza para mejorar las búsquedas de contenidos a través de motores de búsqueda [29]. En Kosala [19]**,** esta diferenciación es basada desde dos puntos de vista: desde el Information retrieval (IR), y desde las bases de datos. El primero apunta a asistir a los usuarios en la búsqueda o filtración de información, y el segundo tiene por objetivo modelar la información de la Web, para así realizar consultas más complejas.

2.4.4. WEB USAGE MINING (WUM)

Corresponde a la aplicación del *Web Mining* que extrae patrones acerca del comportamiento de los usuarios a través del sitio web [27]. Para esto, la sesionización de los datos obtenidos de los *web logs* cobra relevancia, ya que

permite definir los vectores de características para cada una de las sesiones almacenadas en el servidor web, donde se suele considerar la secuencia de las páginas visitadas por el usuario, y el tiempo destinado a cada una de ellas [37].

Entre los beneficios del WUM para las organizaciones [9], está la determinación del valor de ciclo de vida de los clientes, cruzar las estrategias de marketing entre los productos, y la medición de la efectividad de las campañas de promoción, entre otras. Además, puede guiar una reestructuración del sitio, con el objetivo de crear una mejor presencia organizacional. Para sitios que contienen secciones publicitarias, los patrones de uso permiten enfocar los avisos en grupos de usuarios caracterizados.

2.4.5. WEB STRUCTURE MINING (WSM)

Corresponde a la técnica utilizada para obtener patrones acerca de la estructura de *hyperlinks* del sitio web. Esto permite obtener rankings de las páginas del sitio, que consideran tanto aquellas que la apuntan (*inbound links*), como también a las que apunta (*outbound links)* [22]*.* Para esto, el sitio web es representado a través de un grafo dirigido, donde los nodos corresponden a las páginas, y los arcos a los hyperlinks que existen entre ellas.

El modelo obtenido a través del WSM también permite comparar sitios [19]**.** Por un lado, es posible encontrar la similitud y las relaciones existentes entre distintos portales. Así como también es posible identificar sitios autoritativos para los usuarios (*authorities)*, que son definidos como los que contienen varios links apuntándolos; y sitios de información general para los usuarios (*hubs*), que son aquellos que apuntan a los sitios autoritativos.

	Web mining			
	Web content mining		Web structure mining	Web usage mining
	IR view	DB view		
Vista de los datos	○ Sin estructura ○ Semi estructurada	○ Semi estructurada ○ Estructurada (DB)	○ Estructura de links	○ Interacciones
Datos principales	○ Texto ○ Hipertextos	○ Hipertextos	○ Estructura de links	○ Weblogs
Representación	○ Bag of words, n-gramas ○ Términos, frases ○ Conceptos	○ Relacional ○ Grafo etiquetado	○ Grafo	○ Tabla relacional ○ Grafos
Método	○ TF-IDF y variaciones ○ Machine Learning ○ Estadística	○ Reglas de asociación ○ Algoritmos propietarios	○ Algoritmos propietarios	○ Machine learning ○ Estadística ○ Reglas de asociación
Categorías de aplicación	○ Clasificación ○ Clustering ○ Patrones en texto ○ Modelamien-to de usuarios	○ Identificar sub-estructuras frecuentes ○ Identificar el schema del sitio	○ Clasifica-ción ○ Clustering	○ Construcción del sitio, adaptación y gestión ○ Marketing ○ Modelamiento de usuarios

Tabla 1: Categorías del Web mining. Fuente: [19]. Elaboración propia.

2.5. TEXT MINING

Hoy en día es posible encontrar grandes cantidades de documentos, tanto en la Web como en sistemas corporativos, bibliotecas digitales, y otros, incrementándose el volumen de los textos día a día, los que a su vez son cada vez más accesibles a través de los buscadores web. Esto conlleva a la necesidad de encontrar técnicas de procesamiento de los documentos que permitan extraer información de manera automática a partir de ellos.

El *Text Mining* es un área de investigación que intenta resolver el problema de la sobrecarga de documentos, a través de técnicas de la minería de datos, *machine learning, natural language processing* (NLP)*, information retrieval* (IR)*,* y *knowledge managemenet* [11]. Para esto, el Text Mining se ocupa del pre-procesamiento de los documentos, el guardado de la representación intermedia de los textos, las técnicas para analizar aquellas representaciones, y la visualización de los resultados.

2.5.1. PREPROCESAMIENTO DE LOS DOCUMENTOS

En esta sección se detallará el proceso de tratamiento de los textos, abarcando la extracción, estandarización y tokenización de los documentos, el *stemming*, y la aplicación del *Vector space model*.

2.5.1.1. Extracción de documentos

El primer paso en el proceso de la minería de texto es la extracción de los documentos relevantes para el estudio desde las fuentes que correspondan. Dependiendo de la naturaleza de la fuente, o de la aplicación que se utilizará para extraer los documentos, este paso puede ser directo, o bien, parte del problema a resolver [38]**.** Por ejemplo, una aplicación que extrae documentos desde una intranet Web indica implícitamente cuáles serán las páginas que deben ser recolectadas, dejando como principal actividad la limpieza del código obtenido. Así, para este caso, la extracción de los documentos resulta directa.

Por otro lado, en algunos escenarios es necesario diseñar un proceso de extracción de documentos. Por ejemplo, si se descargará texto desde distintos sitios web, se debe diseñar un *web crawler* capaz de extraer los documentos relevantes desde los distintos sitios. O en otras ocasiones, el conjunto de documentos es de un tamaño considerablemente grande, lo que dependiendo de los recursos técnicos que se poseen, puede hacer necesario utilizar técnicas que permitan seleccionar sólo los documentos que resulten más útiles para la minería de texto a desarrollar.

Una tarea importante durante la extracción de los documentos resulta la definición de la fuente desde donde serán extraídos los textos. Hoy en día la Web posee una importante colección de documentos de los más diversos temas, transformándose en una potencial fuente de información para diversos problemas de minería de texto. Sin embargo, esto conlleva a que la data es de dudosa calidad, y requiere de una importante limpieza [38]**.** Por esta razón, el diseño del *web crawler* debe considerar tanto la selección de sitios relacionados con el tema, para lo que es usual utilizar comunidades web específicas, como también debe considerar la selección de sitios que contengan documentos con datos de calidad.

2.5.1.2. Estandarización de los documentos

Una vez que los documentos han sido recolectados, deben ser llevados a un formato único para todos, ya que cada documento tendrá un formato que dependerá de la forma en que fue generado, lo que impide un posterior procesamiento fluido de los documentos [38][37]**.** Por ejemplo, en algunos casos los documentos corresponden a una página web, por lo que el texto útil se encuentra entre tags propias del código HTML. En otros casos, el documento ha sido obtenido como imagen, lo que hace necesario llevarlo a un formato de texto a través de alguna técnica de procesamiento de imágenes. O también, el problema

puede provenir del formato de texto utilizado, los que pueden ser ASCII o Unicode, entre otros.

La importancia de la estandarización de los documentos radica en que las herramientas de minería de datos que se utilizarán no dependerán del formato de origen de los documentos, ya que la extracción de conocimiento a partir de textos no debe depender de cómo éstos fueron generados [38]**.**

2.5.1.3. Tokenización

Una vez que los documentos han sido extraídos y transformados a un formato único, se lleva a cabo la etapa de tokenización. Ésta consiste en transformar el texto en distintas unidades, las que usualmente corresponden a palabras de la lengua en que el documento fue escrito [11]**.** Conceptualmente, cada *token* corresponde a una instancia de un *tipo,* por lo que un documento debería tener un número mayor de *tokens* que *tipos* [38]. El siguiente es un ejemplo de la tokenización, que permite entender de manera más clara la diferencia entre un *token* y un *tipo*:

- "*La casa de Tamara es de color azul, y está ubicada en la R.M. La de Miguel es blanca".*

El resultado de la tokenización es:

- la casa de tamara es de color azul y esta ubicada en la r.m. la de miguel es blanca.

La oración posee 19 *tokens,* los que corresponden a 18 palabras más el *token "r.m."*. Sin embargo, el número de *tipos* es tan solo 16, ya que tanto "*la", "de"* y *"es"* se repiten en una ocasión.

Dentro de los principales problema ligados a este proceso cuando es realizado de manera automática por un computador, está en la identificación de los elementos que representan a un delimitador de *tokens,* ya que existen algunos caracteres que en algunos casos son parte de un *token*, y en otros funcionan como delimitadores [38]. Por ejemplo, en el ejemplo mencionado más arriba, "*r.m."* posee dos puntos que son parte de la abreviación de *"Región metropolitana"*. Sin embargo, es común que un punto sea un delimitador de oraciones o párrafos.

2.5.1.4. Stemming

El pre-procesamiento de los documentos durante el desarrollo de un estudio de minería de textos tiene por objetivo transformar los documentos en vectores numéricos que los representen, y así someterlos posteriormente a las técnicas de minería de datos. Sin embargo, para manipular grandes cantidades de

documentos, se hace necesario llevar a cabo mecanismos que sean capaces de reducir las dimensiones de los vectores generados, preservando eficientemente la representación de los documentos [15]**.** Una solución a esto corresponde al *stemming*.

El *stemming* es un proceso que se utiliza en sistemas de información desde la década de los 60'. Su objetivo en un comienzo fue el de aumentar el rendimiento de los sistemas a través de la disminución de las palabras únicas que se deben almacenar. Sin embargo, con el aumento de las capacidades de los computadores, el *stemming* ha tenido nuevos usos. Por ejemplo, fue utilizado en los sistemas para aumentar las respuestas en la búsqueda de documentos, sacrificando con ellos la eficiencia de las búsquedas exactas [20]**.**

En el caso del *Text mining,* el *stemming* cumple la función de llevar los *tokens* a una forma estándar, asociando a un mismo *tipo* distintas palabras que están relacionadas por su semántica. Sin embargo, su utilidad no siempre está garantizada, y dependerá de la aplicación. Su efecto más notorio es que reduce la cantidad de *tipos* en el corpus[8], y que aumenta la frecuencia de los *tipos* en cada documento [38][37]**.**

2.5.1.5. Vector Space Model

El *Vector Space Model* (VSM) [32] es el modelo del área del *Information Retrieval* más utilizado para representar documentos. Consiste en modelar cualquier texto, ya sea una consulta o un documento, a través de un vector de *n* dimensiones, donde *n* corresponde al número de palabras distintas existentes en la totalidad de los textos. De esta forma, cada una de las palabras de cada texto puede ser relacionada a una de las dimensiones del vector.

Una vez realizada la etapa de *stemming,* cada uno de los textos ha sido transformado a un set de *stem's*. Naturalmente, cada uno de ellos tiene distinta relevancia dentro del documento, por lo que el *Vector Space Model* asigna a cada dimensión del vector un peso específico. Para medir la importancia, un primer supuesto es que una palabra que se repite en varias ocasiones tiene una importancia mayor dentro del documento. Así, el VSM asigna en una primera instancia un mayor peso a aquellas palabras con una mayor frecuencia - *term frecuency* o *tf* - en el documento.

Un segundo supuesto que considera el VSM es que una palabra que se repite en muchos documentos del *corpus* resulta más irrelevante que una que está en menos documentos**.** Por ejemplo, si la palabra *desierto* se encuentra en sólo un documento del *corpus*, esa palabra entonces es importante para ese texto, independiente de su frecuencia en él. Así, una segunda medida a considerar corresponde a la frecuencia inversa de los documentos (o *idf*), la que es proporcional a $\frac{1}{df}$, donde *df* es el número de documentos donde la palabra está.

[8] Corpus: set de documentos que se considera en el estudio.

Finalmente, un tercer factor a considerar es el tamaño del texto. Una palabra en documento grande tiene mayores posibilidades de tener una frecuencia alta que un documento pequeño. Así, es necesario normalizar la frecuencia de la palabra de acuerdo al tamaño del documento, lo que usualmente es realizado a través de la norma euclidiana del vector.

Así, el VSM asigna a cada término del texto un peso determinado por la siguiente ecuación:

$$weight = \frac{tf * idf\ weight}{Norma\ euclidiana\ del\ vector} \qquad (1)$$

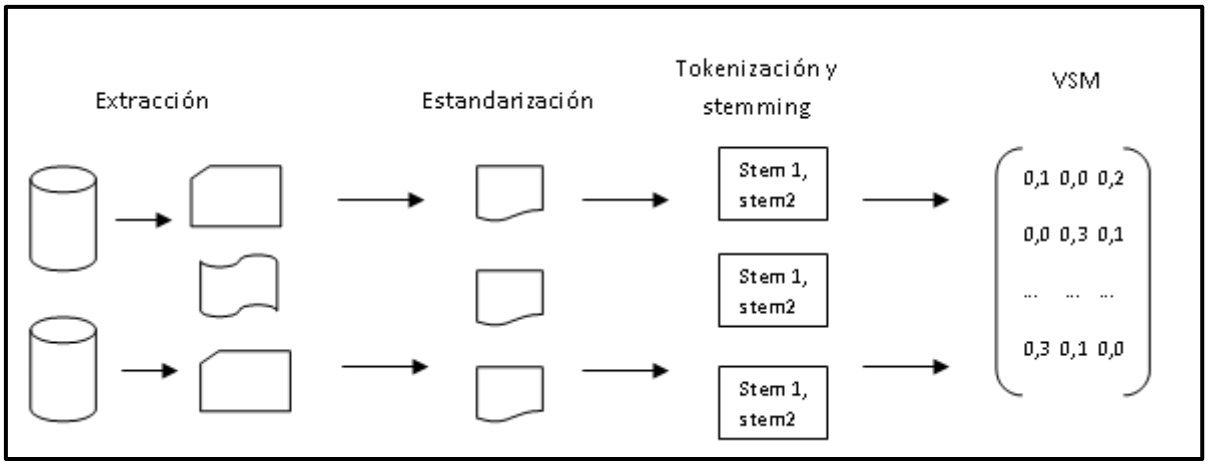

Figura 11: Pre-procesamiento de los documentos. Fuente: elaboración propia.

CAPÍTULO 3 – DISEÑO DEL WEBHOUSE

En este capítulo, se propone el diseño conceptual del *Webhouse* que se utilizará para obtener información a través de los datos del portal. Para esto, se sigue la metodología propuesta en 1.4, basada en los distintos modelos que se mostraron en el capítulo anterior.

3.1. REQUERIMIENTOS

La primera etapa del diseño del *Data Mart* consiste en definir los requerimientos que tendrá el desarrollo del trabajo.

A continuación se presenta la metodología que se utilizó para reconocer las necesidades de información, las fuentes de datos que se disponían, y el uso que se daría al repositorio.

3.1.1. REQUERIMIENTOS DE INFORMACIÓN

El primer paso consistió en conocer cuáles eran las necesidades de información que se poseían en el equipo del portal, con el objetivo de profundizar el análisis de las preferencias de los usuarios que hoy se realiza. Para esto, se llevo a cabo una serie de entrevistas con las personas relacionadas con la dirección del portal y el análisis de las sesiones de los usuarios, además de aquellos integrantes que generan el contenido del portal.

Las entrevistas tenían el objetivo de conocer qué necesidades de análisis resultaban prioritarias para el equipo, teniendo en cuenta las limitantes técnicas del estudio. Así, se definió que las entrevistas tuvieran las siguientes etapas:

- Conocer el funcionamiento del portal
- Conocer la información a la que hoy se tiene acceso.
- Reconocer al usuario del estudio.
- Identificar indicadores que hoy no son conocidos a través de las herramientas de pago utilizadas.
- Definir la factibilidad técnica de la medición de indicadores nuevos.

Funcionamiento del portal

El sitio con el que se trabajó es de carácter educativo, y cuenta con una cantidad de páginas superior a 30.000. La mayor parte de ellas corresponde a artículos con material educativo, ya sea en formato texto, visual o multimedia. En menor medida, el portal también cuenta con artículos de actualidad educativa.

Se tienen cuatro tipos de visitantes: estudiantes, docentes, directivos, y apoderados, donde cada uno de ellos tiene su espacio dentro del portal a través de los denominados *escritorios.* Estos consisten en una página compuesta de

secciones, donde se publican vínculos a artículos relacionados con los intereses del usuario objetivo del escritorio. Estas páginas tienen la particularidad de poseer, en su mayoría, vínculos a artículos de actualidad o de utilidad para el día a día, dando poco espacio a material con contenido educativo.

El material expuesto en cada uno de los escritorios es dinámico, siendo usual que a diario exista nuevo contenido publicado, y por consiguiente, otro que haya sido relegado de su posición. Esta tarea es llevada a cabo por un equipo de edición, el que sigue un calendario de publicaciones para los escritorios.

Otra página que tiene características propias es el *Home*, que posee, al igual que los escritorios, distintas secciones con actualidad educativa y recursos. Sin embargo, por ser la página principal del portal, no persigue cautivar a un tipo de usuario en específico.

Además, el portal cuenta con páginas relacionadas con herramientas disponibles para los visitantes. Por ejemplo, existe *Planificacción,* en la que los docentes pueden encontrar o diseñar planificaciones para sus clases en el aula. O *PSU*, donde los estudiantes pueden encontrar material de estudio para la Prueba de Selección Universitaria, existiendo la posibilidad de que sean orientados por el docente a través de la herramienta del portal.

Información a la que se tiene acceso

El equipo de análisis del portal trabaja con dos herramientas de pago: Certifica y Google Analytics, que entregan indicadores estandarizados para portales web.

A través de estas herramientas, el equipo tiene acceso a información acerca de las sesiones de los usuarios en el tiempo, como también indicadores de las páginas más visitadas, y de los links más preferidos dentro de las páginas.

Algunos ejemplos de indicadores a los que hoy se tiene acceso son:

- ¿Cuánto tiempo duran en promedio las sesiones de los usuarios?
- ¿Cuántas páginas en promedio son visitadas por los usuarios?
- ¿Cuál es la página más frecuente presente en las sesiones de los usuarios?
- ¿Cuál es el porcentaje de sesiones que visitan sólo una página?
- ¿Cuántas sesiones provienen del buscador “Google”?
- ¿Cuál es el día de la semana que posee un mayor número de sesiones?
- ¿Cómo se distribuyen las sesiones durante el día?
- ¿Cuál es el ranking de las páginas visitadas del portal?

A pesar de que los indicadores de las sesiones y las páginas más visitadas resultan útiles para el equipo, no ocurre lo mismo para aquellos que indican la cantidad de clicks en los links de cada página. Esto ocurre porque en los escritorios y en las páginas de las herramientas, el contenido es dinámico, y tanto *Certifica* como *Google Analytics* no son capaces de medir indicadores que entreguen las preferencias de los usuarios en el tiempo para cada sección en la página, independiente del artículo que sea publicado. Adicionalmente, las

herramientas de pago consideran en sus indicadores las *url's* de los artículos, y debido al importante número de enlaces en cada página, resulta dificultoso verificar a cuál corresponde, y en qué página se encuentra.

Usuario del estudio

Con las entrevistas se pudo identificar que existía un potencial usuario del estudio, que corresponde al analista de preferencias de los usuarios. Así, se decidió trabajar con él en la definición de indicadores a medir. Sin embargo, con el transcurso del tiempo, se verificó que existen otros usuarios, relacionados con la dirección del portal, y con la publicación de contenido en los escritorios y el home, que también son parte fundamental a la hora de tomar decisiones a partir del análisis de los datos.

Así, se definió que el usuario del prototipo a desarrollar, sería tanto el analista de datos, como el encargado de la publicación de contenidos, y que por otro lado, las personas ligadas a la dirección estratégica del portal también debían ser parte de la definición de los alcances del estudio.

Indicadores a medir a través del estudio

Teniendo en cuenta los indicadores que ya son conocidos por el equipo del portal, y el objetivo de obtener nueva información a través del procesamiento de los archivos *weblog*, se procedió a definir aquellas necesidades de información que resultaran prioritarias para el equipo, y que estuvieran dentro de las limitantes técnicas del estudio.

En este sentido, primero se procedió a restringir el análisis a 14 páginas del portal, las que corresponden a los 4 escritorios, y 10 páginas relacionas con herramientas para los usuarios –a partir de ahora, *páginas relevantes para el estudio*-.

Posteriormente se observó que un indicador importante para el equipo, y que no ha sido medido hasta hoy, es la cantidad de clicks que se realizan en las secciones de cada página. Este indicador permitiría al equipo evaluar las preferencias de los tipos de usuarios, identificando aquellas secciones que despiertan mayor interés en las páginas, y cuáles no. Esto es de especial interés, ya que permitirá conocer si los recursos humanos que se utilizan en la etapa de edición están siendo bien utilizados, lo que puede conllevar a nuevas políticas de publicación de contenidos, y un rediseño en la forma en que el usuario visualiza el contenido.

Por otro lado, existe otra variable dentro de las 14 páginas a estudiar que podría producir un sesgo en las preferencias de los usuarios, y que no ha sido medida. Esta corresponde a las denominadas *pestañas* o *viñetas*, que permiten dividir secciones en varias temáticas. El problema radica en que en la sección donde se utilizan las pestañas, sólo puede haber una que se encuentre por defecto activada, y por lo tanto, sólo será el contenido publicado en ella el que será visualizado por el usuario. Aquellos enlaces que estén dentro de las pestañas que no están activadas, sólo podrán ser vistos a través de un click en la correspondiente

pestaña. Así, a través de las herramientas de pago no es realizada aún una medición que indique si el contenido publicado en las pestañas está siendo visitado o no por los usuarios.

Por lo tanto, el estudio tiene por objetivo medir indicadores tradicionales para los portales web, como son los relacionados con las sesiones y las páginas, además de medir indicadores que aporten nueva información al equipo de análisis del portal. Estos nuevos indicadores son:

- Clicks hechos en cada sección de la página.
- Clicks hechos en cada pestaña de la sección.

3.1.2. REQUERIMIENTOS FUNCIONALES

La implementación del sistema a diseñar requiere de la definición de la arquitectura a utilizar, la que, en el contexto de sistemas de información, corresponde a un conjunto de componentes y sus relaciones, los que constituyen el marco para el diseño y desarrollo del sistema, donde los componentes corresponden a hardware, software, comunicación y seguridad.

En 2.2.2.3, se indicó que un Data Warehouse posee 4 elementos principales: los sistemas operacionales, desde donde se extraen los datos; la Data Staging Area, donde se lleva a cabo el procesamiento de los datos; la Data Presentation Area, donde los datos son almacenados en un repositorio multidimensional; y las herramientas de acceso a datos, donde el usuario puede visualizar y trabajar con la información obtenida. Así, teniendo en cuenta estos 4 componentes, se decidió trabajar con una arquitectura de 3 capas basada en la web, la que permite separar los datos, la lógica y la presentación.

3.1.2.1. Modelo de 3 capas

La arquitectura de 3 capas es una evolución del modelo de dos capas cliente/servidor, y es utilizada en grandes aplicaciones de negocios [1]. La principal característica es que en ella se consideran 3 componentes:

- Datos.
- Aplicación.
- Presentación.

La diferencia con su modelo antecesor es que la funcionalidad es separada en una capa intermedia, llamada *capa de aplicación.* Así, los usuarios se conectan a múltiples servidores de aplicación, los que a su vez pueden trabajar sobre variadas bases de datos [1].

La Figura 12 ilustra el funcionamiento del modelo de 3 capas. En primer lugar, el cliente realiza peticiones al servidor de aplicación, que para responder a dicha petición, usualmente debe realizar consultas al servidor de datos. Así, a través de la comunicación de las tres componentes es posible responder a las solicitudes del usuario.

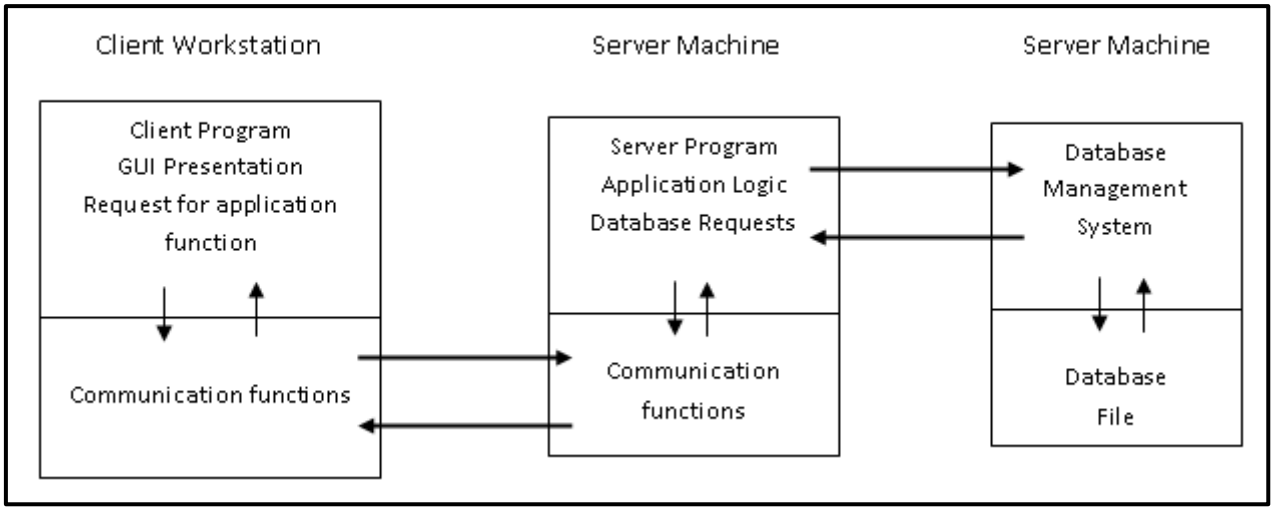

Figura 12: Modelo de 3 capas. Fuente: [1]. Elaboración propia.

Como se aprecia en la Figura 12, el usuario sólo puede interactuar con la capa de presentación, que corresponde al programa que visualiza el usuario. Esto permite abstraer al usuario, tanto de los datos, como de aplicación, a quienes realizan peticiones a través del programa que utiliza. Esto agrega una importante componente de seguridad, ya que los datos estarán protegidos en la medida que el programa utilizado por el usuario no permita una interacción directa con ellos, y su potencial modificación.

El modelo de 3 capas conlleva ventajas tanto en el desarrollo del sistema como en su rendimiento, las que son presentadas en la Tabla 2.

	Ventajas	Desventajas
Desarrollo	• Fácil implementación de complejas aplicaciones en la capa media. • Lógica de negocios abstraída de los datos y el cliente. • Cambios en la lógica de negocios sólo requieren de nuevo software a instalar en la capa media. • La lógica de aplicación es portable a otras plataformas de servidores de datos a través del software.	• Estructura más compleja. • Más difícil de mantener.
Rendimiento	• Mayor rendimiento en entornos medios y grandes.	• Implementación física en distintos lugares de la capa de datos y de aplicación afecta el rendimiento.

Tabla 2: Ventajas y desventajas del modelo de 3 capas. Fuente: [1]. Elaboración propia.

3.1.2.2. Modelo de 3 capas para un Webhouse

La aplicación del modelo de 3 capas en el Webhouse a diseñar, debe asignar cada una de las 4 componentes de éste último a alguna de las capas.

Como se ilustra en la Figura 13, en la capa de datos están tanto la DSA como la *Data Presentation Area*/modelo estrella. En la capa de aplicación están todos los programas que permiten extraer, transformar y cargar los datos. Y en la capa de presentación están las herramientas de acceso a los datos, desde donde el usuario obtiene los indicadores.

Figura 13: Componentes del Webhouse en el modelo de 3 capas. Elaboración propia.

3.1.3. USUARIO DEL SISTEMA

En el capítulo 3.1.1 se mencionó la existencia de dos usuarios del repositorio, pero dadas las características de los indicadores a medir, y la forma en que se produce el proceso de toma de decisiones en la organización, donde los distintos miembros aportan su mirada de la información, se identificó tan sólo un perfil de usuario para el repositorio. Este deberá tener acceso total a la información del repositorio, la que le será presentada a través de dos cubos OLAP (cubo “sesiones” y cubo “secciones y viñetas), cuyas dimensiones son las especificadas en el modelo de datos de la sección 3.3. Esto les permitirá a ambos analizar el portal a través de los indicadores definidos anteriormente, y aportar en el proceso de toma de decisiones a partir de la misma información.

3.2. DESCRIPCIÓN GENERAL

La obtención de los objetivos planteados necesita de una serie de fuentes de datos, repositorios de información, y programas de procesamiento, los que interactúan entre sí como se aprecia en la Figura 14**.** Así, el estudio consta de los siguientes elementos:

Portal web
El sitio web a estudiar corresponde a www.educarchile.cl. Tanto las opiniones vertidas por los usuarios, extraídas desde el código *html* de sus páginas, como los archivos *weblog,* donde se encuentra la interacción de los usuarios con el portal, son las fuentes de datos de las que se alimenta el estudio, y que permitirán la búsqueda de nuevo conocimiento para la organización que lo administra.

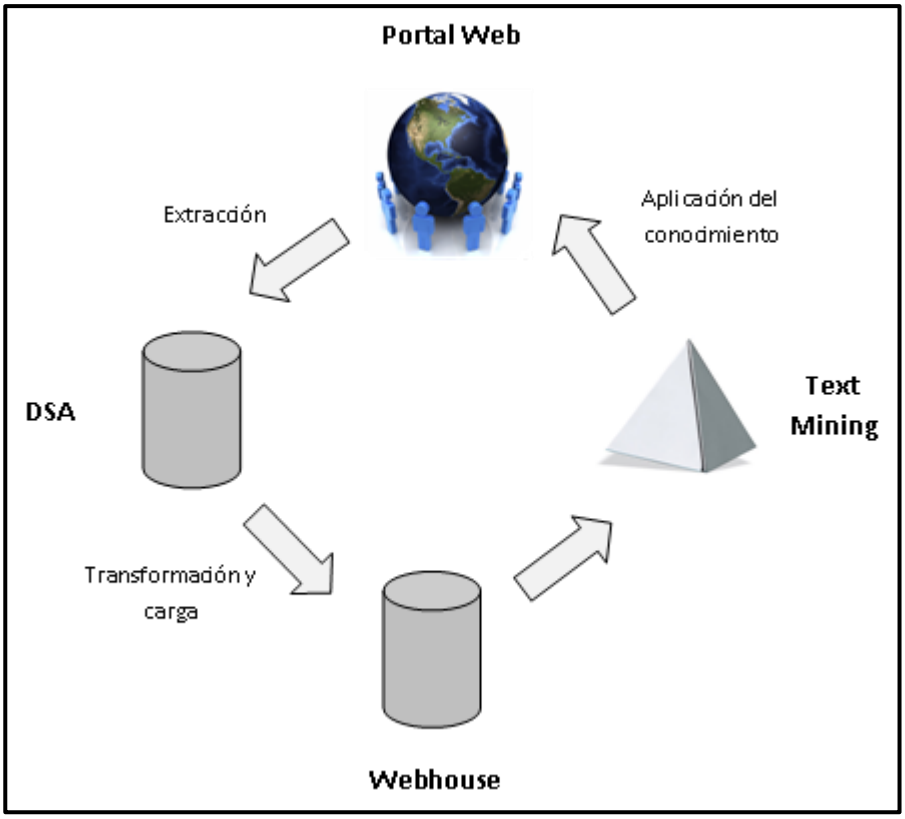

Figura 14: Descripción general del estudio. Elaboración propia.

Data Staging Area

En esta estructura de datos son almacenadas las peticiones al servidor web del portal que resultan útiles para el estudio, lo que hace necesario un pre-procesamiento de los datos en la etapa de *extracción*. Además, se almacenan las opiniones vertidas por los usuarios en la sección de debate del portal. La DSA permitirá el procesamiento de los datos en la etapa de *transformación*, lo que se traduce en la *sesionización* de las peticiones de los visitantes, como también en la *tokenización* de las opiniones. La utilización de esta estructura de datos da la posibilidad de utilizar distintos lenguajes de programación para llevar a cabo el pre-procesamiento de los datos en la etapa de extracción.

Data Presentation Area – Webhouse

Corresponde a la estructura donde los datos, una vez realizadas las etapas de transformación y carga, son almacenados de acuerdo al modelo multidimensional, lo que permite la realización de consultas OLAP para la obtención de indicadores de uso del portal. Además, funciona como fuente de datos para la realización de algoritmos de minería de datos, para lo cual el contenido de las opiniones es almacenado siguiendo el *Vector space model*.

Text mining

Acá se llevan a cabo los algoritmos de minería de datos sobre las opiniones almacenadas en el Webhouse. El *text mining* tendrá por objetivo evaluar un

clasificador de opiniones. Esto quiere decir, se buscará el modelo que prediga de manera más efectiva la clase a la que cada opinión pertenece.

Programas de extracción
Interactúa tanto con las fuentes de datos del portal, como con la DSA. Los distintos programas utilizados deben ser capaces de pre-procesar los archivos *weblog*, llevando aquellas entradas útiles para el estudio a la DSA. Además, los programas deben pre-procesar el contenido *html* de las páginas, almacenando sólo las opiniones en la estructura de datos mencionada.

Programas de transformación y carga
Aquí los datos provenientes desde la DSA son procesados con el objetivo de transformarlos al formato que se define en el repositorio Webhouse, lugar hasta donde son llevados a través del programa de carga. La transformación consta básicamente de dos etapas:

- Sesionización: consiste en reconstruir las visitas de los usuarios al portal, asignando cada petición a una única sesión. Para esto, se emplea un algoritmo que identifica las sesiones a través de la combinación *ip-agente* de los registros.
- Tokenización: el programa toma como input las opiniones almacenadas en la DSA, y lleva a cabo sobre el texto las tareas de borrado de stopwords y *stemming*.

Una vez llevada a cabo la etapa de transformación de los datos, se procede a realizar la carga hasta el repositorio de información, que se basa en la tecnología *Data Warehousing*. Por esta razón, el programa de carga debe preocuparse de que cada registro en la tabla *Fact* respete las llaves foráneas que caracterizan a las distintas dimensiones del modelo.

3.3. REPOSITORIO DE DATOS

El portal a estudiar cuenta con dos fuentes tradicionales de datos que permiten modelar el Webhouse. Estas fuentes corresponden a:

- Los archivos web logs.
- El código de las páginas:
  - El contenido de cada página.
  - La estructura de links del portal.
  - Las opiniones vertidas por los usuarios.

Adicionalmente, y con la finalidad de reconocer los cambios en el contenido de las páginas a estudiar, se ha considerado una fuente extra de información, que corresponde a una planilla de datos generada por los encargados de publicar contenido en el portal. En esta se registran todos los cambios que se producen en las páginas relevantes del estudio.

Como se indicó en la sección 2.3, el repositorio de información web tiene como finalidad ser una fuente de datos, tanto para el análisis multidimensional a través de consultas OLAP, como para aplicaciones técnicas del Data Mining. Para el desarrollo de este estudio, se decidió utilizar un modelo estrella del tipo constelación, debido a que se utilizarían dos tablas fact's para la obtención de los indicadores deseados.

Así, teniendo en cuenta los datos a utilizar, los indicadores que se desea obtener, y el objetivo de almacenar las opiniones que los usuarios realizan en la sección de debate del portal, se ha diseñado el siguiente modelo para el *Webhouse*:

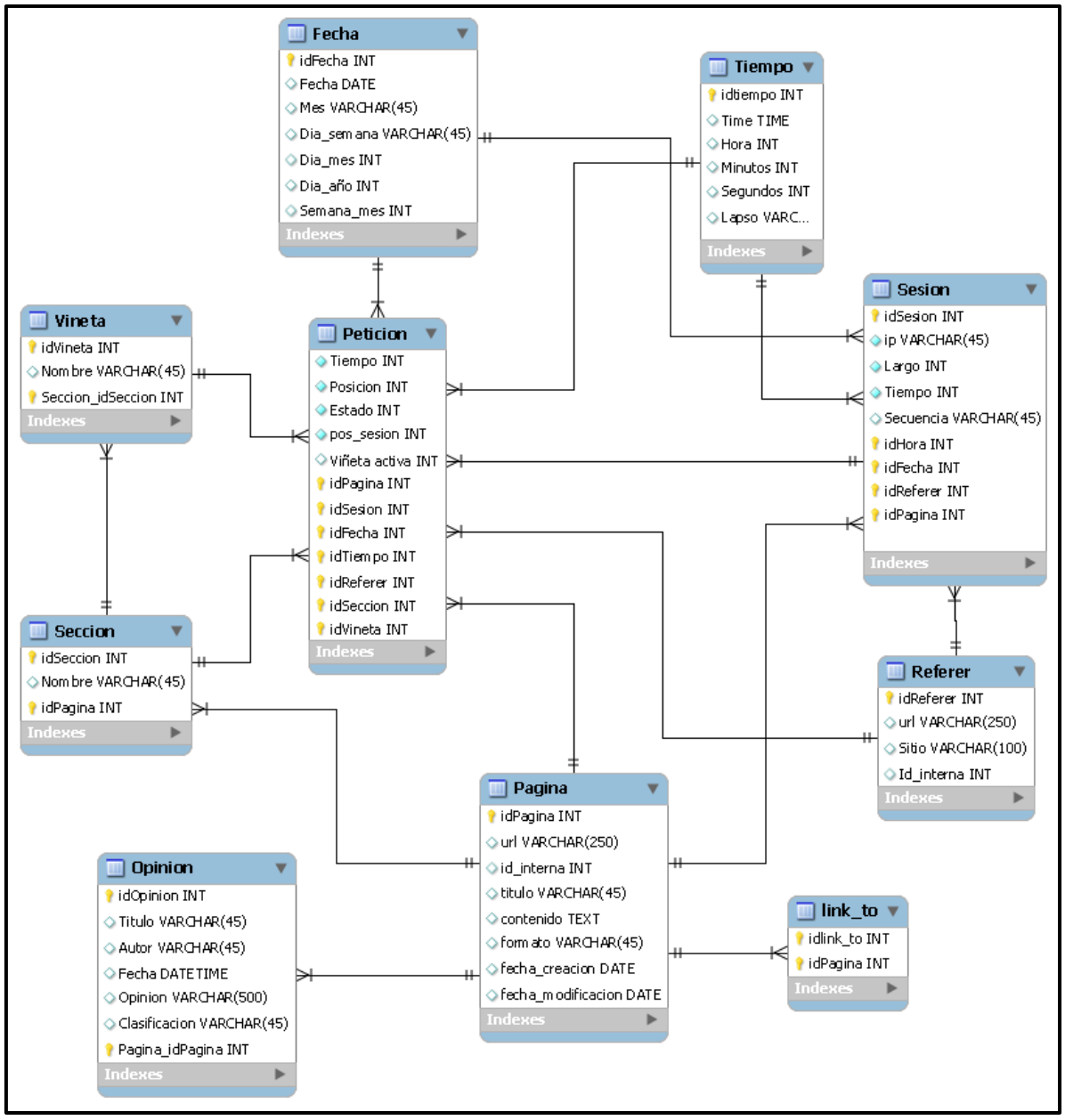

Figura 15: Modelo lógico del Webhouse, consistente en dos tablas facts.

- **Dimensión "Tiempo"**
  En ella se registra la hora del día en la cual la petición a la página fue realizada.
  - Hora: representa a la hora del día en que fue hecha la petición. Va desde 0 a 23.
  - Minutos: representa el minuto de la petición. Va desde 0 a 59.
  - Segundos: representa el segundo de la petición. Va desde 0 a 59.
  - Time: Corresponde al tiempo del registro en formato hh:mm:ss.

- Lapso: momento del día. Ejemplos: almuerzo, madrugada.

- **Dimensión "Fecha"**
  Tiene por objetivo registrar la fecha en la que se realizó la petición. Los niveles que considera son:
  - Fecha: En formato DATE.
  - Dia_semana: representa en cadena de texto el día de la semana. Va de 'lunes' a 'domingo'.
  - Dia_mes: En formato INT, toma valores desde 1 a 31.
  - Dia_año: En formato INT, va desde 1 hasta 366.
  - Semana: corresponde a la semana del año. Va desde 1 a 52.
  - Mes: corresponde al mes del año en formato INT. Va desde 1 a 12.
  - Año: representa al año de la petición. Para este estudio sólo considera el año '2011'.

- **Dimensión "Referer"**
  En ella se registra la página desde donde se llegó a la página del portal.
  - Url: indica la dirección de la página en una cadena de texto.
  - Sitio: Representa al sitio desde dónde se llegó a la página.
  - Id_interna: Para cuando el *referer* es una página de educarchile, se indica la id que lo representa internamente.

- **Dimensión "Pagina"**
  En ella se guardan los datos que caracterizan a las páginas del portal. Cabe mencionar que debido a que las páginas a estudiar están sufriendo constantes cambios, y que estos deben ser guardados en el repositorio, por cada modificación se debe tener un nuevo registro que represente a la página dentro del repositorio
  - Url: corresponde a la dirección de la página.
  - Id_interna: indica la clave interna que representa a la página dentro del sistema de educarchile.
  - Título: indica el título de la página.
  - Contenido: representa el texto de la página.
  - Fecha_creación: indica la fecha en que se creó la página, en formato yyyy:mm:dd.
  - Fecha_modificación: indica la fecha en que la página fue reemplazada por otra, en formato yyyy:mm:dd.
  - Formato: corresponde a la extensión del archivo que representa a la página.

- **Dimensión "Link_to"**
  En ella se guardan los links internos a los que apunta cada página.
  - Pagina_id: indica la clave de la página a la que se apunta.

- **Dimensión "Seccion"**
  Permite registrar la sección de la página en la que se efectuó el *click*. No todas las páginas poseen secciones.
  - Nombre: representa al título de la sección dentro de la página.
  - Pagina_id: indica la clave de la página donde se encuentra la sección.

- **Dimensión "Vineta"**
  Al igual que *Seccion,* esta dimensión permite registrar la viñeta en la que se realizó el *click*. No puede ser parte de *Seccion*, ya que no todas las secciones poseen viñetas.
  - Nombre: representa al nombre de la viñeta dentro de la sección.
  - Sección_id: indica la clave de la sección en donde se encuentra la viñeta.

- **Tabla Fact: petición**
  Corresponde a la tabla donde se registra cada una de las peticiones hechas al servidor, representada por las llaves de las distintas dimensiones. También se guardan las medidas para cada petición, que corresponden al tiempo empleado en la página, la posición en la sesión, y el estado de la petición.
  - fecha_id: clave de la fecha en que fue hecha la petición.
  - tiempo_id: clave de la hora en que fue hecha la petición.
  - sesión_id: clave de la sesión en que fue hecha la petición.
  - pagina_id: id de la página solicitada.
  - referral_id: clave de la página desde donde se hizo la petición, en los casos en que esto ocurra.
  - Seccion_id: apunta a la sección de la página desde donde se hizo la petición al servidor.
  - Vineta_id: clave de la viñeta de la pagina desde donde se realizó la petición al servidor.
  - time_spent: indica el tiempo en segundos que se estuvo en la página solicitada
  - pos_sesion: indica la posición de la petición con respecto a la sesión del usuario.
  - estado: indica el estado de la petición.

- **Tabla Fact: Sesion**
  En ella se almacenan los datos de las sesiones obtenidas a través de la sesionización de los web logs.
  - IP: corresponde a la IP del usuario.

- Agente: Indica el programa utilizado por el usuario para abrir la página.
- Time: tiempo total empleado en la sesión, medido en segundos.
- Largo: cantidad de páginas visitadas en la sesión.
- Secuencia: indica la lista de páginas visitadas durante la sesión, en una cadena de texto.
- idTiempo: apunta a la dimensión Tiempo.
- idFecha: apunta a la dimensión Fecha.
- idReferer: apunta a la dimensión Referer.
- idPagina: apunta a la dimensión Página.

Adicionalmente, se almacenan en el *Webhouse* los datos de las opiniones que servirán como fuente para el estudio basado en las técnicas de *Data Mining*.

- **Tabla Opinion:**
  - Título: representa el nombre principal que le da el usuario a su opinión.
  - Autor: Indica el *nickname* que el usuario registra en su opinión.
  - Fecha: En formato DATETIME, corresponde a la fecha en que se registró la opinión.
  - Opinión: Corresponde a la opinión misma que el usuario publica en el sitio.
  - Clasificación: Representa la clase que se le da a la opinión, que permitirá realizar el entrenamiento de los datos.
  - idPagina: apunta a la página del sitio en que fue vertida la opinión.

3.4. PROCESO ETL

El repositorio de información se obtiene a través del proceso ETL, en el que los datos son extraídos desde las distintas fuentes, luego son procesados para limpiar aquellos que no son útiles, y transformados al formato que se desea, para finalmente ser cargados en el repositorio, siguiendo el modelo de datos presentado anteriormente.

A continuación se explica el diseño del proceso ETL para el presente estudio.

3.4.1. EXTRACCIÓN DE LOS DATOS

Los datos provienen desde tres fuentes, que corresponden a los archivos weblogs, el código fuente de las páginas, y la planilla de cambios de las páginas relevantes del estudio. En la tapa de extracción, los datos son llevados desde su origen hasta la DSA, para luego llevar a cabo el procesamiento en la etapa de transformación.

Weblogs

Los archivos *weblogs* deben ser pre-procesados, con el objetivo de registrar en la DSA todas las peticiones al servidor útiles para el estudio. Así, los pasos que se deben llevar a cabo son:

- Identificar cada atributo de los registros almacenados, como pueden ser la hora, la fecha y el método de la petición. Estos atributos suelen ser separados a través de un espacio en blanco.
- Filtrar los registros correspondientes a peticiones que no son exitosas[9].
- Filtrar todas las peticiones a objetos que no resultan útiles para el estudio, como son las imágenes GIF o JPG, las hojas de estilo en cascada CSS, o los programas javascript.
- Filtrar aquellas peticiones a objetos del tipo *aspx* que no corresponden a páginas del portal, sino que a recursos que son utilizados en búsquedas, o a imágenes insertadas en una página.
- Filtrar las peticiones hechas por web crawlers.
- Filtrar las peticiones duplicadas a páginas que son realizadas por un mismo usuario en un período corto de tiempo, las que pueden ocurrir por la utilización del doble click por parte del visitante, o por una tardía respuesta del servidor.

Adicionalmente, se genera un nuevo atributo para las entradas, que corresponde a la id numérica de las páginas, utilizada en el portal para identificar cada recurso de manera única. Esta id debe ser extraída desde el atributo *query* de los registros.

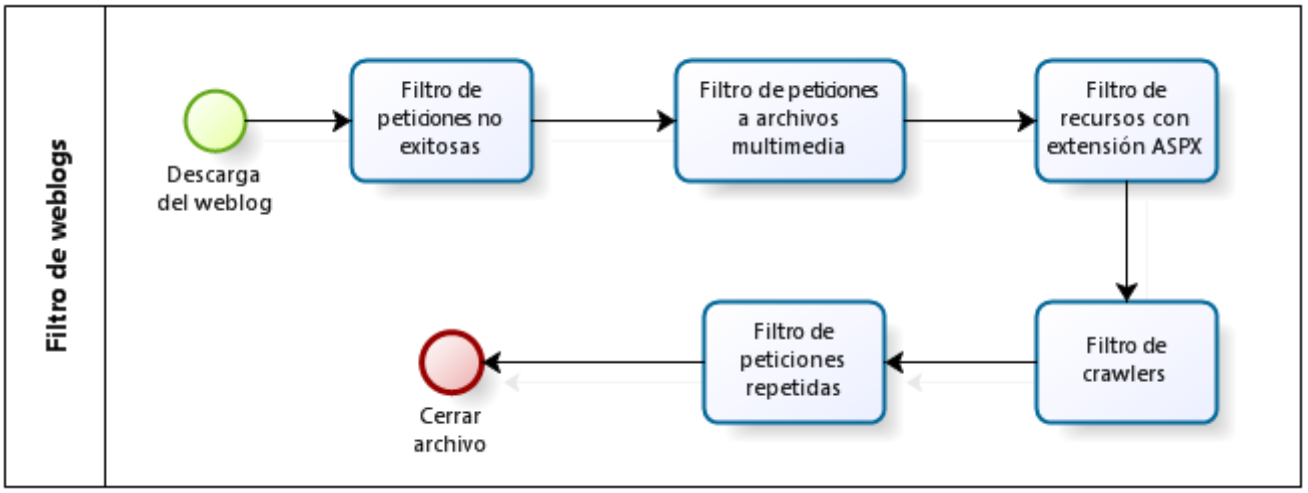

Figura 16: Proceso de filtrado de weblogs. Elaboración propia.

[9] De acuerdo a los protocolos definidos por la W3C, se consideran exitosas las peticiones con status "200", "302" y "304".

Debido a que el portal utiliza 4 servidores web para responder a las solicitudes de los usuarios, son 4 los archivos weblogs que se generan. Por esta razón, una vez realizado el primer filtro de los archivos, se debe llevar a cabo una etapa de unión de las entradas, siguiendo un orden cronológico. Debido a que cada archivo se encuentra ordenado a través del atributo hora, la etapa de unión tiene un costo lineal, ya que por cada inserción se compara el primer elemento de cada archivo que aún no ha sido ordenado.

Con la finalidad de reducir el número de entradas que serán consideradas en el estudio, se ha diseñado un segundo filtro, el cual se queda con todas las peticiones cuya IP en algún instante solicitó al servidor una de las 14 páginas relevantes. Esto no perjudica la medición de los indicadores ajustados al portal, debido a que no se pierden los registros que corresponden a un click hecho sobre una de las páginas a estudiar.

Páginas

Para el código *HTML* de las páginas, el proceso de extracción en este estudio tiene por objetivo identificar tanto el contenido, como los hyperlinks que se encuentran en cada página. Los pasos a seguir son:

- Extraer el código fuente de las páginas a través de un *Web crawler*. Para llevar a cabo esta tarea, se puede indicar al software que comience la extracción desde la página de inicio del portal, para que posteriormente siga a través de los links internos. O bien, se puede indicar que descargue una lista específica de páginas.
- Para cada página se extraen los hyperlinks presentes en ella, lo que se realiza a través de la identificación de los *tags* html encargados de representar un hipervínculo. En esta etapa también se deben clasificar los enlaces en internos (apuntan a una página del sitio), o externos (apuntan a una página de otro sitio).
- Se extrae el contenido de las páginas. Para esto, se deben ignorar aquellos *tags html* que representan a objetos no relevantes para el estudio, como imágenes o archivos. También se debe filtrar aquel código que llama a scripts u hojas de estilo.

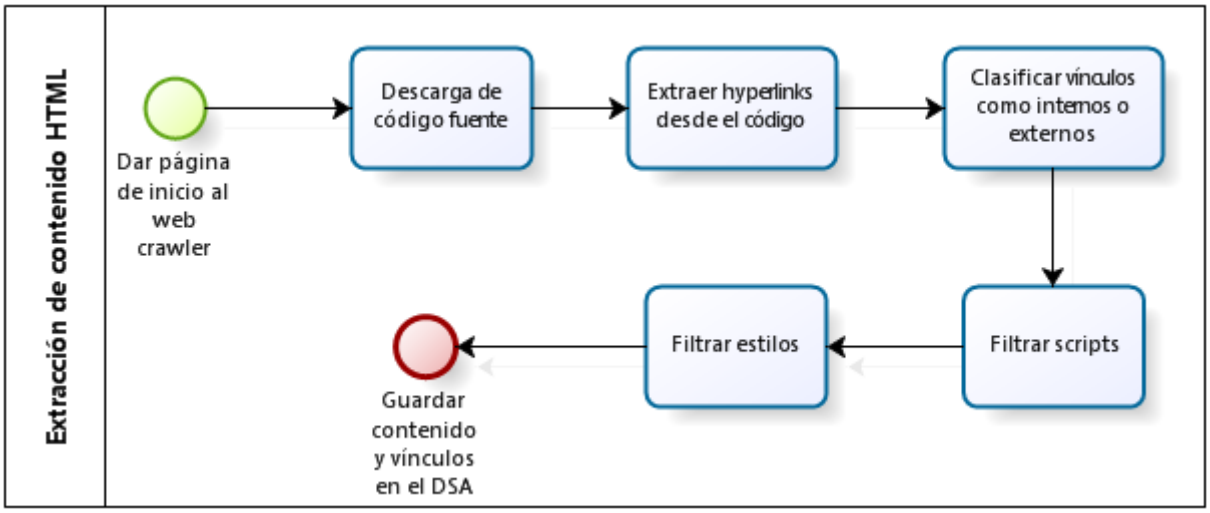

Figura 17: Proceso de extracción de contenido HTML. Elaboración propia.

Cambios en las páginas

Para la detección de los cambios hechos en el contenido de las páginas relevantes del estudio, éstos deben ser identificados a través del código html, y posteriormente verificados en la planilla de datos generada por los administradores de contenido del portal. En caso de haber diferencias, éstas deben ser solucionadas por ambas partes.

3.4.2. TRANSFORMACIÓN DE LOS DATOS

En esta etapa, los datos existentes en la DSA son procesados con la finalidad de ser llevados a los formatos utilizados en el repositorio, como también para poder extraer información acerca de la navegación de los usuarios en el portal. Esto último consiste en reconstruir las sesiones de los usuarios a partir de heurísticas de sesionización.

El proceso tradicional de reconstrucción de sesiones a través de una estrategia reactiva [37] es llevada a cabo en los siguientes pasos:

- Agrupar las entradas almacenadas en el DSA según IP y agente. Esto permite identificar todas las peticiones almacenadas para un mismo usuario. Adicionalmente, la utilización del atributo agente permite diferenciar usuarios que acceden desde una red local – los que acceden a través de una misma IP- siempre y cuando utilicen distintos navegadores para acceder al portal.
- Definir las sesiones a través del intervalo de tiempo en que son hechas las peticiones. La literatura indica que una sesión debería tener una duración máxima de 30 minutos [33]. Esto se traduce en que a un mismo usuario pueden corresponderle distintas sesiones, las cuales estarían distanciadas en un tiempo de al menos 30 minutos.

Por otro lado, las opiniones almacenadas en la DSA debe ser *tokenizadas*. El proceso de tokenización se puede dividir en las siguientes etapas:

1. Eliminar todas aquellas palabras que no entregan significado semántico a la oración, como son artículos y preposiciones. Esta tarea es conocida como borrado de *stopwords*.
2. Llevar las palabras que no fueron filtradas en el paso anterior a su raíz, lo que es conocido como *stemming.* Así, palabras que se escriben de distinta forma, pero que tienen similar raíz semántica, son caracterizadas como sólo un *token*, lo que permite no penalizar la frecuencia de las palabras dentro de un texto por la forma en que están escritas. Un ejemplo de esta tarea son las palabras *comer*, *comida*, y *comieron*, las cuales pueden ser representadas por su raíz *com-*.

El proceso de tokenización debe ser realizado tanto para el contenido HTML de las páginas, como para las opiniones vertidas por los usuarios en el portal.

3.4.3. CARGA DE LOS DATOS

Finalmente, el repositorio de información Web se obtiene a través de la carga de los datos ya procesados en la estructura de almacenamiento definida en la sección 3.3. Para esto, se deben llenar las dimensiones del repositorio con los valores almacenados en la DSA. Un ejemplo de esto corresponde a las páginas y su contenido. Adicionalmente, se deben llenar aquellas tablas que tienen registros definidos y limitados, como son las dimensiones *Tiempo*, *Fecha*, *Calendario*, *Seccion* y *Vineta*. Por otro lado, la tabla *Sesión* debe ser llenada con los resultados obtenidos a través del proceso de sesionización definido en 3.4.2.
Por último, el proceso de carga debe rellenar la tabla Fact *Peticion*, para lo cual debe leer todas las peticiones al servidor almacenadas en la DSA, y buscar las claves primarias que caracterizan a cada campo en el repositorio.

3.5. TEXT MINING

El estudio contempla la búsqueda de un modelo que permita clasificar las opiniones vertidas por los usuarios en el sitio. El proceso ETL almacena las opiniones pre-procesadas en el repositorio de datos diseñado, por lo que los algoritmos a utilizar deben comenzar con la representación de las opiniones a través de *stem's*.

El problema de clasificación a afrontar consiste en la predicción de tres únicas clases: *ayuda, planificación* y *resto de los documentos*; y es muy similar al problema de los filtros anti-spam, donde el objetivo es identificar aquellos correos con contenido comercial o pornográfico, entre otros. En aquel problema se da la existencia de dos únicas clases por predecir. La literatura [31] indica que el clasificador *Naive Bayes* se comporta de buena forma en los problemas de categorización de texto, a pesar de su simpleza. En este sentido, existen diversos casos donde este clasificador es utilizado para afrontar el problema de los filtros

anti-spam [30] [2] [39]. Por esta razón, a continuación se detalla el algoritmo de Naive Bayes, que será utilizado para predecir la categoría de las opiniones del portal educarchile.

3.5.1. NAIVE BAYES

La descripción del clasificador *bayesiano* será a partir de lo señalado por McCallum, Nigam, Thrun y Mitchell en [25]. El clasificador bayesiano asume que los documentos son generados a partir de una distribución de probabilidad que es definida por parámetros θ, que consta de un conjunto de componentes $C = \{c1, \dots, cn\}$, que corresponden a las distintas clases de los objetos. Cada componente de *C* es parametrizada por un set disconjunto de parámetros θ.

La creación de un documento, denotado como d_i, comienza con la selección de un componente de acuerdo a la probabilidad de las clases, $P(c_j/\theta)$, para luego generar el documento a partir de sus parámetros, a través de la probabilidad $P(d_i/c_j;\theta)$. Así, la probabilidad de cada documento se puede definir a partir de la suma de las componentes:

$$P(d_i/\theta) = \sum_{j=1}^{|C|} P(c_j/\theta) * P(d_i/c_j;\theta) \qquad (2)$$

Un supuesto del algoritmo corresponde a que existe una correspondencia uno a uno entre las componentes del modelo y las clases de los documentos. Así, al referirse al componente c_j, también se referirá a la clase j-ésima.

Naive Bayes representa un documento d_i como una lista ordenada de eventos de palabras, $\langle w_{d_{i,1}}, w_{d_{i,2}}, \dots \rangle$, donde $w_{d_{i,k}}$ se refiere a la palabra w_t en la posición *k* del documento d_i, siendo w_t una palabra del vocabulario $V = \langle w_1, \dots, w_{|V|} \rangle$.

Cuando el algoritmo genera un documento a partir de una componente c_j, el tamaño del documento, $|d_i|$, es escogido de forma independiente a c_j. Esto quiere decir que un segundo supuesto de Naive Bayes corresponde a que el tamaño de los documentos es independiente de las clases. Por otro lado, una vez seleccionado el largo del documento, la componente genera una secuencia de palabras del tamaño escogido.

Así, la probabilidad de un documento dada una componente, señalada en la Ecuación 2**¡Error! No se encuentra el origen de la referencia.,** puede ser edefinida a partir de su largo y las palabras del documento:

$$P(d_i/c_j;\theta) = P(\langle w_{d_{i,1}}, \dots, w_{d_{i,|d_i|}} \rangle / c_j;\theta) = P(|d_i|) \prod_{k=1}^{|d_i|} P(w_{d_{i,k}}/c_j;\ \theta;\ w_{d_{i,q}}, q < k) \qquad (3)$$

Como se observa, la probabilidad de una palabra está condicionada por las palabras anteriores. Sin embargo, el supuesto general de Naive Bayes indica que las palabras en un documento son generadas de forma independiente del contexto, lo que se traduce en que las palabras son independientes unas de otras.

Incluso, es usual asumir que la probabilidad de una palabra no depende de su posición. Así, se obtiene:

$$P(w_{d_{i,k}}/c_j;\ \theta;\ w_{d_{i,q}}, q < k) = P(w_{d_{i,k}}/c_j;\ \theta) \qquad (4)$$

Insertando la Ecuación 4 en la Ecuación 3**,** la probabilidad de un documento queda definida por:

$$P(d_i/c_j;\theta) = P(|d_i|)\prod_{k=1}^{|d_i|} P(\ w_{d_{i,k}}/c_j;\ \theta) \qquad (5)$$

Por lo tanto, se obtiene que los parámetros de una componente corresponden a una distribución multinomial sobre las palabras, donde la probabilidad de una palabra, $P(w_t/c_j;\ \theta)$, será denotada por θ_{w_t/c_j}, cumpliéndose que $\sum_t \theta_{w_t/c_j} = 1$.

El modela consta de otros parámetros, los que corresponden a las probabilidades de las clases, denotadas por θ_{c_j}. Estos indican la probabilidad de escoger las distintas componentes.

Los parámetros Θ del modelo no son conocidos, y son estimados a partir del conjunto de documentos de entrenamiento, los que ya se encuentran categorizados. La estimación se denota por $\hat{\theta}$. El algoritmo utiliza el máximo de la estimación a *posteriori*, lo que corresponde al valor de Θ que es más probable a partir de la evidencia del conjunto de entrenamiento.

El resultado de la estimación es el ratio de apariciones empíricas del parámetro. Para el caso de las palabras, $\hat{\theta}_{w_t/c_j}$ es el número de veces que w_t aparece en el conjunto de entrenamiento para la clase c_j, sobre la cantidad total de apariciones de palabras para la clase en el set de entrenamiento. Sin embargo, se utilizan ajustes para aquellos casos donde la poca frecuencia de las palabras provoca una probabilidad cercana a cero.

$$\hat{\theta}_{w_t/c_j} \equiv \frac{1 + \sum_{i=1}^{|D|} N(w_t,\ d_i) * P(y_i = c_j/d_i)}{|V| + \sum_{s=1}^{|V|}\sum_{i=1}^{|D|} N(w_s, d_i) * P(y_i = c_j/d_i)} \qquad (6)$$

Donde $N(w_t,\ d_i)$ indica el número de apariciones de la palabra w_t en el documento d_i, y $P(y_i = c_j/d_i)$ toma los valores $\{0, 1\}$, lo que depende de la clase del documento.

Para el caso de las probabilidades de las clases, la estimación se calcula de la misma forma:

$$\hat{\theta}_{c_j} \equiv \frac{1 + \sum_{i=1}^{|D|} P(y_i = c_j/d_i)}{|C| + |D|} \qquad (7)$$

Una vez realizado el entrenamiento de los documentos, y los parámetros han sido estimados, se procede a utilizar los resultados en la predicción de las clases para nuevos documentos. La clasificación se obtiene a través de la regla de Bayes, y la

sustitución de la Ecuación 2**¡Error! No se encuentra el origen de la referencia.** y a Ecuación 5**:**

$$P(y_i = c_j/d_i;\hat{\theta}) = \frac{P(c_j/\hat{\theta})P(d_i/c_j;\hat{\theta})}{P(d_i/\hat{\theta})} = \frac{P(c_j/\hat{\theta})\prod_{k=1}^{|d_i|}P(w_{d_{i,k}}/c_j;\hat{\theta}\,)}{\sum_{r=1}^{|C|}P(c_r/\hat{\theta})\prod_{k=1}^{|d_i|}P(w_{d_{i,k}}/c_r;\hat{\theta}\,)} \qquad (8)$$

Finalmente, la clasificación selecciona el argumento de la mayor probabilidad:

$$\arg\ max_j\ P(y_i = c_j/d_i;\hat{\theta}) \qquad (9)$$

Lo que se interpreta como la clase con mayor probabilidad a posteriori dado el documento que se desea clasificar, luego de realizar la estimación de los parámetros con el conjunto de documentos de entrenamiento.

CAPÍTULO 4 – APLICACIÓN AL SITIO WEB

En el presente capítulo se expondrá la manera en que el estudio fue llevado a cabo en el sitio web www.educarchile.cl, administrado por la Fundación Chile.

Se detallará la arquitectura tecnológica, el software, los algoritmos, y las estructuras de datos que fueron utilizadas para implementar el proceso ETL, la generación de indicadores en el prototipo, y el desarrollo del modelo de clasificación de opiniones para la sección de debate del portal a partir de un clasificador bayesiano.

4.1. HERRAMIENTAS UTILIZADAS

La selección de las herramientas fue hecha a partir de los costos monetarios que involucraran. Tanto para los lenguajes de programación, el motor de bases de datos relacionales, y el software que permitiera la implementación de cubos OLAP, se utilizaron opciones gratuitas.

MySQL

Las distintas estructuras de datos implementadas, como son la DSA y el Webhouse, fueron desarrolladas bajo el gestor de bases de datos relacionales MySQL. Este gestor es de carácter gratuito para aplicaciones que no son privativas, lo que representa una importante ventaja con respecto a otros gestores de bases de datos relacionales, debido a que el presupuesto del estudio no consideraba costos en este ítem.

Adicionalmente, MySQL es una herramienta multihilo[10] y multiusuario[11]. En cuanto a su velocidad, se comporta extremadamente rápido en bases de datos pequeñas y medianas [12]. Por último, a pesar de ser una herramienta gratuita, sus desarrolladores realizan mejoras constantemente.

Python

Es un lenguaje de programación interpretado[12], interactivo, y basado en objetos. Provee de estructuras de alto nivel, como listas y arreglos asociativos (diccionarios), escritura dinámica, módulos, clases, excepciones, manejo automático de memoria, entre otros. Fue utilizado en este estudio debido a su sintaxis simple y elegante, además de ser un lenguaje poderoso y enfocado a la utilidad [30]. Además, debido a que Python fue diseñado inicialmente para procesar textos, el procesamiento de los archivos weblog se verá facilitado a través de los métodos que ofrece este lenguaje.

[10] Corresponde a cuando una aplicación puede llevar a cabo distintos procesos concurrentemente, compartiendo recursos como memoria o archivos.
[11] En aplicaciones de bases de datos, corresponde a la característica que permite la realización de operaciones a uno o más usuarios a la vez.
[12] Se refiere a que es ejecutado a través de un intérprete, sin la necesidad de un compilador.

Jaspersoft

Es un software perteneciente a *JasperSoft Corporation*, basado en un set de herramientas, las que permiten a las organizaciones generar información a través de la integración de distintas fuentes de datos para la toma diaria de decisiones de manera online. Esto se logra a partir de indicadores generados por medio de técnicas de análisis multidimensional, los que son visualizados en tableros de control y reportes dinámicos, diseñados para ser presentados tanto a analistas como a gerentes. El software cuenta con la ventaja de poseer una versión gratuita, que permite desarrollar proyectos como el presente, sin costo monetario.

En este estudio fueron utilizadas las siguientes herramientas de JasperSoft:

- **Jasper ETL**
  Desarrollado para la implementación del proceso de extracción, transformación y carga de los datos. Permite la elaboración de diseños gráficos, ejecución de movimientos y transformación de datos, útiles para proyectos de *Data Warehousing*. A través de Jasper ETL, es posible desarrollar, manejar y documentar los procesos de integración de datos.

- **JasperAnalysis Workbench**
  Herramienta gráfica desarrollada para la construcción y prueba de schemas OLAP (cubos). Para esto, se definen tanto las relaciones existentes entre la tabla *Fact* y las dimensiones de la base de datos multidimensional, como también los indicadores que serán medidos, las dimensiones del cubo, y las jerarquías de cada una de estas últimas. Su output son archivos XML que representan a los cubos, y que son utilizados por la herramienta JasperServer.
- **Jasper Server**
  Provee de análisis de datos OLAP a los usuarios de negocios, a través de las funciones de drill, pivot y filtro de datos en tiempo real. Incluye un servidor ROLAP, el que utiliza los cubos generados en JasperAnalysys Workbench para realizar las consultas. Además, provee de una interfaz Web para usuarios y analistas sin conocimientos técnicos.

4.2. CARACTERÍSTICAS DE LOS WEBLOGS

El sitio web estudiado cuenta con aproximadamente un millón de sesiones mensuales, de acuerdo a las cifras entregadas por los administradores del portal. El alto número de usuarios genera una gran cantidad de peticiones al servidor web. Por esta razón, el sitio cuenta con 4 servidores que funcionan en forma simultánea, lo que conlleva a la generación de 4 archivos *weblogs* por cada uno.

El período a estudiar corresponde al mes de septiembre, y a las primeras 4 semanas del mes de noviembre, lo que equivale a 56 días. Los archivos weblogs son almacenados de manera semanal, por lo que se procesaron un total de 36 archivos (9 por cada servidor).

Cada archivo weblog cuenta con los siguientes campos:

- **Date** : indica la fecha en que fue realizada la petición, en formato *yyyy-mm-dd.*
- **Time** **:** representa a la hora en que fue realizada la petición, en formato *hh-mm-ss*.
- **S-computername** **:** indica el servidor web que se ocupó de la petición.
- **Cs-method** **:** corresponde al método por cual se realizó la petición.
- **Cs-uri-stem** **:** indica el recurso que fue solicitado.
- **Cs-uri-query** **:** en caso de ser una petición a un artículo, corresponde la id interna con que es identificado.
- **Cs-username** **:** representa el nombre del usuario que realizó la petición. Sin embargo, este campo no tiene entradas válidas.
- **C-ip** **:** indica la ip del usuario que realizó la petición.
- **Cs-user-agent** **:** corresponde al navegador utilizado por el usuario
- **Cs-referer** **:** indica la url del referer, cuando aplica.
- **Cs-host** **:** representa en todas las entradas el host del portal.
- **Sc-status** **:** corresponde al estatus de la respuesta a la solicitud.
- **Sc-bytes** **:** indica los bytes descargados por el usuario.
- **Cs-bytes** **:** corresponde a los bytes enviados por el usuario.

4.3. PROCESO DE EXTRACCIÓN

En esta etapa, los archivos *weblog* son pre-procesados para ser almacenados en la DSA. Además, el set de páginas con opiniones que serán consideradas en el estudio debe ser descargado y almacenado en la DSA.

4.3.1. DSA

Tiene como finalidad guardar en primera instancia los datos provenientes desde las fuentes de información, previo procesamiento de limpieza.

La utilización de la DSA permite trabajar con distintos lenguajes de programación para cada fuente de datos, y posteriormente cargar los datos procesados en el repositorio web [37]. Sin embargo, tanto para la extracción de los archivos *weblogs*, como del set de páginas a considerar, se utilizó el lenguaje de programación Python.

Teniendo en cuenta los indicadores y objetivos que se desean obtener a través del repositorio, el DSA debe almacenar los siguientes datos:

- Los cambios registrados en las páginas a estudiar.
- Las entradas útiles registradas en los *weblogs* del portal.
- Datos que caractericen a las páginas del portal, incluyendo algunos como las secciones y viñetas.
- Las opiniones de los usuarios en los artículos.

Así, se ha diseñado el siguiente modelo relacional para el DSA:

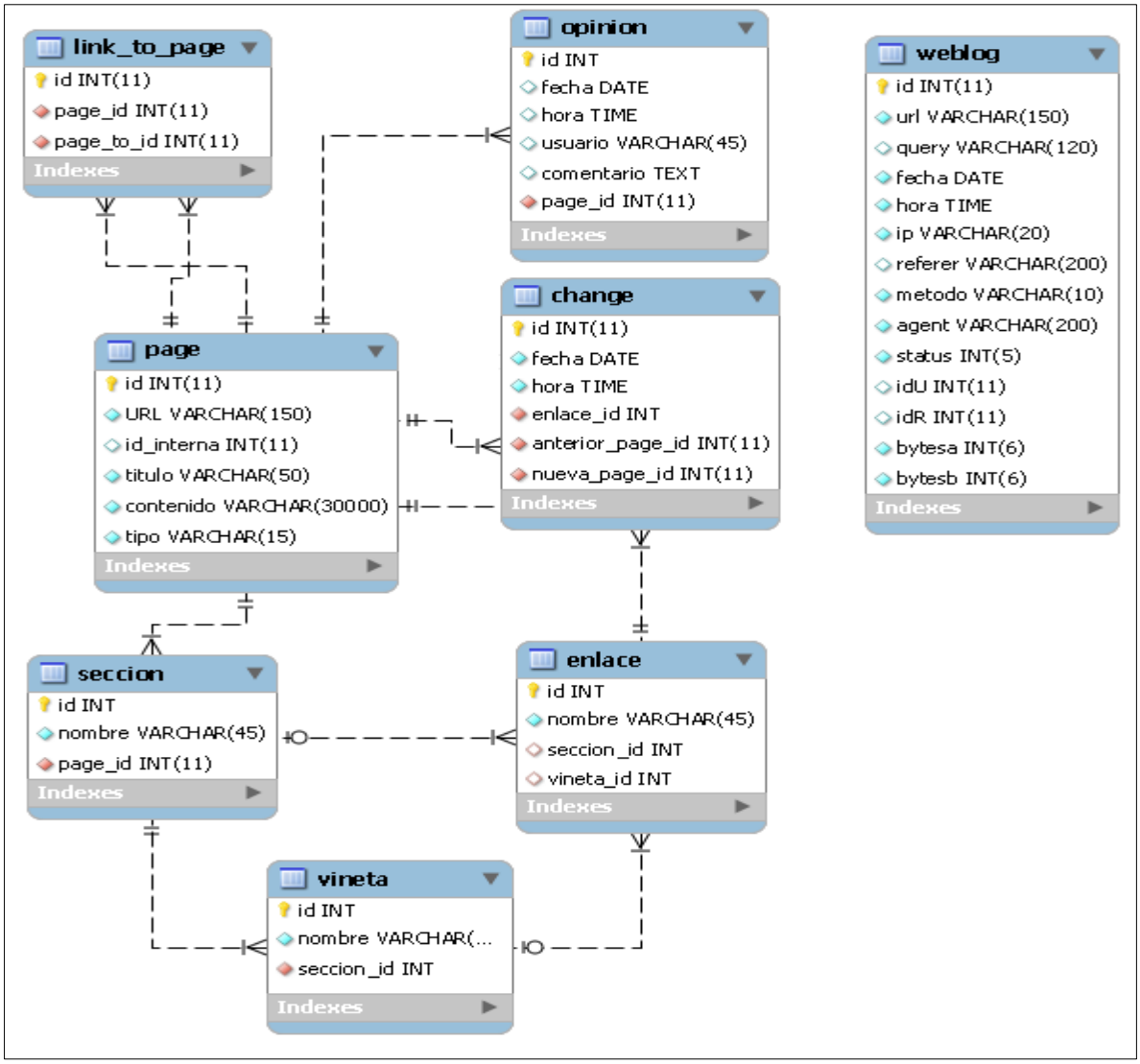

Figura 18: Data Staging Area para los datos del portal. Elaboración propia.

- **Tabla weblog**
  Se almacenan los datos que caracterizan a cada petición válida al servidor web.
  - url : representa la dirección URL base de la página.
  - query : corresponde al recurso del portal solicitado.
  - fecha : día de la petición, en formato DATE.
  - hora : hora del día de la petición, en formato TIME.
  - ip : representa a la ip del cliente que realizó la petición.
  - referer : URL desde donde el cliente hizo la petición del recurso.
  - agent : navegador web utilizado el cliente.
  - status : indica el código que representa el estado de la solicitud.

- idU : corresponde al código interno que representa al recurso solicitado.
- idR : indica el código interno que representa la página desde donde se solicitó el recurso.
- bytesa : cantidad de bytes transmitidos por el cliente al servidor web.
- bytesb : cantidad de bytes transmitidos por el servidor web al cliente.

- **Tabla page**
  En esta tabla se almacenan los datos que caracterizan a las páginas del portal.
  - url : dirección URL de la página.
  - id_interna : código interno del portal que representa a cada página.
  - titulo : título de la página.
  - contenido : corresponde al texto que se visualiza en la página.
  - tipo : formato de la página (*aspx, html*, entre otros).

- **Tabla link_to_page**
  Aquí se almacenan los hipervínculos entre las páginas del portal.
  - page_id : indica la clave primaria de la página donde se ubica el hipervínculo.
  - page_id_to : indica la clave primaria de la página a la que apunta el hipervínculo.
- **Tabla sección**
  En esta tabla se almacenan las distintas secciones de las páginas relevantes del estudio.
  - nombre : indica el nombre de la sección.
  - page_id :indica la clave primaria de la página donde se encuentra la sección.
- **Tabla vineta**
  Utilizada para almacenar las viñetas que componen a cada sección de la tabla anterior.
  - nombre : corresponde al nombre de la viñeta
  - sección_id : indica la clave primaria de la sección que contiene a la viñeta.
- **Tabla enlace**
  En ella se almacenan los hipervínculos que se encuentran en las secciones de las páginas relevantes del estudio. Permite asociar un link a una sección o viñeta.
  - nombre : nombre que caracteriza al enlace.
  - seccion_id : indica la clave primaria de la sección donde está ubicado el enlace.

  - vineta_id : indica la clave primaria de la viñeta donde está ubicado el enlace
- **Tabla change**
  Aquí se almacenan los cambios en los enlaces de las páginas relevantes, lo que permite identificar los clicks hechos en cada enlace, y por consiguiente, en las secciones y viñetas.
  - fecha : día en el que ocurre el cambio, en formato DATE.
  - hora : hora del día en que ocurre el cambio, en formato TIME.
  - Enlace_id : indica la clave primaria del enlace donde ocurre el cambio.
  - anterior_id : indica la clave primaria de la página que se apuntaba en el enlace antes del cambio.
  - nueva_id : indica la clave primaria de la página que es apuntada en el enlace luego del cambio.
- **Tabla opinión**
  Aquí se almacenan los datos que caracterizan las opiniones vertidas por los usuarios en la sección de debate del portal.
  - fecha : día en el que se emitió la opinión, en formato DATE.
  - hora : hora del día en que se emitió la opinión, en formato TIME.
  - usuario : nombre que el usuario indica en la opinión.
  - comentario : corresponde a la opinión misma del usuario.
  - pagina_id : indica la clave primario de la página en que se emitió la opinión.

4.3.2. ARCHIVOS WEBLOG

El proceso de extracción de los archivos *weblogs* consta de 5 etapas. La primera de ellas corresponde al filtro de entradas válidas. La segunda al ordenamiento de los 4 archivos generados por los servidores, de acuerdo a la fecha y hora de las peticiones. Posteriormente se realizó un segundo filtro, que reduce el número de peticiones a considerar en el estudio, de acuerdo a si el usuario visitó alguna de las 14 páginas relevantes para el estudio. En la cuarta etapa, se almacenan en la DSA aquellas páginas, referer's, IP's y agentes que caracterizan las peticiones al servidor, y que aún no son registradas en la DSA. Por último, las peticiones del archivo *weblog* son almacenadas en la DSA.

4.3.2.1. Filtro de entradas válidas

Este filtro recibe un archivo *weblog*, y entrega como output un archivo de texto para cada día procesado, donde sus líneas representan las entradas válidas al servidor.

La implementación de este filtro sigue las siguientes etapas:

1. Se abre el archivo weblog.

2. Se ignoran las primeras 4 líneas, que contienen información acerca del archivo.
3. Se lee la siguiente línea.
4. Se extraen los atributos de la entrada, a través de la función *split()* de la clase *String* de Python.
5. Se corrobora que el status de la respuesta sea válido[13]. En caso contrario, se vuelve al paso 3.
6. Si la entrada corresponde a un nuevo día, se abre un nuevo archivo de texto.
7. La entrada no es considerada si el recurso solicitado (campo *stem*) corresponde a una imagen, hoja de estilo css, archivo pdf, entre otros[14]. En ese caso, se vuelve al paso 3. El tipo de recurso es identificado a través de su extensión.
8. Se verifica que no corresponda a un recurso que es utilizado dentro de las páginas[15]. En ese caso, se vuelve al paso 3.
9. Se verifica que no corresponda a una solicitud hecha por un web *crawler*. Esto es implementado a través de una lista de *bots*, y de la búsqueda de palabras claves en el atributo *agent*[16]. En caso de ser una entrada no válida, se vuelve al paso 3.
10. Si la petición es válida, se extrae la id interna de la página solicitada, la que se encuentra en el campo *query*. Adicionalmente, se extrae la id interna del *referer*, en caso de ser una página del sitio.
11. La petición es ignorada en caso de estar repetida en las últimas 15 registradas. Esto es realizado para evitar duplicaciones en el *weblog* producto que el usuario realiza varios clicks para solicitar la página. En tal caso, se vuelve al paso 3.
12. La petición es almacenada en el archivo de texto del día. Adicionalmente a los campos que se incluyen en el *weblog*, se registran las id's internas de la página solicitada y del *referer*.
13. Si no es la última línea, se vuelve al paso 3.

4.3.2.2. Ordenamiento de los archivos

En la etapa anterior se obtienen 4 archivos de texto por cada día procesado. *Aquí* los archivos diarios son unificados, siguiendo un ordenamiento ascendente de acuerdo al campo *hora*. Debido a que los archivos vienen ordenados, esta etapa tiene un costo lineal, ya que por cada registro almacenado, se realiza sólo una comparación.

La implementación de esta etapa sigue los siguientes pasos para cada día procesado:

[13] Se consideran válidos los status "200", "302" y "304".
[14] La lista de archivos no válidos son: gif, js, ico, css, jpg, asp, xml, ppt, bmp, doc, zip, swf, png, txt, mp3, peg, rar, mov, wmv, pps, rtf, exe, flv, xls, mpg,kmz, axd, dll.
[15] Se elaboró una lista de 9 recursos que no son válidos, que en su mayoría son buscadores y calendarios que son utilizados dentro de páginas del portal.
[16] Las palabras claves consideradas fueron: spider, bot, crawler.

1. Se abren los 4 archivos generados en el filtro de entradas válidas, y se abre un archivo donde se almacenarán las entradas ordenadas.
2. Se lee la primera línea de cada archivo.
3. Se comparan las horas de las 4 líneas, y se almacena aquella línea con hora de petición menor.
4. Si el archivo cuya línea fue almacenada posee más peticiones, se lee un nuevo registro y se vuelve al paso 3. En caso contrario, si todos los archivos fueron recorridos completamente, el algoritmo para. Si no, se repite el paso 3 con los archivos que aún no han sido almacenados completamente.

4.3.1.3. Filtro de páginas relevantes

El último procesamiento de las entradas almacenadas en los archivos *weblogs* corresponde a un segundo filtro de peticiones, el que elimina aquellas sesiones que no visitan alguna de las 14 páginas relevantes para el estudio, con el objetivo de reducir el procesamiento de datos en las siguientes etapas del proceso ETL, sin perder información relevante para la generación de los indicadores de uso del portal, definidos en 3.1.1.

Este filtro identifica sesiones sólo a través del atributo IP de las peticiones, y su implementación sigue los siguientes pasos para cada día:

1. Se abre el archivo donde se encuentran almacenadas las peticiones, y un nuevo archivo donde se registrarán las entradas que pasen este filtro.
2. Se lee una línea del archivo de peticiones.
3. Si la entrada corresponde a una de las 14 páginas relevantes del estudio, se agrega la IP del usuario a un arreglo de IP's.
4. Mientras haya otra línea, se repite el paso 3. En caso contrario, el archivo se cierra.
5. Se vuelve a abrir el archivo de peticiones.
6. Se lee una línea.
7. Se busca la IP de la petición en el arreglo de IP's[17]. SI se encuentra en él, se almacena la petición en el nuevo archivo.
8. Se vuelve al paso 6 si aún quedan peticiones por procesar.

4.3.1.4. Registro de dimensiones en la DSA

Previo registro de las entradas del *weblog* en la DSA, es necesario almacenar las páginas, referer's, IP's y agentes que caracterizan cada petición en sus respectivas tablas. Este proceso necesita recorrer completamente el archivo, buscando en la DSA si los atributos ya han sido almacenados. Para reducir el tiempo de ejecución, cada inserción en las tablas considera 100 registros, lo que mejora considerablemente la eficiencia de la etapa.

[17] Por cada búsqueda, se recorre el arreglo de IP's registro a registro hasta que se encuentre la IP del usuario.

La implementación por cada archivo *weblog* procesado [18], sigue los siguientes pasos:

1. Se abre el archivo de peticiones.
2. Se realiza la conexión con la DSA.
3. Se lee una línea del archivo.
4. Se extraen los campos de la petición.
5. Se busca si el *referer* corresponde a una página del portal.
6. Se busca la página solicitada en la tabla *Página*. En caso de no estar, se almacena en ella.
7. Se repite el paso 6 para la IP y el agente de la solicitud.
8. Si el *referer* es una página del portal, se busca en la tabla *Referer*, y se almacena en caso que no esté. Si no es una página del portal, siempre se almacena.
9. Si el archivo no ha sido recorrido completamente, se vuelve al paso 3.

4.3.1.5. Registro del archivo weblog en la DSA

El proceso de extracción de los archivos weblog's finaliza con el almacenamiento de las peticiones en la DSA. La principal característica de esta etapa es que los campos *Pagina, Referer, IP* y *Agente* de las peticiones, apuntan, a través de una llave foránea, a un registro de sus respectivas tablas de la DSA. Esto es útil dado que en la etapa de transformación de los datos, la sesionización trabajará sólo con valores numéricos, lo que facilita la comparación de peticiones (por sobre valores en formato texto). Al igual que en la etapa anterior, cada inserción en la tabla *weblog* considera 100 registros.

Los pasos que sigue esta etapa, por cada día procesado, son:

1. Se abre el archivo weblog.
2. Se realiza la conexión con la DSA.
3. Se lee una línea del archivo.
4. Se extraen los campos de la petición.
5. Se busca el agente en la tabla *Agente*, y se guarda el valor de la id del registró encontrado.
6. Se repite el paso 5 para la IP, página, y *referer*[19]
7. Se almacena la petición en la DSA. Si el archivo no ha sido recorrido por completo, se vuelve al paso 3.

4.3.2. OPINIONES

La extracción de las opiniones tiene por objetivo almacenar en la DSA los comentarios vertidos por los usuarios en el portal. Por esta razón, el primer paso corresponde a la descarga de las páginas que poseen comentarios, cuyo código

[18] Mediante las 3 etapas anteriores, se obtuvo un único archivo weblog por cada día procesado.
[19] Las búsquedas de página y *referer* se realizan a través de las id's internas que las representan en el portal, que poseen la ventaja de ser numéricas.

html contiene las opiniones de los usuarios. Para esto, se utilizó el módulo *urllib*, que es parte de la biblioteca estándar de Python. El programa recibe la lista de url's que se deben descargar, para luego almacenar el código html de la página en un fichero de texto.

El segundo paso corresponde a la extracción de los atributos de las opiniones desde el código html. Para esto, se desarrolló un programa que detecta el set de etiquetas que caracterizaban a una opinión dentro del código. Una vez realizada esa tarea, el programa almacena la opinión dentro de la DSA.

4.4. TRANSFORMACIÓN DE LOS DATOS

En esta sección se detalla la implementación del proceso de transformación de datos, el que consta de 3 elementos: la tokenización de las opiniones almacenadas en la DSA, la reconstrucción de la navegación de los usuarios, y la asociación de las peticiones a secciones y viñetas de las páginas relevantes del estudio.

4.4.1. TOKENIZACIÓN DE LAS OPINIONES

El proceso de transformación de las opiniones comienza con el borrado de tildes, tarea que es realizada a través del módulo *unicodedata* de la biblioteca de Python.

El segundo paso corresponde al borrado de stopwords. Para esto se desarrolló un programa en Python que recorre cada opinión, palabra por palabra, y las compara con un fichero de texto que contiene la lista de *stopwords* en idioma español[20].

Una vez que los dos pasos anteriores han sido realizados, se procede a realizar el *stemming* de las opiniones. La implementación utilizó el algoritmo *snowball* para el idioma español, que se basa en la búsqueda de sufijos en las palabras. Para esto, se ocupó el algoritmo desarrollado en PHP que se presenta en la página Sourceforge.net[21], y se reescribió en el lenguaje Python.

4.4.2. RECONSTRUCCIÓN DE LAS SESIONES

La reconstrucción de las sesiones de los usuarios está basada en la combinación *IP-Agente* del usuario. Sumado a esto, un parámetro importante para el algoritmo es el que considera como tiempo máximo de una sesión los 30 minutos.

La implementación de esta etapa hace uso de dos nuevas tablas en la DSA. En la primera se almacenan los campos que caracterizan a una sesión, como es el

[20] La lista de stopwords utilizada se encuentra en http://www.elwebmaster.com/referencia/stopwords-en-espanol

[21] La implementación en PHP se puede encontrar en http://stemmer-es.sourceforge.net/

número de páginas involucradas, el *referer*[22], el tiempo de duración de la sesión, la secuencia de páginas visitadas, entre otros. En cambio, la segunda tabla asocia una petición de la tabla *Weblog* con una sesión de la tabla *Sesión.*

Así, el proceso de sesionización sigue los siguientes pasos:

1. Se realiza la conexión con la DSA.
2. Se seleccionan todas las combinaciones IP-Agente distintas de la tabla *Weblog* que correspondan al día al que se aplicará el algoritmo.
3. Por cada registro del resultado del paso 2:
   3.1. Seleccionar todas las peticiones donde coincidan la IP y el agente con el resultado.
   3.2. Iniciar una nueva sesión
   3.3. Por cada registro en este nuevo resultado:
      3.3.1. Calcular el tiempo total de la sesión. Si supera los 30 minutos, iniciar una nueva sesión.
      3.3.2. Si el número de peticiones de la sesión es mayor a uno, actualizar el registro de la petición anterior en la tabla *Sesion*, con el tiempo que el usuario visitó la página[23].
      3.3.3. Registrar en la tabla *Sesion* la petición que se itera[24].
4. Seleccionar todas las peticiones asociadas a cada sesión.
   4.1. Actualizar el tamaño de la sesión de acuerdo al número de peticiones involucradas.

4.4.3. PETICIONES A SECCIONES Y VIÑETAS

Uno de los principales requerimientos de información del estudio es la medición de indicadores relacionados con el uso de las secciones y viñetas de las páginas relevantes del portal.

El desafío que representaba el logro de este objetivo era la imposibilidad de asociar un click directamente a una sección o viñeta a partir de los datos contenidos en los archivos *weblogs*. A esto se suma que el contenido que es publicado en las páginas es dinámico, existiendo una alta rotación de artículos. Por esta razón, era necesaria la existencia de una planilla de datos, o base de datos, que contuviera todos los cambios que se dieran en las secciones y viñetas a estudiar.

La obtención de los cambios generados en las páginas se implementó a través de dos formas distintas:

- Mediante un acuerdo con la encargada de publicación de contenido de las páginas consideradas en el estudio, se definió que el equipo del portal

[22] El que corresponde al *referer* de la primera página visitada en la sesión.
[23] El tiempo es calculado como la diferencia entre la hora de la petición que se itera con la hora de la petición anterior.
[24] El tiempo que se asigna al momento de la inserción es 0, el que es actualizado en la siguiente iteración.

generaría una planilla de datos que contuviera todos los datos asociados a los cambios generados en las secciones y viñetas. Esta planilla era compartida a través de la tecnología *Google Docs*, y se registraron los cambios de los meses de septiembre, octubre, y noviembre.

- Se generó un programa, bajo el lenguaje de programación Python, que detectaba los cambios en cada una de las viñetas y secciones de las páginas consideradas en el estudio. Este programa se ejecutó durante todas las noches del período que abarca desde el 24 de octubre hasta el 30 de noviembre. Los resultados fueron almacenados en la DSA.

Una vez detectados los cambios de forma automática, se observaron diferencias entre lo que se registraba en la planilla de datos generada por el equipo del portal, y lo que se detectaba a través del programa. Debido a esto, se definió que el mes de octubre no sería considerado en el estudio, ya que los datos de la planilla no eran fiables para ese mes. Septiembre sí fue considerado, bajo la hipótesis que el registro de los cambios sí fue realizado de manera óptima durante ese mes, por ser el primero donde se realizaba la tarea. Los cambios para ese período fueron almacenados en la DSA. Por último, para el mes de noviembre, se consideraron los datos generados por el programa desarrollado.

Así, se utilizaron los cambios registrados en la DSA para asociar los registros de la tabla *Weblog* con clicks de usuarios en las secciones y viñetas de las páginas relevantes. La implementación siguió los siguientes pasos:

1. Se realiza la conexión con la DSA.
2. Se seleccionan todos los registros de la tabla *Weblog* cuyo *referer* corresponda a una de las páginas relevantes del estudio.
3. Por cada uno de los resultados:
   3.1. Seleccionar los registros de la tabla *Publicación* que correspondan a un enlace contenido en el *referer* del resultado, y cuyo período de publicación incluya la fecha del resultado.
   3.2. Actualizar el registro de la tabla *Weblog* con la sección y viñeta que indica el resultado de la consulta anterior[25].

4.5. CARGA DE LOS DATOS

Este proceso se llevó a cabo con la herramienta Jasper ETL, perteneciente al paquete de JasperSoft, y consistió en traspasar los datos almacenados en la DSA hacia la base de datos multidimensional. Su implementación siguió los siguientes pasos:

Creación de un nuevo proyecto en JasperETL

El traspaso de los datos debe estar contextualizado dentro de un proyecto de la herramienta. Es un paso sencillo, en donde se define el nombre, el tipo de

[25] Del resultado se extrae la llave de la sección, la llave de la viñeta, y su condición: activa o inactiva.

conexión (en este caso local), y el lenguaje que se desea utilizar, que en este caso fue Perl, el cual es más robusto dentro de la herramienta que la otra opción, Java.

Creación de un nuevo trabajo

La carga es diseñada dentro un trabajo. Por esto, el segundo paso consiste en crear un nuevo trabajo dentro de la herramienta.

Carga de las dimensiones

Una vez dentro del trabajo, se diseña el proceso de carga, que debe comenzar por las dimensiones, para finalizar con la fabla *Fact*.

Por cada una de las dimensiones del Data Mart, en el trabajo se debe indicar un proceso de carga. Para esto, la herramienta utiliza 2 objetos: *tMysqlInput* y *tMysqlOutput*, que representan, para los datos, la extracción desde el origen, y la introducción en el destino, respectivamente. La serie de pasos que se desarrollaron para llevar a cabo la carga de cada dimensión fue:

1. Se indican los parámetros del servidor MySQL para el objeto tMysqlInput.
2. Realizar el mapeo de los datos. En él se deben indicar qué atributos de la tabla que representa a la dimensión en la DSA se considerarán en la carga.
3. Guardar el mapeo en un archivo XML.
4. Cargar el archivo XML en el objeto tMysqlInput.
5. Conectar ambos objetos a través de la funcionalidad “row-main”.
6. Configurar el objeto tMysqlOutput. Se debe indicar la base de datos de destino, y la tabla que representará a la dimensión.
7. Finalmente, se sincroniza la carga de los datos entre ambos objetos.

Carga de la tabla Fact

La carga de la tabla Fact en la herramienta es muy similar a la de las dimensiones. Primero se seleccionan los objetos tMysqlInput y tMysqlOutput, y posteriormente se desarrollan los siguientes pasos:

1. Se indican los parámetros del servidor *MySQL* para el objeto *tMysqlInput*.
2. Para *tMysqlInput,* se indican todas las relaciones existentes entre sus tablas (llaves foráneas).
3. Se realiza el mapeo de los datos. En este caso, de deben indicar los atributos de cada tabla que serán utilizados en la tabla Fact.
4. Guardar el mapeo en un archivo XML.
5. Utilizar el archivo XML en el objeto *tMysqlInput.*
6. Conectar ambos objetos a través de “*row-main*”.
7. Configurar el objeto de destino *tMysqlOutput.*
8. Sincronizar la carga de los objetos entre ambos elementos.

Para el desarrollo de este estudio, debido a limitaciones de procesamiento y de memoria, anterior a la carga de la tabla Fact en la base de datos de destino se realizaron las siguientes transformaciones dentro de la DSA:

1. Se realizó el cruce entre la tabla *sesion, fecha,* y *weblog*, para crear una tabla *Fact* temporal en la DSA, que contenía las llaves de todas las dimensiones, a excepción de tiempo. Adicionalmente, al considerarse *sesion* en el cruce, se extraen todos los atributos del registro asociados a la sesión del usuario, como son *ord*, *id_sesion*, y *time_spent.*
2. Se realizó el cruce de la tabla recién creada con tiempo. La nueva tabla temporal contiene todos los atributos relacionados con las dimensiones, referenciando a sus respectivas tablas a través de llaves foráneas.
3. Se realiza una carga directa de la última tabla creada hasta la tabla *Fact* de la base de datos de destino.

4.6. CREACIÓN DE CUBOS

Para el análisis de indicadores se consideraron dos cubos: *sesiones*, y *sección y viñetas*. El primero tiene por objetivo la medición de indicadores de uso del portal, asociados a las sesiones de los usuarios, como son el número de sesiones en la dimensión fecha, el tiempo promedio de las sesiones, y el tiempo promedio de las sesiones, entre otros. El segundo cubo tiene por objetivo la generación de indicadores de uso de las secciones y viñetas.

Para la generación de los cubos se utilizó la base de datos definida en 3.3. La implementación, realizada a través de *JasperAnalysis Workbench* y *Jasper Server*, siguió los siguientes pasos:

Creación del Schema con JasperAnalysis-Workbench

La herramienta permite mapear la base de datos multidimensional con las *fact's tables* y las dimensiones, a través de la generación de un *schema* lógico, en el que se definen cubos, dimensiones, jerarquías, niveles, y miembros.

La creación de los *schemas* fue hecha mediante las siguientes etapas:

1. Realizar la conexión con la base de datos, y guardarla.
2. Abrir un nuevo *schema*, utilizando la conexión definida en 1.
3. Crear un nuevo cubo, asignándole un nombre.
4. Asociar el cubo a una *Fact table*.
5. Añadir los indicadores. Para esto se asocia un atributo de la *Fact table* con una medida[26].
6. Añadir las dimensiones. Se debe indicar cuál es la llave foránea que la representa en la *Fact Table*.
7. Se asocian las jerarquías correspondientes a las dimensiones.
8. Si el cubo lo amerita, se asocia a una jerarquía distintos niveles.
9. Se guarda el *schema*, el que es representado a través de un archivo XML.

Opcionalmente, el programa permite realizar consultas en lenguaje MDX para probar los esquemas generados.

La creación física del cubo, y sus diferentes vistas, es realizada a través de la herramienta *Jasper Server*.

[26] La herramienta consta de medidas tales como suma, promedio, y cantidad.

Creación de un Datasource con Jasper Server

El paso siguiente a la creación del *schema* lógico del cubo, consiste en utilizar *Jasper Server* para seleccionar la base de datos a utilizar como fuente (datasource). Este es un paso sencillo, y en el que se aconseja realizar una prueba para verificar la correcta conexión con la base de datos.

Activar el schema en Jasper Server

El siguiente paso en la creación de un cubo en *Jasper Server*, consiste en indicar cuál será el schema a utilizar, el que fue creado en la primera etapa a través de *JasperAnalysis Workbench.*

Crear una Analysis client connection con Jasper Server

En este paso se realiza una conexión cliente, que consiste en asociar el *datasource* con el *schema* ya cargado en *Jasper Server*. Esta conexión es del tipo *Mondrian*.

Crear un Analysis view con Jasper Server

El último paso en la creación del cubo radica en definir las vistas de análisis que tendrá el usuario, que consisten en consultas MDX. Para esto, se carga la conexión cliente creada en el paso anterior, y se añade la consulta MDX del indicador que se desea medir.

Este paso es realizado tantas veces como indicadores se hayan definido para el cubo.

4.7. NAIVE BAYES

Para la construcción del clasificador de opiniones se utilizó el algoritmo *Naive Bayes* aplicado a texto, a través del lenguaje de programación Python 2.7. Dado que *Naive Bayes* utiliza documentos pre-clasificados para realizar el entrenamiento de los datos, el primer paso consistió en clasificar el conjunto de opiniones según 3 categorías:

- Ayuda
- Planificación
- Resto de los comentarios

Posteriormente, se aplicó el algoritmo para entrenar los datos, el que sigue los siguientes pasos:

1. Se inicializa una categoría.
2. Se abre un archivo de la clase.
3. Se lee una palabra del documento:
    a. Si es parte del vocabulario del set completo de documentos, se aumenta en 1 su frecuencia. En caso contrario, se agrega al vocabulario y se define su frecuencia en 1.

   b. Si es parte del vocabulario de la clase, se aumenta en 1 su frecuencia. En caso contrario, se agrega al vocabulario y se define su frecuencia en 1.
4. Si quedan palabras, volver al paso 3. En caso contrario, continuar.
5. Si quedan archivos, volver al paso 2. De lo contrario, pasar a 6.
6. Si restan clases por entrenar, se vuelve al paso 1.

Posteriormente se realiza la clasificación de los documentos. En ella simplemente calculan los términos de la Ecuación 8, basándose en los vocabularios obtenidos en la etapa de entrenamiento de los datos.

CAPÍTULO 5 – RESULTADOS

En este capítulo se presentan los resultados obtenidos en el trabajo realizado. En primer lugar se muestra el rendimiento del proceso ETL para los datos. Luego se indican los principales indicadores de uso del portal obtenidos a partir del repositorio de datos. Y finalmente, se indican los resultados obtenidos de los filtros aplicados sobre las opiniones.

5.1. PROCESAMIENTO DE LOS DATOS

El trabajo consideró un período de evaluación que corresponde a los meses de septiembre y noviembre. Para esto, de procesaron un total de 36 archivos *weblog*, compuestos en 9 para cada uno de los 4 servidores que utiliza el portal. El procesamiento se inició con los datos del mes de noviembre, debido a que se tenía información fiable acerca de los cambios producidos en las páginas a estudiar, tal como se indicó en 4.4.3.

Las siguientes tablas están elaboradas a partir del total de peticiones hechas a los 4 servidores durante la semana del 12 al 18 de septiembre, que ascienden a 71.078.293 entradas.

Status	Cantidad	Porcentaje
200	61.590.628	86,65%
204	0	0,00%
206	0	0,00%
301	17.505	0,02%
302	369.889	0,52%
304	8.078.698	11,37%
400	456	0,00%
401	16	0,00%
404	158	0,00%
405	0	0,00%
500	9.935	0,01%

Tabla 3: Status de las peticiones hechas a los weblogs.

Archivo	Cantidad	Porcentaje	Archivo	Cantidad	Porcentaje	Archivo	Cantidad	Porcentaje
GIF	29.133.587	40,99%	**BMP**	40.118	0,06%	**MOV**	721	0,00%
CSS	23.882.672	33,60%	**DOC**	60.018	0,08%	**WMV**	35	0,00%
ASP	1.247.866	1,76%	**ZIP**	6.601	0,01%	**PPS**	861	0,00%
JS	3.764.675	5,30%	**PDF**	85.294	0,12%	**EXE**	2.316	0,00%
ICO	419.456	0,59%	**TXT**	4.420	0,01%	**FLV**	896	0,00%
JPG	6.101.505	8,58%	**MP3**	154.880	0,22%	**RTS**	319	0,00%
PNG	329.055	0,46%	**JPGE**	266.762	0,38%	**XLS**	536	0,00%
XML	190.824	0,27%	**SWF**	375.177	0,53%	**MPG**	0	0,00%
PPT	28.066	0,04%	**RAR**	243	0,00%	**KMZ**	1.598	0,00%
AXD	90.471	0,13%	**DLL**	179	0,00%			
						Total	**65.188.177**	**91,71%**

Tabla 4: Filtro de archives no útiles para el estudio.

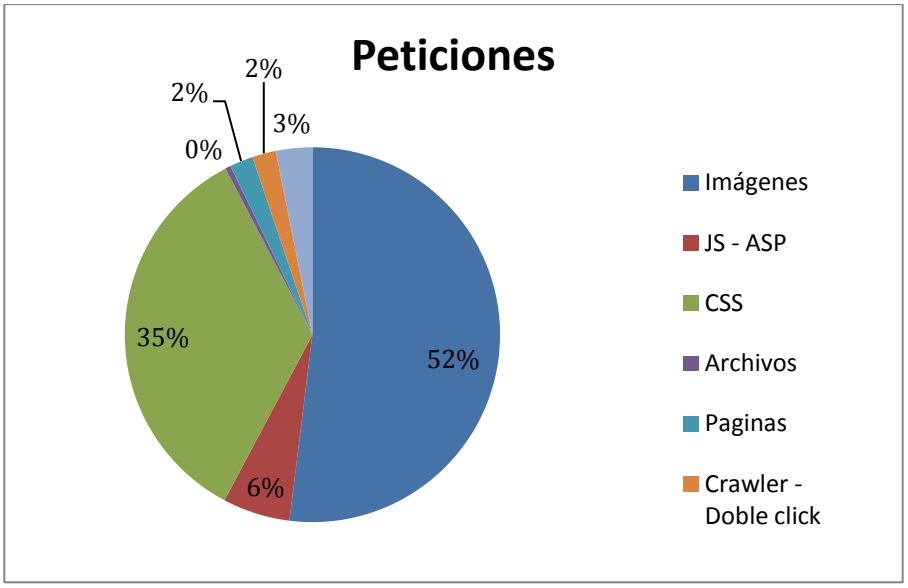

Gráfico 1: Resultado del filtro de entradas útiles.

De acuerdo a la Tabla 3, aproximadamente un 98,44% de las solicitudes hechas por los visitantes fue respondido de manera correcta[27], lo que habla muy bien del rendimiento de los servidores utilizados por el portal.

Por otro lado, se observa que un 91,71% de las solicitudes corresponde a archivos que no resultan útiles para el estudio, ya que no representan a una página particular del portal. De acuerdo a la Tabla 4, más de un 49% de las solicitudes representan imágenes del tipo *GIF* y *JPG.* Y además, un 33,6% corresponde a hojas de estilo CSS.

Adicionalmente, se detectó que alrededor de un 1,05% de las peticiones hechas al servidor son hechas por *web crawlers*.

El factor resultante de los dos filtros aplicados sobre los weblog's fue de un 0,77%. Considerando que se almacenaron un total de 4.795.543 de peticiones, **el total de entradas procesadas fue de 622 millones aproximadamente**.

5.2. INDICADORES

La hipótesis del presente estudio planteaba que el procesamiento de los archivos *weblog* del portal permitiría, a través de un *Data Mart*, determinar indicadores de uso del sitio, tanto tradicionales como ajustados a las necesidades de análisis de educarchile. En concreto, permitiría conocer el uso de las viñetas y secciones de las páginas relevantes para el estudio.

A continuación se presentan los principales resultados obtenidos a través de la realización de consultas MDX sobre los datos almacenados en el *Webhouse*.

[27] Se consideran exitosos los status 200, 302, y 304.

5.2.1. INDICADORES TRADICIONALES

La implementación del cubo "sesiones" tenía por objetivo obtener indicadores tradicionales asociados al uso de un portal web. Tal como se puede apreciar en el Gráfico 2, a través de *Jasper Server* fue posible acceder a indicadores relacionados con las sesiones de los usuarios del portal, tanto en la dimensión *Tiempo,* como en la dimensión *Fecha.*

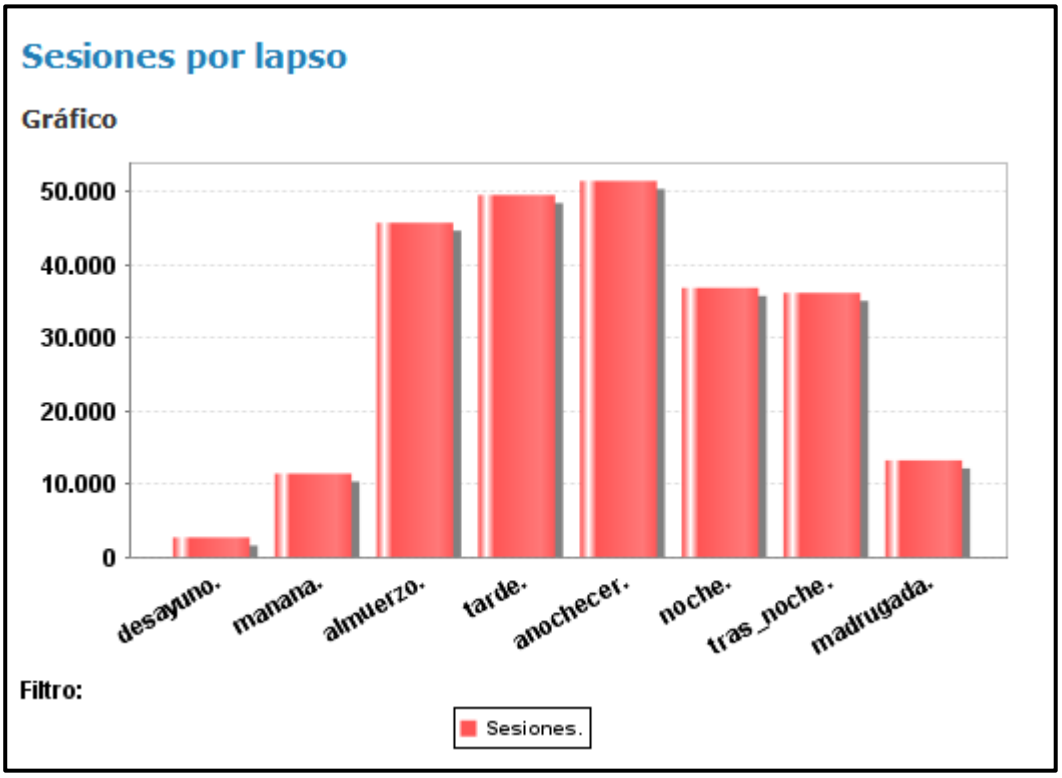

Gráfico 2: Sesiones durante los lapsos del día.

Adicionalmente, a través del cubo "secciones y viñetas", fue posible obtener otros indicadores tradicionales, como por ejemplo, la cantidad de visitas a cada una de las páginas relevantes del estudio, en las dimensiones *Tiempo* y *Fecha.*

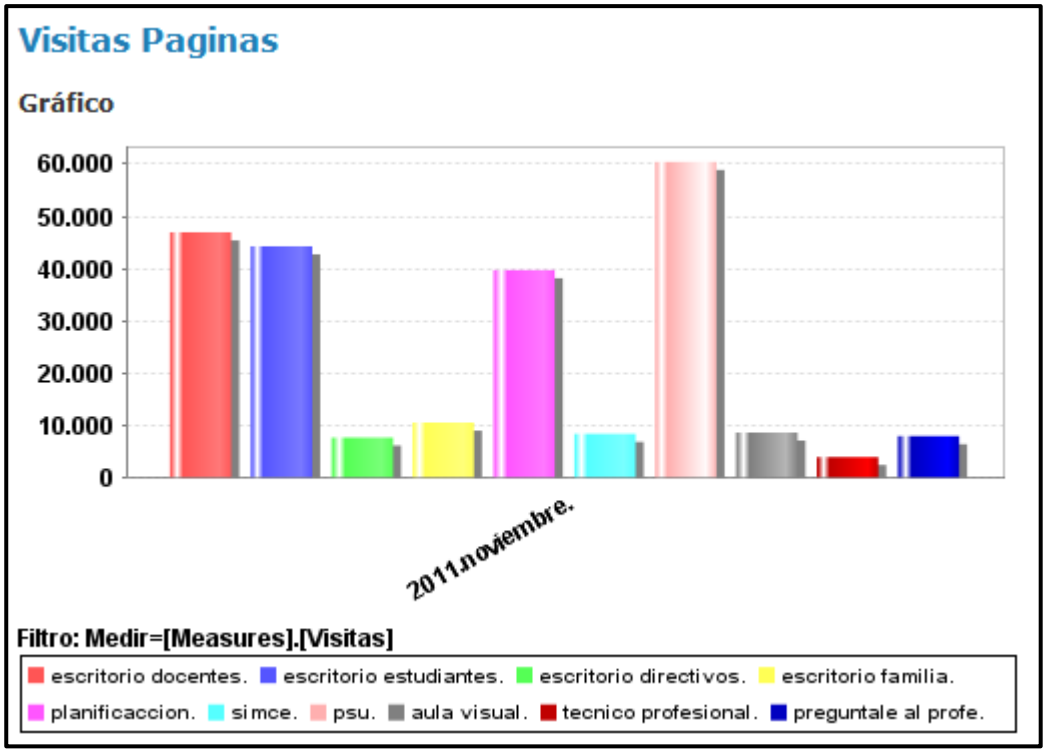

Gráfico 3: Visitas a las páginas relevantes del estudio.

5.2.2. INDICADORES AJUSTADOS A LAS NECESIDADES DE ANÁLISIS DEL PORTAL

El diseño del repositorio de datos incluía una tabla *Fact Peticion*, cuyo grano correspondía a cada una de las peticiones válidas hechas al servidor del portal. En ella se almacenaría información acerca del uso dado a las diferentes secciones y viñetas que poseen las páginas relevantes para el estudio, con el objetivo de obtener indicadores que aún no eran medidos por el equipo del portal a través de las herramientas de pago utilizadas.

A continuación, se presentan los principales resultados obtenidos para este tipo de indicadores.

Secciones de las páginas

Para cada página considerada, se obtuvo la cantidad de click's asociados a las secciones. En la mayor parte de los casos se pudo observar que existe una preferencia hacia aquellas secciones que se encuentran en la parte superior de las páginas. Un ejemplo de esto es el Gráfico 4, donde "destacados" y "te ayudo estudiar" tienen una amplia preferencia por sobre el resto de las secciones, y ambas están situadas en la parte superior de la página.

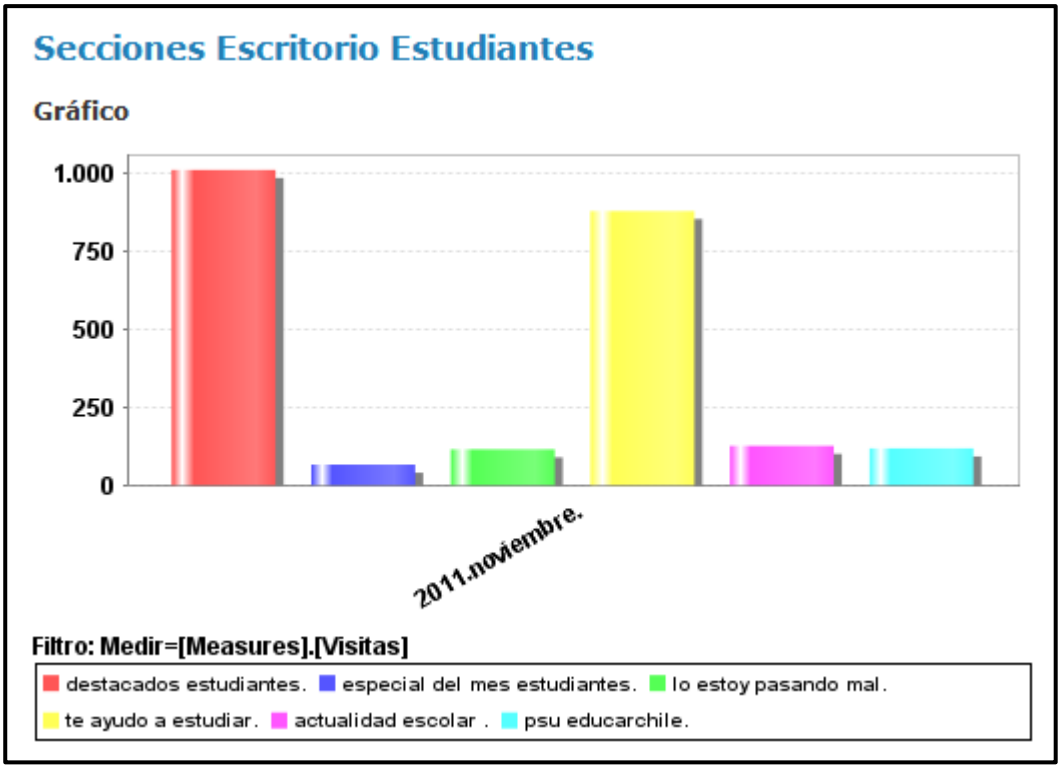

Gráfico 4: Uso de las secciones del escritorio estudiantes.

Otro resultado relevante es la clara preferencia que tienen los usuarios hacia las secciones de "recursos destacados", las que se incluyen en 3 de las páginas estudiadas[28]. Esto se puede explicar tanto por la posición que estas secciones utilizan en sus respectivas páginas, que siempre es la esquina superior izquierda, como también por la cantidad significativa de contenido que poseen, ya que son las únicas secciones que utilizan un recurso del tipo *asp* para la visualización del contenido que se ofrece a los usuarios, lo que les permite exhibir una lista amplia de enlaces que rota constantemente. Un ejemplo de esto se da tanto en el Gráfico 4 como en el Gráfico 5. En este último, incluso se observa que la otra sección ubicada en la parte superior, "en sus aprendizajes", no tiene el mismo nivel de preferencia que "recursos destacados", situación disímil a la que se exhibe en el Gráfico 4.

[28] Los recursos destacados se incluyen en "escritorio docente", "escritorio estudiantes", y en "escritorio familia".

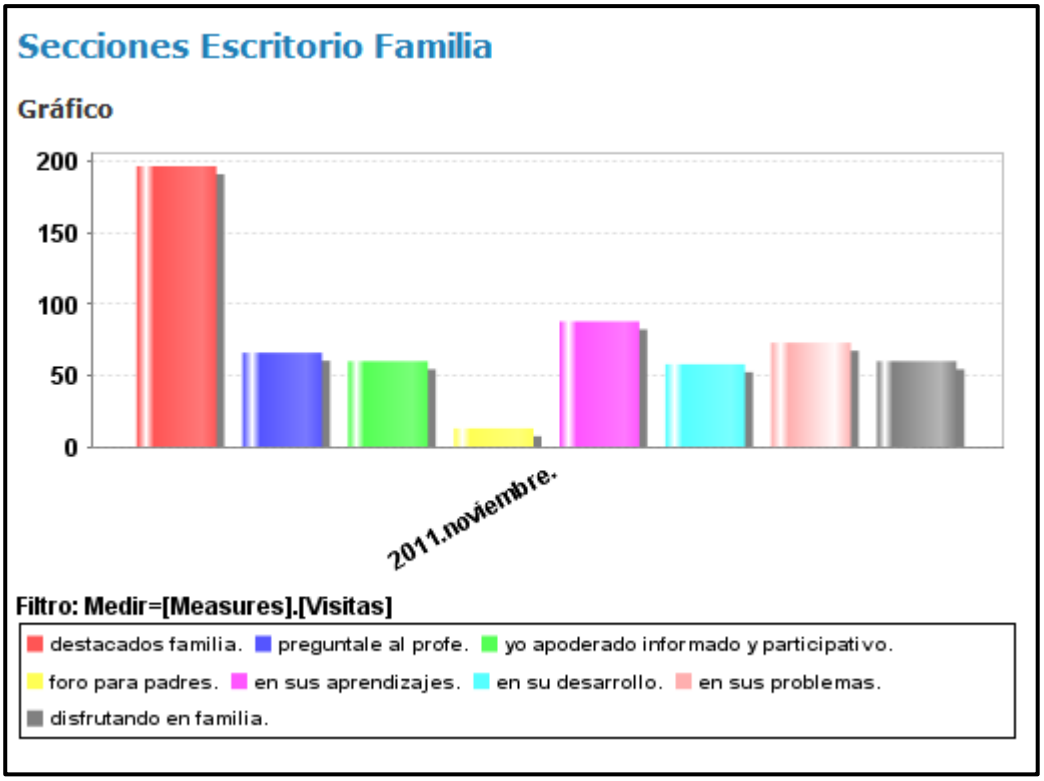

Gráfico 5: Uso de las secciones del escritorio familia

Un tercer resultado importante es el obtenido para la página “PSU”. En primer lugar, tal como se aprecia en el Gráfico 6, esta página es la que obtiene un mayor número de click’s por sobre sus secciones, lo que habla bien del contenido publicado en ella. Esto se explica principalmente a que en la sección “estudia”, los usuarios pueden encontrar material para la preparación de cada una de las 4 pruebas que se rinden para el ingreso a la universidad, separada por nivel de estudios. Así, los usuarios que ingresan a la página logran, en comparación con el resto de las páginas estudiadas, y teniendo en cuenta el número de visitas (que se visualiza en el Gráfico 3), un mayor éxito a la hora de encontrar material útil.

Otro resultado obtenido a través del estudio de las secciones de la página PSU, es la cantidad de visitas que logra “cómo usar psu”, lo que se traduce en que un gran número de usuarios se preocupa de la utilización adecuada de la herramienta. Esto podría deberse a que una cantidad importante de usuarios visitan por primera vez la página, lo que a su vez podría significar que los usuarios no suelen volver a visitar esta herramienta.

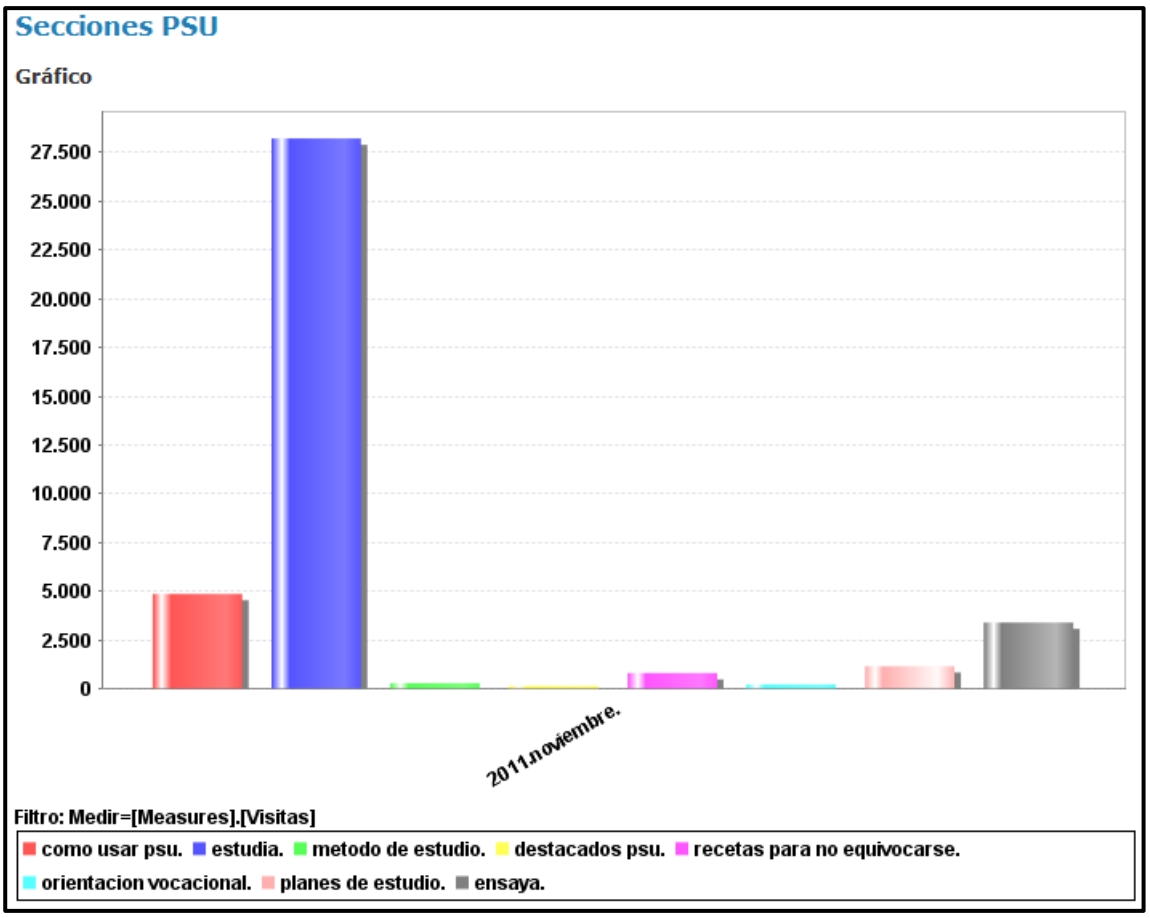

Gráfico 6: Uso de las secciones en PSU.

Un último resultado a destacar es el que se obtiene en la página "Técnico profesional", que se observa en el Gráfico 7. En primer lugar, una de las tres secciones que se encuentra en la parte superior de la página, "establecimientos y empresas", tiene una menor preferencia que las secciones "sectores" y "recursos transversales", que se encuentran en una posición menos privilegiada. Esto indica que el diseño de la página no coindice con el real interés de los usuarios de la sitio. Un segundo resultado, es que tanto estudiantes como docentes visitan en cantidades similares la página, debido a que las secciones enfocadas hacia estos usuarios experimentan una cantidad de click's muy parecida.

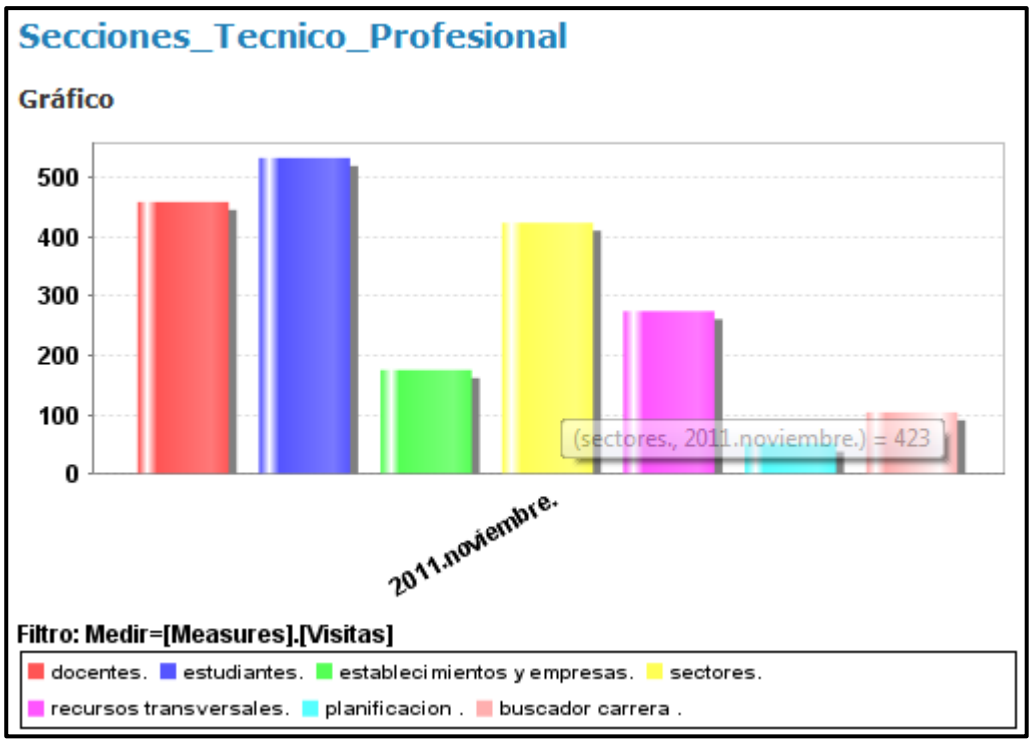

Gráfico 7: Uso de las secciones de Técnico profesional.

Viñetas de las secciones

Para el caso de las secciones que utilizan las llamadas *pestañas* o *viñetas*, se midió el uso que se le daba a este tipo de recursos. La medición de indicadores permitió obtener 3 claros resultados:

1. **Viñetas en secciones informativas o de actualidad**

En los 4 escritorios estudiados se pueden encontrar secciones enfocadas a la publicación de contenido informativo o relacionado con la actualidad del país. En algunas de estas secciones se utilizan las viñetas, para así lograr publicar una mayor cantidad de contenido en un espacio acotado. Los resultados obtenidos mostraron que existe un evidente sesgo de los usuarios hacia el contenido que es publicado en las viñetas activas. Esto se traduce en que el contenido que se encuentra en aquellas pestañas ocultas (que no es visualizado por el usuario a menos que realice un click en la viñeta) es ignorado casi por completo. Esta situación se repitió en cada una de las viñetas este tipo, y un ejemplo es mostrado en el Gráfico 8. Así, los resultados evidencian que la organización no utiliza de manera óptima los recursos que son desarrollados por parte de su equipo de edición de contenido, y por esa razón, este resultado es el más importante del estudio.

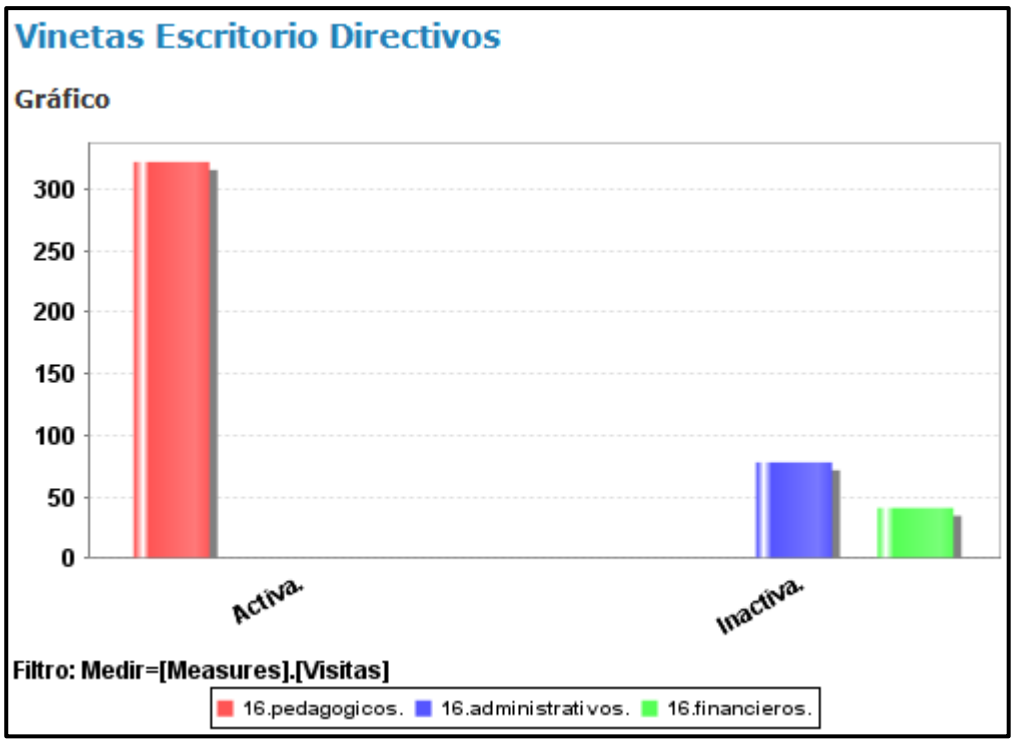

Gráfico 8: Uso de las viñetas en la sección "recursos para la buena dirección" del escritorio directivos.

2. Viñetas en secciones "destacados"

Tal como se mencionó en los resultados de los indicadores que estudian el uso de las secciones de las páginas, existen, en 3 de los 4 escritorios, recursos del tipo *asp* que permiten exhibir contenido a través de listas dinámicas. Una condición adicional a estas secciones es la utilización de viñetas, lo que se traduce en que por cada sección de recursos destacados, existen 2 o 3 pestañas con listas de contenido.

De acuerdo a los resultados obtenidos, se observa que en los 3 casos el comportamiento de los usuarios no es similar, y por lo tanto, sus resultados deben ser analizados individualmente. Por un lado, en el escritorio "docentes" existe una preferencia hacia aquella pestaña que tiene la condición activa. Sin embargo, la diferencia no es mayor a un 100% con respecto al resto de las pestañas. Aun así, el equipo de educarchile se manifiesta conforme con los resultados de este indicador, ya que ellos buscaban potenciar el uso del contenido de la pestaña activa.

Por otro lado, en el escritorio "estudiantes" no existe una preferencia clara hacia la pestaña activa, ya que 2 de las 3 viñetas tienen una cantidad de click's sin diferencias significativas. Sin embargo, en ese mismo resultado se puede apreciar que existe una pestaña que presenta un número muy bajo de visitas (menor a 40 para el mes de noviembre).

Por último, en el escritorio “familia” existe un claro sesgo por parte de los usuarios hacia el contenido exhibido en la pestaña activa, evidenciado en una diferencia de alrededor de un 500%. Así, se puede desprender que este tipo de usuarios no está tan familiarizado con el uso de este tipo de recursos, en comparación a los visitantes de los otros dos escritorios.

Los indicadores que muestran estos resultados pueden ser vistos en los anexos.

3. Viñetas en secciones de contenidos

El tercer caso corresponde a la única sección que se caracteriza por presentar material para el estudio de los estudiantes, “estudia” de la página PSU. En aquella sección se utilizan 4 secciones, una para cada asignatura evaluada en la prueba. Los resultados, mostrados en el Gráfico 9 muestran que existe una preferencia de los usuarios hacia la pestaña activa, y un comportamiento similar para el resto de las viñetas. Sin embargo, la diferencia no supera un 100%, lo que sumado a la importante cantidad de click’s que posee esta sección, se traduce en que los visitantes sí logran acceder al contenido de las 4 asignaturas. Aun así, el sesgo producido por la condición de la pestaña activa indica que debe existir una política clara por parte de la organización hacia la asignatura que se desea potenciar. Una solución ante aquel sesgo es la implementación de un cambio continuo en la elección de la pestaña activa.

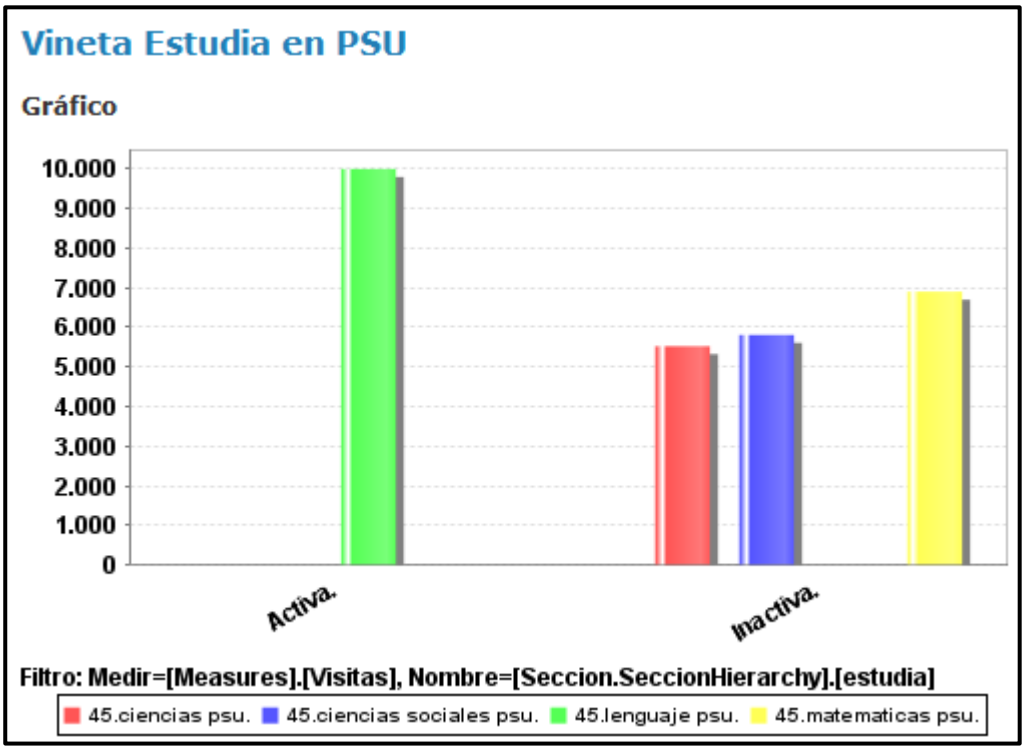

Gráfico 9: Uso de las viñetas en la sección "Estudia" de PSU.

5.3. NAIVE BAYES

El algoritmo de clasificación de documentos se aplicó sobre un total de 839 opiniones pre-clasificadas en las categorías *ayuda*, *planificación* y *resto de opiniones*. Para las dos primeras clases, se utilizaron páginas que fueron previamente identificadas como propicias a poseer comentarios solicitando ayuda o planificaciones. En tanto para la clase *resto de las opiniones,* las páginas consideradas fueron aquellas de la sección de debate del portal.

5.3.1. ENTRENAMIENTO

Para el entrenamiento de los textos, se utilizó un 41% del total de documentos del corpus (346 opiniones), distribuido en:

Clase	Nº de documentos	P(clase)
Ayuda	130	37,5 %
Planificación	42	12,3%
Resto de las opiniones	174	50,2%

Tabla 5: Entrenamiento de las opiniones.

A la vez, el número de palabras diferentes de cada vocabulario obtenido son:

Clase	Palabras diferentes
Ayuda	282
Planificación	142
Resto de las opiniones	529
Vocabulario del corpus	583

Tabla 6: Vocabularios obtenidos.

Para cada vocabulario obtenido a través del modelo, los 4 *tokens* con mayor frecuencia corresponden a:

Clase	*Token*	Frecuencia
Ayuda	Necesito	48
	Preu	39
	Hola	32
	PSU	31
Planificación	Necesito	21
	Comprar	20
	Básica	14
	Hola	11
Resto de las opiniones	Debe	115
	Colegio	114
	Subvención	93
	Lucro	83

Tabla 7: Tokens con mayor frecuencia en cada clase.

Para el caso de la clase *Ayuda*, las palabras *necesito*, *Preu* y *PSU* se explican porque la mayor parte de las opiniones que solicitaban ayuda en el portal, atienden a necesidades de información con respecto a la Prueba de Selección Universitaria. Por otro lado, las palabras obtenidas para la clase *Planificación* responden a que, tal como se identificó como el problema del sitio, muchas veces existen ventas o compras de planificaciones en el portal. Finalmente, en la clase *Resto de las opiniones* se observan palabras relacionadas a la educación. Esto se explica por el set de opiniones que se seleccionó para representar la clase, las que estaban en su mayoría expuestas en la sección de debate del portal, donde el año 2011 se discutió constantemente acerca de temas ligados al *Movimiento Estudiantil* de aquel año.

5.3.2. TEST

La etapa de prueba del modelo obtenido considero un total de 493 opiniones, las que siguieron la siguiente distribución:

Clase	Nº de documentos
Ayuda	75
Planificación	42
Resto de las opiniones	376

Tabla 8: Documentos del set de prueba.

Una vez ejecutada la clasificación de las opiniones, los resultados obtenidos fueron los siguientes:

	Ayuda	Planificación	Resto de las opiniones
Precision	82,278%	28,671%	99,631%
Recall	86,667%	97,619%	71,809%

Tabla 9: Precision and recall de los resultados obtenidos.

Como se observa, las dos clases a las que se tiene por objetivo identificar, que corresponden a *Ayuda* y *Planificación,* tienen un alto índice de *recall*. Esto significa que el clasificador logra reconocer cuando un comentario pertenece a aquellas clases con bastante efectividad. Sin embargo, el indicador *Precision* es bastante pobre para la clase *Planificación*. Esto se explica porque un porcentaje cercano al 28% de los comentarios pertenecientes a la clase *Resto de las opiniones* es clasificada como *Planificación*, y considerando que la distribución de comentarios tiene un mayor peso en la clase *Resto de las opiniones*, resulta un gran porcentaje de falsos positivos, respecto a los verdaderos positivos.

CAPITULO 6 – CONCLUSIONES

La hipótesis del estudio fue validada, ya que se obtuvieron indicadores limpios y consolidados respecto del uso del portal educarchile, con datos de los meses de septiembre y noviembre del año 2011, los que aun no eran medidos con las herramientas que el equipo del sitio posee. Esto les permitirá realizar cambios en el diseño de las páginas más visitadas, de acuerdo a las preferencias de los usuarios del portal.

Se logró saber cuáles eran las preferencias de los usuarios respecto a las secciones que componen cada una de las páginas estudiadas, y además se midió el efecto que producen las viñetas en las secciones que son utilizadas.

Los resultados del estudio permiten evaluar las estrategias de publicación implementadas por la organización, y definir otras nuevas, ya sea en pos de potenciar las actuales preferencias de los usuarios, o con el objetivo de generar mayores visitas en otras menos requeridas.

El sesgo que producen las viñetas en las distintas secciones estudiadas, desde hoy puede ser gestionado de acuerdo a los intereses de la dirección del portal.

El estudio evidenció la mala utilización que se da al tiempo de trabajo del equipo de redacción del sitio, ya que constantemente son publicados artículos informativos en secciones que son ignoradas por los usuarios. Así, los resultados permiten a la organización tomar medidas con el objetivo de evitar dos consecuencias negativas que se producen actualmente:

1. El tiempo de redacción de artículos que es desperdiciado por el equipo del sitio.
2. La no visualización de contenido por parte de los usuarios.

Adicionalmente, dada la importancia que tiene la página PSU en el sitio, respecto al número de visitas, es indispensable manejar el sesgo producido por la utilización de viñetas en la sección de material de estudio. Los resultados indicaron que los usuarios visitan mayormente el material de Lenguaje, dada la condición activa de la viñeta. Sin embargo, esta situación no responde a ninguna política específica de la organización, a pesar de las consecuencias que tiene a la hora de orientar el estudio de los visitantes. Por esa razón, es necesario que se fije la política de publicación de acuerdo a los intereses de la organización.

Aunque los indicadores mostraron tendencias que permiten tomar medidas en pos de mejorar la experiencia de los usuarios, existe información que no siempre es contundente, ya que no es posible identificar con seguridad si los números se deben al diseño o al contenido de las páginas. Así, se abren nuevas posibilidades, ya que a través de nuevos estudios, que tengan en cuenta los resultados obtenidos, es posible conseguir información contundente a la hora de tomar medidas.

En cuanto al rendimiento del *Webhouse*, la utilización del gestor de bases de datos relacionales *MySql* permitió almacenar información y extraer conocimiento a través del set de módulos que componen *Jasper*, con la ventaja de ser ambas herramientas gratuitas. Sin embargo, el notable decaimiento del rendimiento una vez que las tablas superaron el millón de datos, indica que se debe pensar en otras alternativas para proyectos que incluyan un período de estudio mayor.

Por su lado, el modelo de datos utilizado permitió acceder a cada uno de los indicadores que se quiere medir. Además, la inclusión de la tabla “Opinión”, que no tenía como objetivo alimentar consultas OLAP, mostró que el diseño del *Webhouse* es lo suficientemente versátil como para acomodarse a distintos requisitos de procesamiento de datos.

Respecto a los resultados obtenidos con el filtro de opiniones, a pesar del tamaño reducido del corpus que fue utilizado, la implementación del algoritmo *Naive Bayes* resultó ser bastante efectiva en relación al problema planteado, dado que es posible identificar con un porcentaje mayor al 85% las opiniones que solicitan ayuda, o que se refieren a las planificaciones de docentes. Así, la implementación en el sitio del modelo permitiría a la organización evitar comentarios que no se ajustan a su política de contenidos a través de esta tecnología.

Como mejoras al trabajo realizado se consideran:

1. Considerar todo el sitio en los análisis, quitando el segundo filtro de los archivos *weblogs.* Esto último permitirá acceder aproximadamente al 60% del total de sesiones que no visitan alguna de las páginas relevantes de este estudio, enriqueciendo el análisis.

2. Incluir datos operacionales del portal dentro del estudio. Específicamente, considerar información del registro de los usuarios, lo que permitiría potenciar el proceso de sesionización. Además, acceder a las tablas que registran los cambios que se realizan diariamente en el sitio evitaría la necesidad de *crawlear* a diario el portal, ya que limita las posibilidades de expansión del modelo.

3. Considerar otros algoritmos de minería de datos para evaluar los filtros de opiniones, y así contrastar los resultados obtenidos a través de *Naive Bayes.*

Por otro lado, como posible trabajo a futuro junto al equipo de educarchile se encuentran:

1. Inclusión del *Home* dentro del análisis de preferencias de los usuarios, debido a que es una de las pocas páginas del portal que no está enfocada a un tipo de usuario específico, lo que contrasta con los escritorios.

2. Medición de los indicadores, incluyendo cambios en las posiciones de las secciones dentro de las páginas, lo que permitiría evaluar en qué medida se explican las preferencias de los usuarios a partir de la posición o contenido de las secciones.

3. Elaborar un análisis de las preferencias a través de técnicas de minería de datos. Esto permitiría conocer, entre otras cosas, qué tópicos resultan más relevantes para los usuarios, lo que complementaría el estudio del diseño de las páginas. Así, el equipo de redacción de contenido podría enfocar sus esfuerzos en elaborar artículos que generen un mayor número de visitas.

4. Diseñar y construir una plataforma web que realice automáticamente todo el proceso ETL, y que permita a los usuarios acceder a los indicadores con información reciente. Además, implementar la utilización de la versión de pago de *Jasper Server*, lo que permitiría a los usuarios ampliar las posibilidades de análisis.

5. Implementar un módulo que evalúe en tiempo real cada una de las opiniones vertidas en el portal, lo que se hará imprescindible una vez lanzada la nueva versión del sitio, ya que potenciará la colaboración de los usuarios en la publicación de contenido.

.

CAPÍTULO 7 – BIBLIOGRAFÍA

[1] AARSTEN, A.; D. BRUGALI y G.MENGA. 1996. Patterns for Three-Tier Client/Server Applications.

[2] ANDROUTSOPOULOS, I.; J. KOUTSIAS; K. CHANDRINOS y C. SPYROPOULOS. 2000. An experimental comparison of naive Bayesian and keyword-based anti-spam filtering with personal e-mail messages. ACM.

[3] ARLITT, M. F. y C. L. WILLIAMSON. 1996. Web server workload characterization: the search for invariants. ACM SIGMETRICS, Volume 24 Issue 1. New York, USA.

[4] BALLARD, C.; D. HERREMAN; D. SCHAU; R. BELL; E. KIM, y A. VALENCIC. 1998. Data Modeling Techniques for Data Warehousing. IBM.

[5] BERNERS-LEE, T., C. CAILLIAU, J.F. GROFF, Y B. POLLERMANN, 1992. World-Wide We: The information universe. Communications of the ACM.

[6] BONTEMPO, C. y G. ZAGELOW. 1998. The IBM Data Warehouse Architecture. Communications of the ACM, Volume 41. New York, USA.

[7] CHAUDHURI, S. y M. DAYAL. 1997. An Overview of Data Warehousing and OLAP Technology. Newsletter ACM SIGMOD Record, New York, USA.

[8] CODD, E.F. 1970. A relational model of data for large shared data banks. Magazine Communications of the ACM, Volume 13 Issue 6. New York, USA.

[9] COOLEY, R.; B. MOBASHER y J. SRIVASTAVA. 1997. Web Mining: Information and Pattern Discovery in the Word Wide Web. Proceedings Ninth IEEE International Conference on Tools with Artificial Intelligence.

[10] DUJOVNE, L. Y J.D. VELÁSQUEZ. 2009. Design and Implementation of a Methodology for Identifying Website Key objects. Universidad de Chile, Chile.

[11] FELDMAN, R. y J. SANGER. 2007. The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data. Cambridge University Press.

[12] GREENSPAN, J. y B. BULGER, 2001. MySQL/PHP Database Applications. M&T Books.

[13] HAN, J. y M. KAMBER. 2006. Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers.

[14] HARB, A.; M. PLANTIÉ; G. DRAY; M. ROCHE; F. TROUSSET, y P. PONCELET. 2008. Web opinion mining: how to extract opinions from blogs? CSTST '08 Proceedings of the 5th international conference on Soft computing as transdisciplinary science and technology.

[15] HOWLAND, P. y H. PARK. 2007. Cluster-Preserving Dimension Reduction Methods for Document Classification. Springer.

[16] HU, X. y N. CERCONE. 2004. A data warehouse/online analytic processing framework for web usage mining and business intelligence reporting. Journal International Journal of Intelligent Systems - Granular Computing and Data Mining, Volume 19 Issue 7. New York, USA.

[17] KAMBALYAL, C. 3-Tier Architecture.
[18] KIMBALL, M. y M. ROSS, 2002. The Data Warehouse Toolkit. Segunda Edición. John Wiley y Sons.
[19] KOSALA, R. y H. BLOCKEEL. 2000. Web Mining Research: A Survey. ACM.
[20] KOWALSKI, G. y M. MAYBURY. 2002. Information Storage and Retrieval Systems: Theory and Implementation. Second Edition. Kluwer Academic Publishers.
[21] LEVENE, M. y G. LOIZOU. 2003. Why is the snowflake schema a good data warehouse design? School of Computer Science and Information Systems, Birkbeck College, University of London.
[22] MARKOV, Z. y D. T. LAROSE. 2007. Data Mining the Web: Uncovering patterns in Web content, structure and Usage. John Wiley & Sons.
[23] MCFADDEN, F.R. 1996. Data warehouse for EIS: some issues and impacts. Wailea, HI, USA.
[24] MOODY, D. 2000. From Enterprise Models to Dimensional Models: A Methodology for Data Warehouse and Data Mart Design.
[25] NIGAM, K.; A. MCCALLUM; S. THRUN y T. MITCHEL. 1999. Text Classification from Labeled and Unlabeled Documents using EM. Journal Machine Learning - Special issue on information retrieval.
[26] LIU, B. 2008. Opinion Mining and Summarization – Sentiment Analysis. University of Illinois. Chigaco, USA.
[27] PAL. S. K. TALWAR, V. y P. MITRA, 2002. Web Mining in soft computing framework: relevance, state of the art and future directions. IEEE transactions on neural networks.
[28] QUIRK, R.; S. GREENBAUM; G. LEECH, y J. SVARTVIK. 1985. A Comprehensive Grammar of the English Language. Longman.
[29] REBOLLEDO, V. 2008. Plataforma para la extracción y almacenamiento del conocimiento extraído de los web data. Tesis, Universidad de Chile, Chile.
[30] SANNER, N. 1999. Python: a programming language for software integration and development. The Scripps Research Institute.
[31] SCHNEIDER, K. 2003. A Comparison of Event Models for Naive Bayes Anti Spam E-mail Filtering. EACL '03 Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics - Volume 1.
[32] SINGHAL, A. y G. SALTON. 1995. Automatic Text Browsing Using Vector Space Model. Proceedings of the Dual-Use Technologies and Applications Conference.
[33] SPILIOPOULOU, M.; B. MOBASHER; B. BERENDT, y M. NAKAGAWA, 2003. A Framework for the Evaluation of Session Reconstruction Heuristics in Web-Usage Analysis. INFORMS Journal on Computing.
[34] SWEIGER, M.; M. MADSEN; J. LANGSTON y H. LOMBARD. 2002. Clickstream Data Warehousing. John Wiley y Sons.
[35] TEIXEIRA, C. y G. DAVID. 2006. Higher Education Web Information System Usage Analysis with a Data Webhouse. Computational Science and Its Applications – ICCSA.

[36] VELASQUEZ, J. D.; R. WEBER; H. YASUDA y T. AOKI. 2005. Acquisition and maintenance of knowledge for web site online navigation suggestions. IEICE Transactions on Information and Systems.
[37] VELÁSQUEZ, J.D. y V. PALADE. 2008. Adaptative Web site: a knowledge extraction form Web data approach. IOS Press, Netherlands.
[38] WEISS, S.; N. INDURKHYA; T. ZHANG y F. DAMERAU. 2005. Text Mining :Predictive Methods for Analyzing Unstructured Information. Springer.
[39] ZDZIARSKI, J. 2005. Ending Spam: Bayesian Content Filtering and the Art of Statistical Language Classification. No Starch Press.

ANEXOS

A. INDICADORES DE USO DEL PORTAL

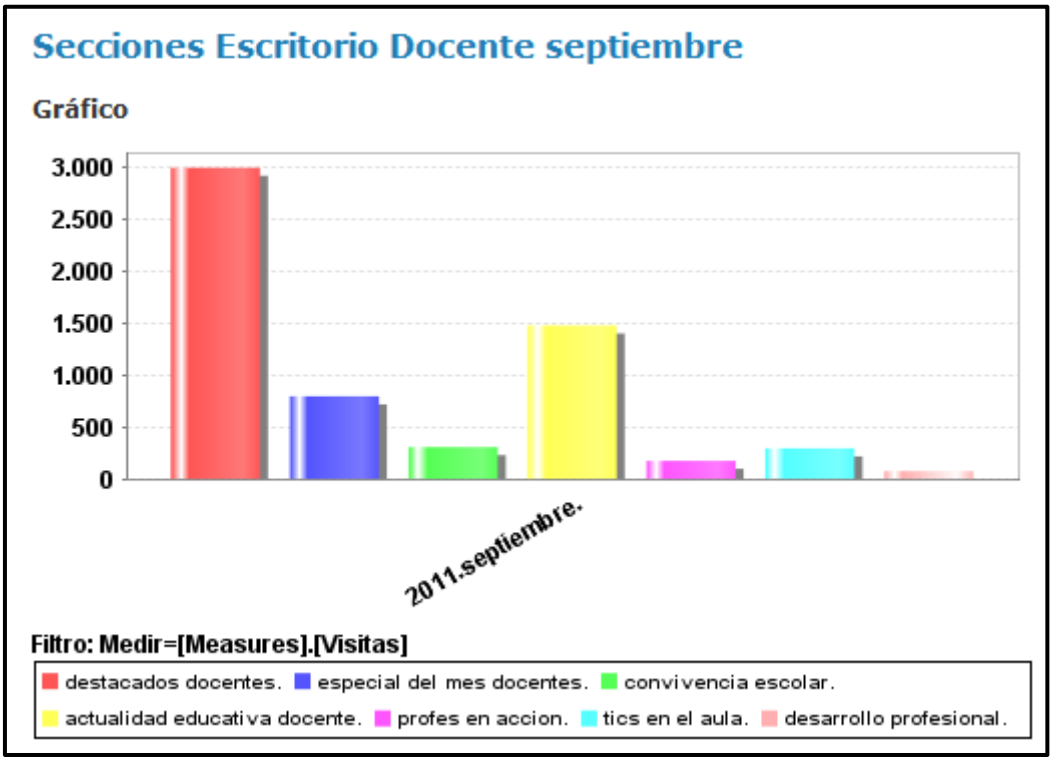

Gráfico 10: Uso de las secciones del escritorio docentes. Septiembre.

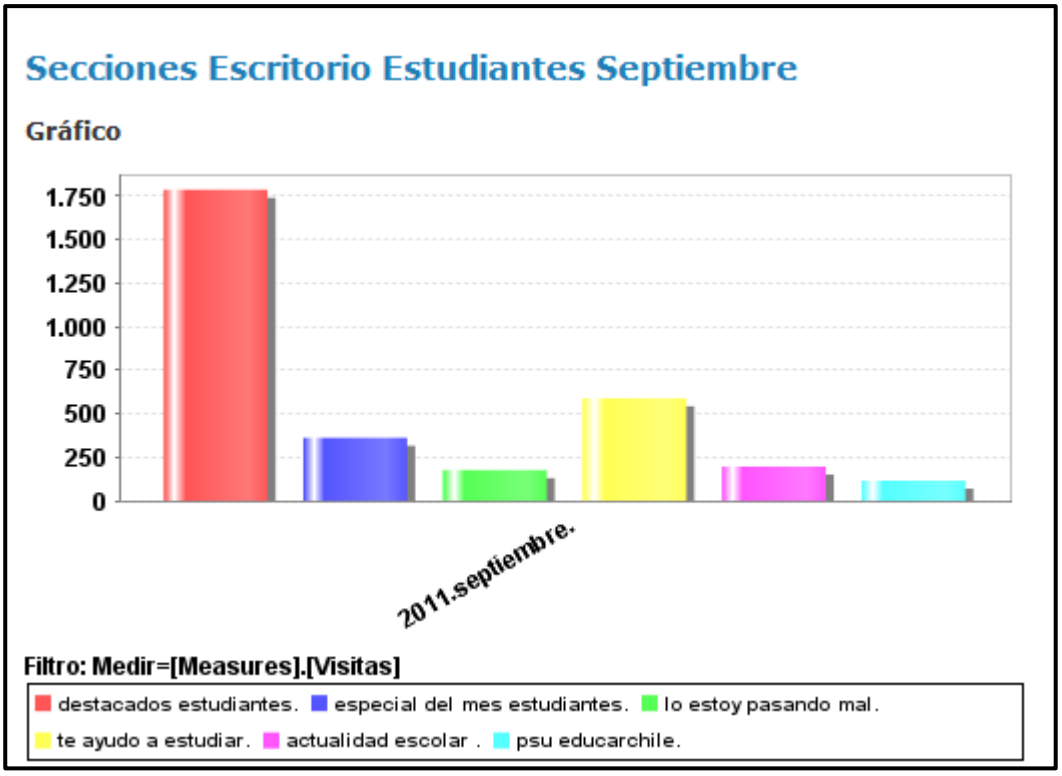

Gráfico 11: Uso de las secciones del escritorio estudiantes. Septiembre

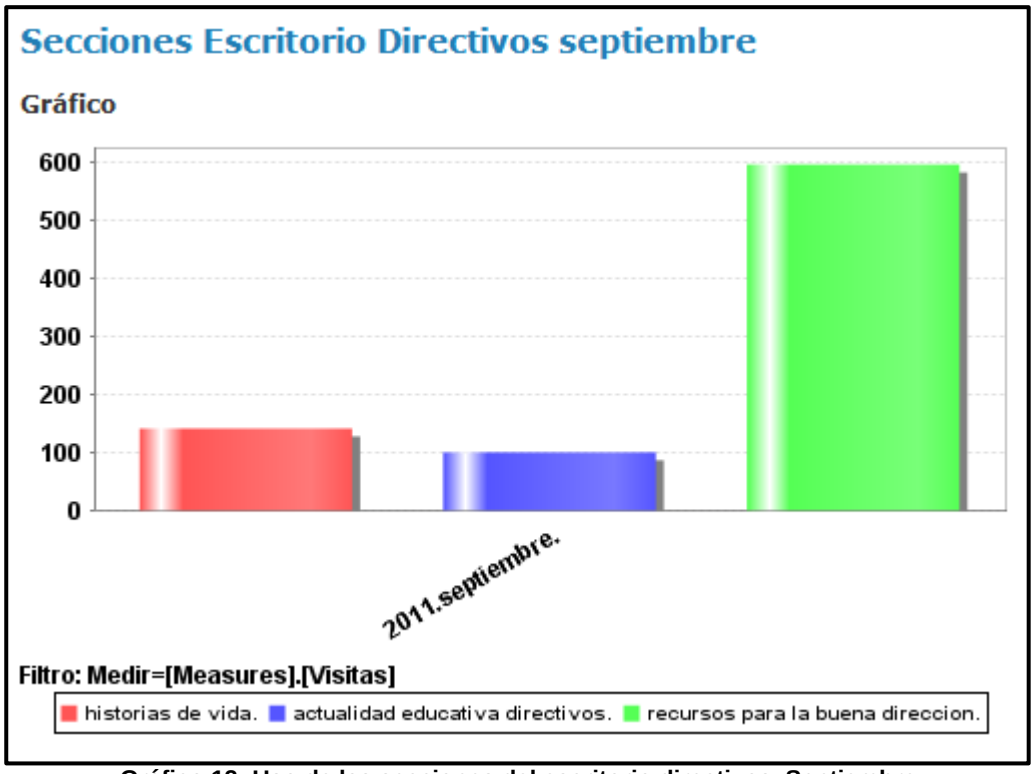

Gráfico 12: Uso de las secciones del escritorio directivos. Septiembre

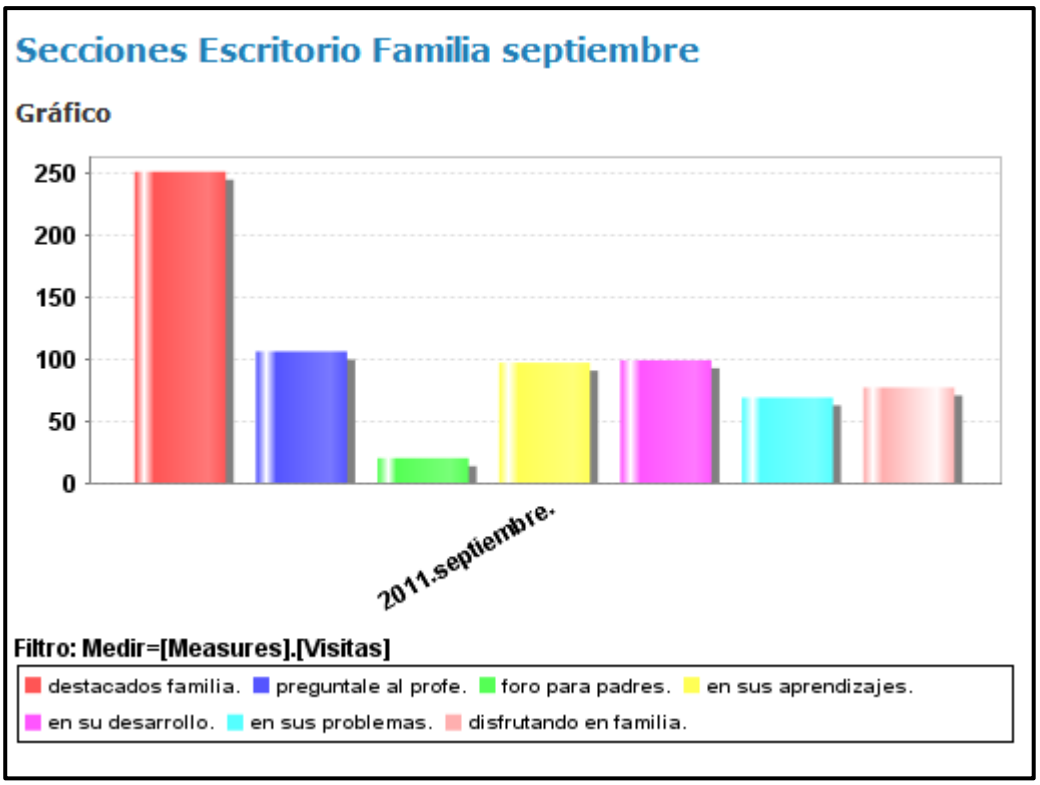

Gráfico 13: Uso de las secciones del escritorio familia. Septiembre

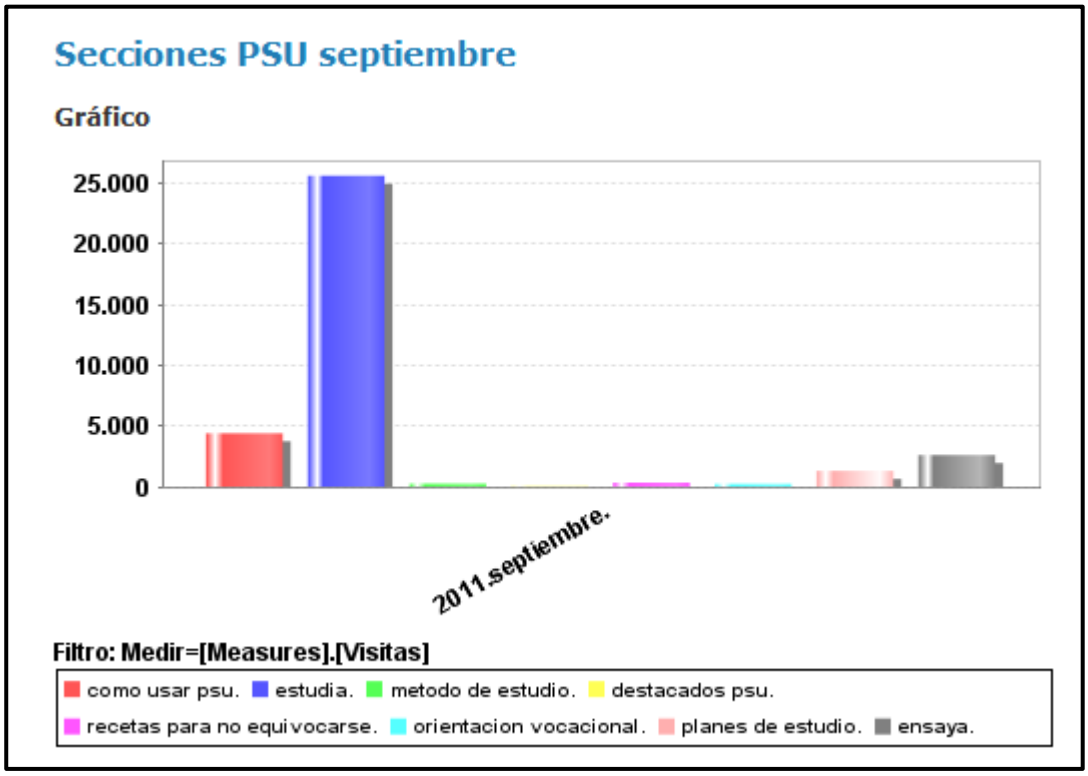

Gráfico 14: Uso de las secciones de la página PSU. Septiembre.

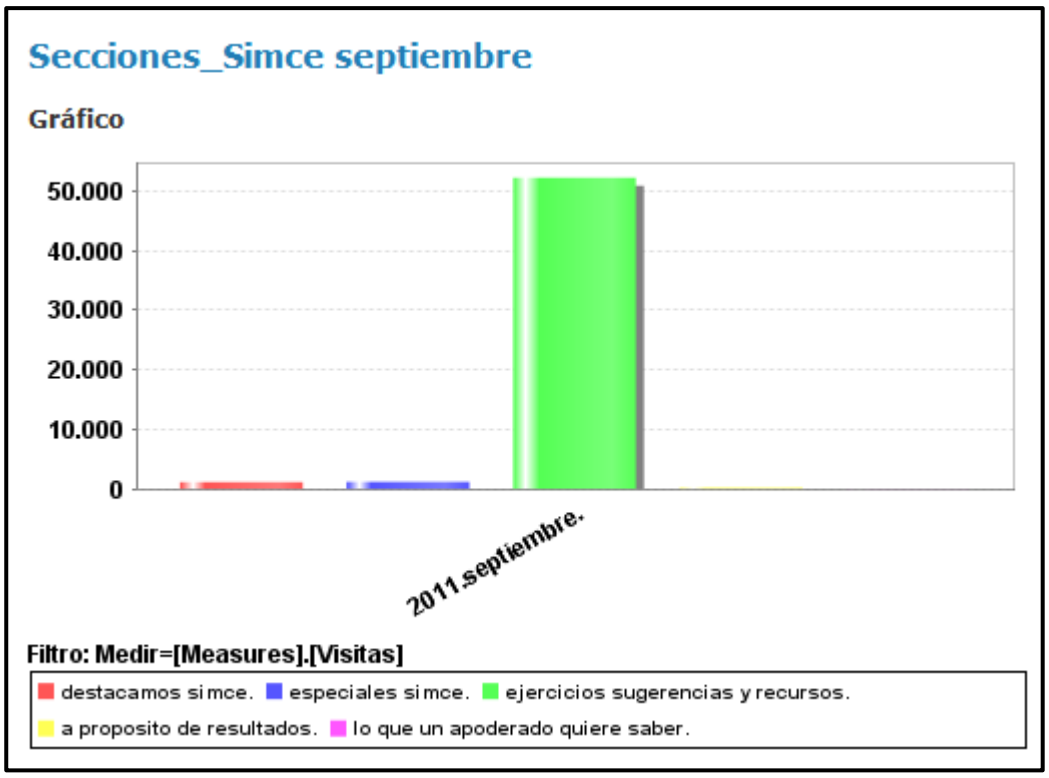

Gráfico 15: Uso de secciones de la página Simce. Septiembre.

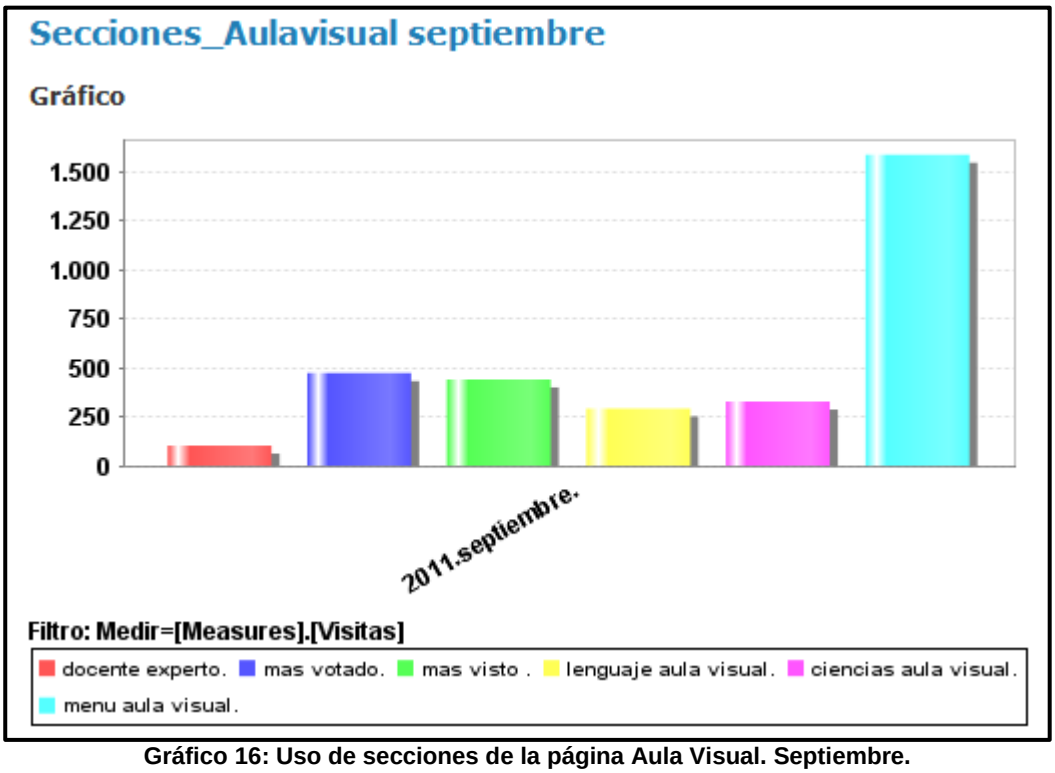

Gráfico 16: Uso de secciones de la página Aula Visual. Septiembre.

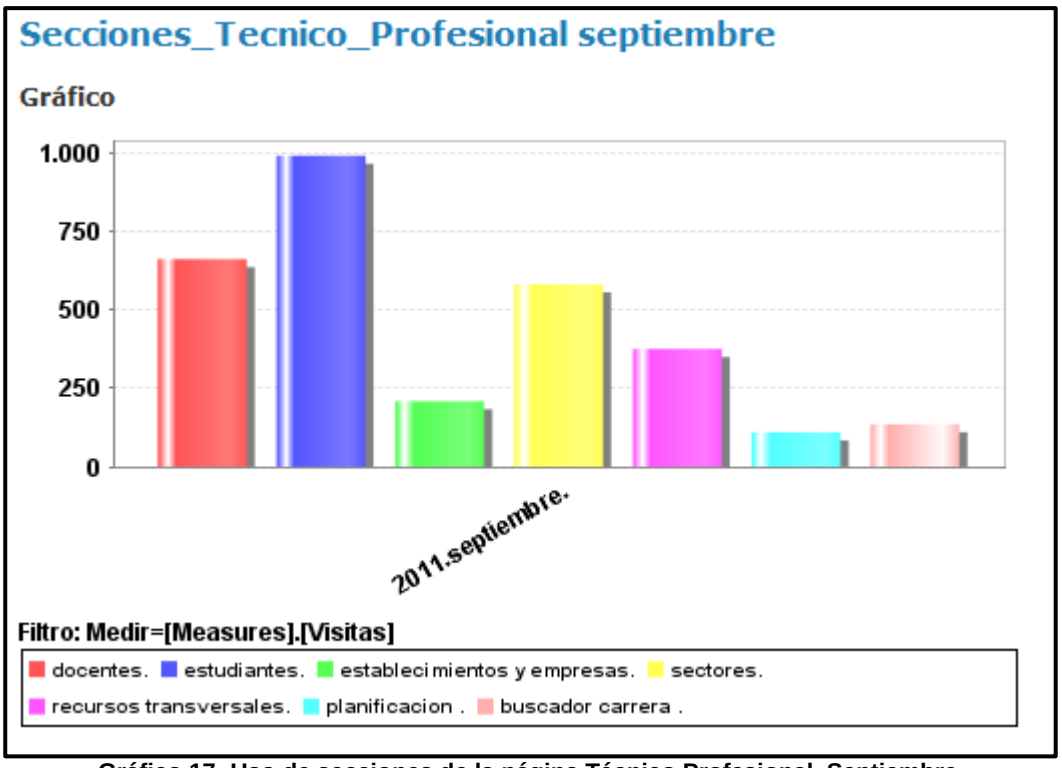

Gráfico 17: Uso de secciones de la página Técnico Profesional. Septiembre.

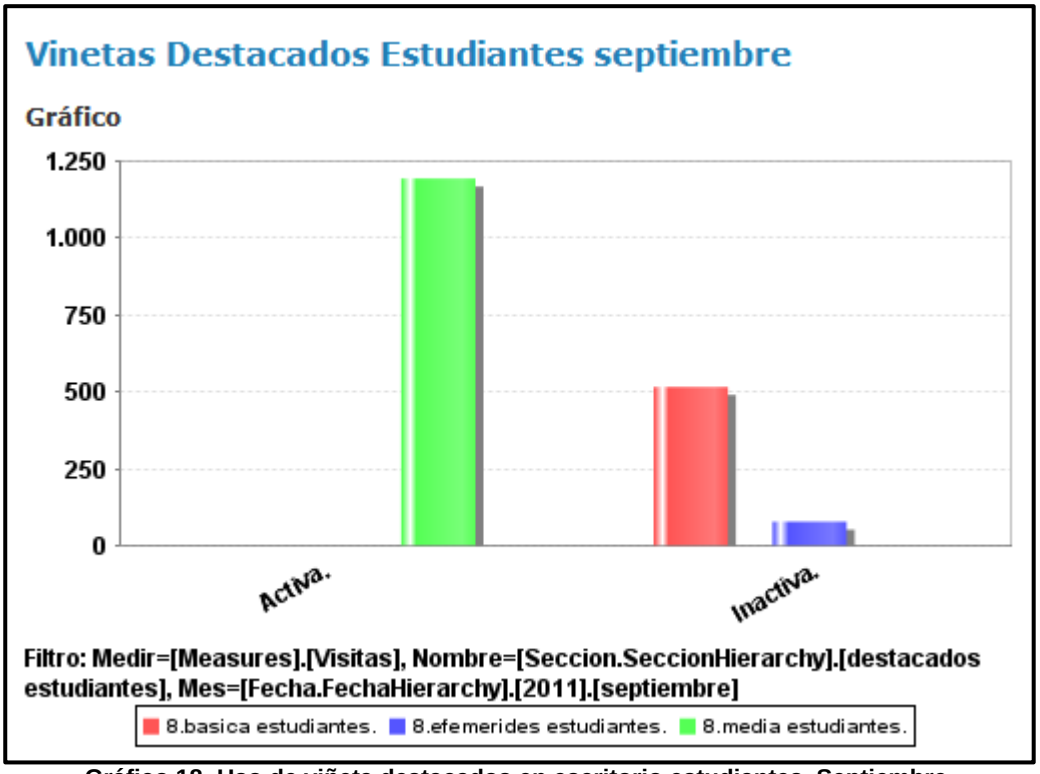

Gráfico 18: Uso de viñeta destacados en escritorio estudiantes. Septiembre.

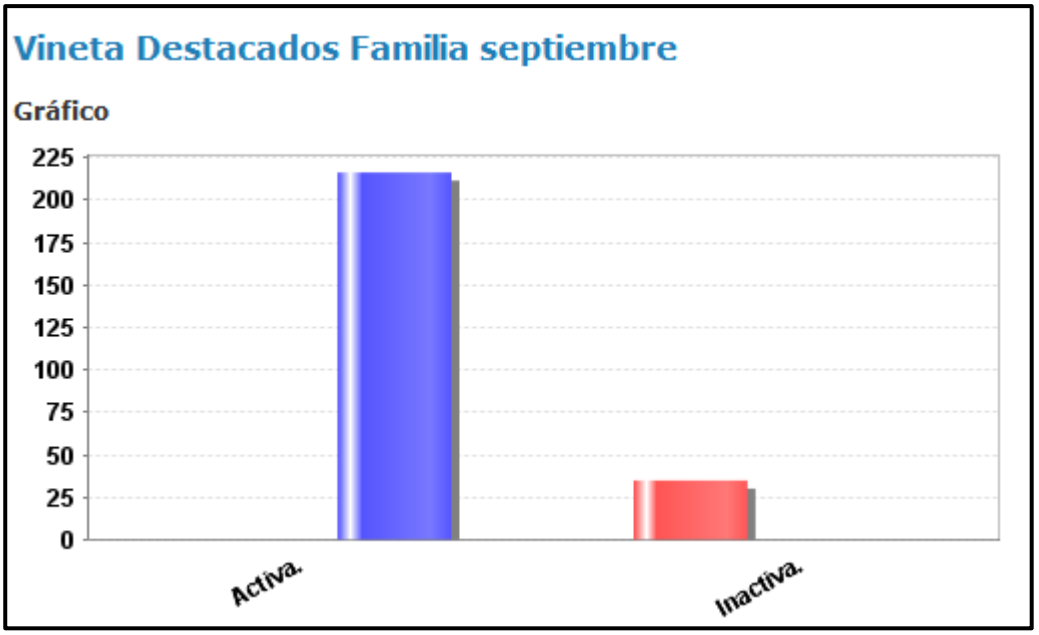

Gráfico 19: Uso de viñeta destacados en escritorio familia. Septiembre.

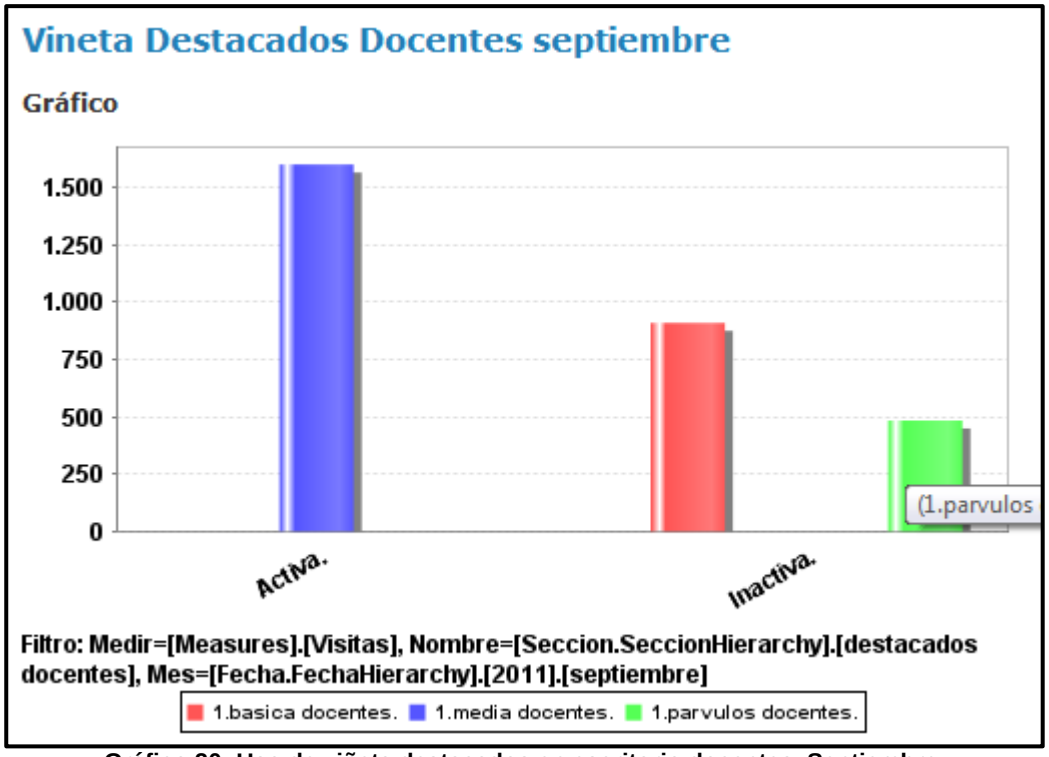

Gráfico 20: Uso de viñeta destacados en escritorio docentes. Septiembre.

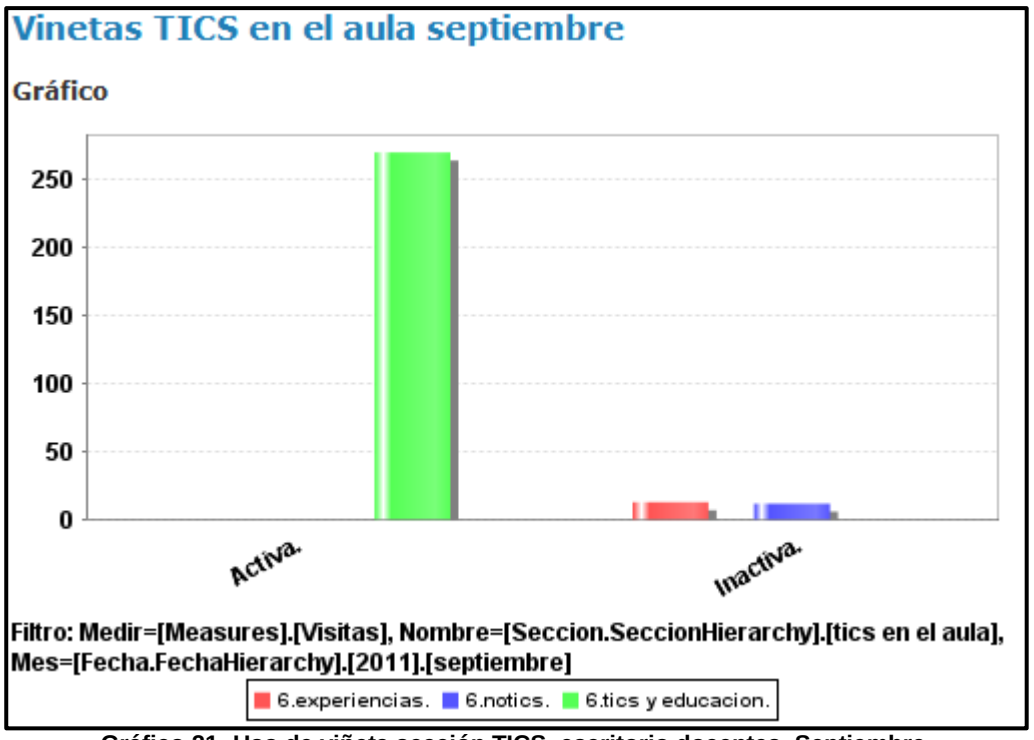

Gráfico 21: Uso de viñeta sección TICS, escritorio docentes. Septiembre.

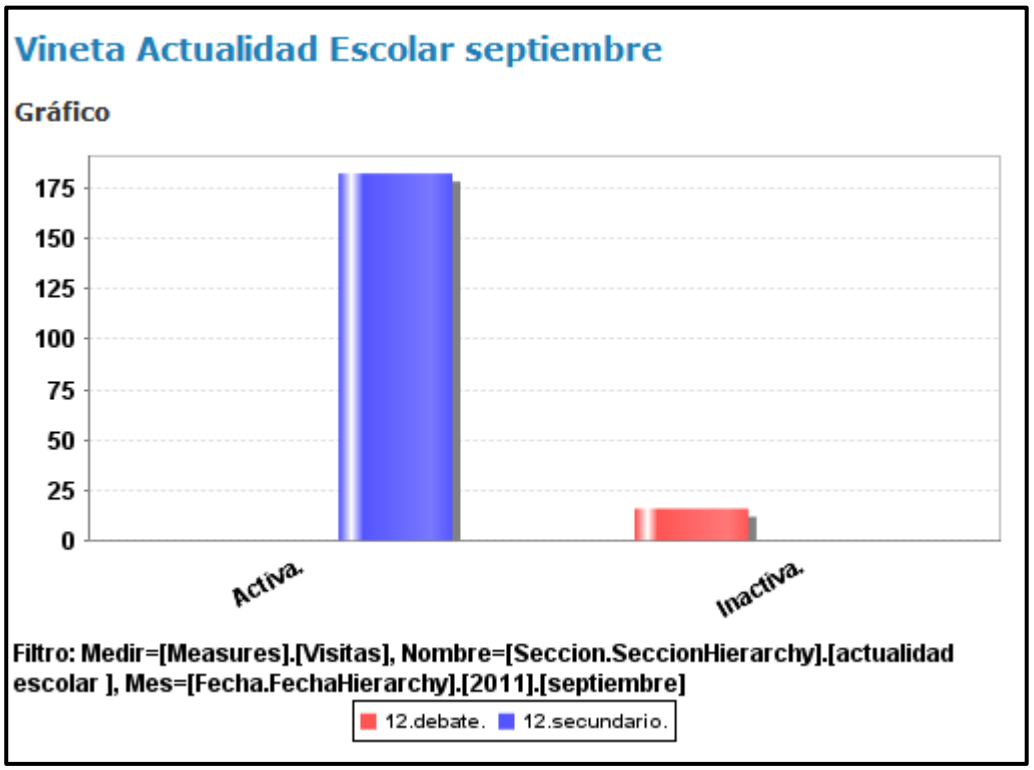

Gráfico 22: Uso de viñeta sección actualidad escolar, escritorio docente. Septiembre.

B. CODIGOS UTILIZADOS

B.1. SESIONIZACION

```
import time
import MySQLdb
import winsound

def resta(tiempo1, tiempo2): #funcion que calcula el tiempo transcurrido.    #se
pasan a strings
    tiempo1=str(tiempo1)
    tiempo2=str(tiempo2)
    #se extraen los tiempos de cada uno
    time1=tiempo1.split(":")
    time2=tiempo2.split(":")

    #se pasa cada hora a segundos
    time1=int(time1[0])*60*60+int(time1[1])*60+int(time1[2])
    time2=int(time2[0])*60*60+int(time2[1])*60+int(time2[2])

    #se retorna la diferencia
    return time2-time1

def sesion(dia, carpeta, sesion):
    timeStart= time.localtime()
    #abrir archivo para escribir sql
    nombresql=carpeta+"sesiones_"+dia+".sql"
    f = open(nombresql, "w")
    #tiempo maximo en segundos
    max_session=30*60
    #parametros de la base de datos
    host1='localhost'
    user1='root'
    passwd1=''
    db1="dsatest"
    #conexion a la base de datos
    db=MySQLdb.connect(host=host1,user=user1,passwd=passwd1,db=db1)
    cursor=db.cursor()
    #tablas a utilizar
    tabla1='weblog'
    tabla2='sesion'
    #1 weblog
    idW='id'
    ipW='ip'
    urlW='url'
    agenteW='agent'
    horaW='hora'
    fechaW='fecha'
    #sesion
    tiempoS='tiempo'
    idS='id'
    ordS='ord'
    id_registroS='id_registro'
    id_paginaS='id_pagina'
    id_ipS='id_ip'
    id_sesionS='id_sesion'
    sizeS="size"
```

```
#primero se seleccionan todas las combinaciones distintas ip-agente en la tabla
weblog
    sql="select distinct "+ ipW+", "+agenteW+" from "+tabla1+" where
"+fechaW+"='"+str(dia)+"'"
    cursor.execute(sql)
    result1=cursor.fetchall()#cada fila del resultado tiene un arreglo de dos
elementos
    #variables para las id's
    ip=0
    agente=0
    id_sesion=int(sesion)+1
    upd=""
    indupd=0
    ins="insert into "+tabla2+" ("+id_sesionS+", "+id_registroS+", "+ordS+",
"+id_paginaS+", "+id_ipS+", "+tiempoS+") values"
    insTT=""
    indins=0
    siz=[]
    indsiz=0
    upd=[]
    #ahora se recorren todas las combinaciones
    for comb in result1:
        #extraigo las ids
        ip=comb[0]
        agente=comb[1]
        #ahora se extraen todas las entradas del dia que cumplen con esa
combinacion
        sql="select "+idW+", "+urlW+", "+horaW+" from "+tabla1+" where
"+ipW+"="+str(ip)+" AND "+agenteW+"="+agente+" AND "+fechaW+"='"+dia+"'"
        cursor.execute(sql)
        result2=cursor.fetchall() #entrega todas las entradas que cumplen con la
combinacion
        first_time=result2[0][2] #se empieza una nueva sesion
        last_time=result2[0][2] #el tiempo del ultimo resgitro
        ord=0 #la posicion del registro
        i=1
        for registro in result2: #registros de la combinacion
            tiempo=resta(first_time, registro[2])
            if tiempo>max_session: #se paso, entonces generar nueva sesion
                first_time=registro[2]
                last_time=registro[2]
                ord=0
                id_sesion=id_sesion+1
            stay_time=resta(last_time, registro[2])

            if(ord>0): #estamos en la mitad de una sesion --> se updatea la db
con el tiempo del registro anterior
                sql="update "+tabla2+" set "+tiempoS+"="+str(stay_time)+" where
"+id_sesionS+"="+str(id_sesion)+" AND "+ordS+"="+str(ord)
                upd.append(sql)

            ord=ord+1#se avanza en un orden siempre en cada registro
            last_time=registro[2]# se actualiza el tiempo del ultimo que se
ocupara para el proximo
            if indins==0:
                insTT=ins+" ("+str(id_sesion)+", "+str(registro[0])+",
"+str(ord)+", "+str(registro[1])+", "+str(ip)+", 0)"
            else:
                insTT=insTT+", ("+str(id_sesion)+", "+str(registro[0])+",
"+str(ord)+", "+str(registro[1])+", "+str(ip)+", 0)"
            indins=indins+1
```

```
            if indins==100:
                f.write(insTT+";\n")
                indins=0
                for re in upd:
                    f.write(re+";\n")
                upd=[]
        id_sesion=id_sesion+1
    #insertamos los que quedaron sueltos
    if indins>0:
        f.write(insTT+";\n")
    for re in upd:
        f.write(re+";\n")

    f.close()
```

B.2. NAIVE BAYES

```
class BayesText:

    def __init__(self, trainingdir):

        self.vocabulary = {} #vocabulario entero del modelo
        self.prob = {} # los diccionarios para cada clase
        self.probaFi={}
        self.totals = {} #el total de palabras para cada clase
        #self.stopwords = {} #los stopwords
        self.archivos={} #el numero de archivos en cada clase
        self.archivosT=0.0
        self.factorProbC={}
        self.probC={}
        self.sumatoriaVoc={}
        self.vocabularios={}

        categories = os.listdir(trainingdir) #las clases son la lista de
carpetas en el dir

        self.categories = [filename for filename in categories if
os.path.isdir(trainingdir + filename)]
        print("Counting ...")
        for category in self.categories:
            print('    ' + category)
            (self.prob[category], self.vocabularios[category],
self.totals[category]) = self.train(trainingdir, category) #el primero es la
probabilidad, el segundo es el total

        toDelete = []
        for word in self.vocabulary: #elimina todas las palabras que aparecen
menos de 3 veces en total.
            if self.vocabulary[word] < 3:
                    toDelete.append(word)
        for word in toDelete:
            del self.vocabulary[word] #aca las borra
            for category in self.categories:
                if word in self.vocabularios[category]:
                    del self.vocabularios[category][word]
         print "Tokens con mayor peso"
        for category in self.categories:
            palabras=self.vocabularios[category].items()
            palabras.sort(key=lambda tuple: tuple[1], reverse = True)
            print palabras
       vocabLength = len(self.vocabulary) #el largo del vocabulario completo
         print "Corpus", vocabLength
         print("Computing probabilities:")
         for category in self.categories:
            print('    ' + category)
            denominator = self.totals[category] + vocabLength #palabras de la
clase+largovocab
            self.probaFi[category]=self.prob[category]
            for word in self.vocabulary: #todas las palabras del modelo
                if word in self.prob[category]: #si esta en la clase
                    count = self.prob[category][word] #la cantidad de veces
que aparece en la clase
                else:
                    count = 1 #si no esta, es 1
                self.probaFi[category][word] = (count + 1.0) / denominator
#actualiza la cantidad por esta division
```

```
#para calcular el total de documentos
        print "--- --- ---"
        for category in self.categories:
            self.archivosT=self.archivosT+self.archivos[category]
            print category, self.archivos[category]
        print "total de documentos", self.archivosT
        #calcular las probabilidades de cada clase
        print "--- --- ---"
        print "Probabilidades de cada clase"
        for category in self.categories:
            self.probC[category]=self.archivos[category]/self.archivosT
            print category, self.probC[category]
        #calcular los factores de probabilidades de cada clase (P(c)/P(c)
        print "--- --- ---"
        print "Factores de probabilidades"
        factoresAux={}
        for category in self.categories:
            for category2 in self.categories:

factoresAux[category2]=self.probC[category2]/self.probC[category]
            self.factorProbC[category]=factoresAux
            print category, self.factorProbC[category]
        #calcular la sumatorio de los N de cada clase para el vocabulario
        print "--- --- ---"
        print "sumatorias"
        for category in self.categories:
            self.sumatoriaVoc[category]=0.0
            for word in self.vocabulary:
                c=0.0
                if word in self.vocabularios[category]:
                    c=self.vocabularios[category][word]
                    self.sumatoriaVoc[category]=self.sumatoriaVoc[category]+c
            print category, self.sumatoriaVoc[category]
        #se le suma el largo del vocabulario a las sumatorias
        print "---- ---- ----"
        print "sumatorias mas N del voc"
        for category in self.categories:

self.sumatoriaVoc[category]=len(self.vocabulary)+self.sumatoriaVoc[category]
            print category, self.sumatoriaVoc[category]

        print "---- ---- ----"
```

```
def train(self, trainingdir, category):
        """counts word occurrences for a particular category"""
        currentdir = trainingdir + category #el dir de la clase
        files = os.listdir(currentdir) #todos los archivos
        counts = {} #el diccionario de la clase
        counts2={}
        total = 0.0 #total de palabras en la clase
        self.archivos[category]=0.0 #el numero de archivos en la categoria
es 0 inicialmente
        for file in files:
            #print(currentdir + '/' + file)
            f = codecs.open(currentdir + '/' + file, 'r', 'utf-8') #f es el
archivo abierto
            for line in f:
                tokens = line.split() #las palabras de cada linea
                for token in tokens:
                    # get rid of punctuation and lowercase token
                    token = token.strip('\'".,?:-') #elimina casos raros
                    token = token.lower()
                    if token != '': #no es vacio ni esta en los stopwords
                        self.vocabulary.setdefault(token, 0) #si no esta lo
deja en 0
                        self.vocabulary[token] += 1 #agrega una aparicion
                        counts.setdefault(token, 0) #en counts tb lo deja en
0 si no esta
                        counts[token] += 1 #le suma uno a count
                        counts2.setdefault(token, 0) #en counts tb lo deja
en 0 si no esta
                        counts2[token] += 1 #le suma uno a count
                        total += 1 #el total sube uno siempre
            f.close()
            self.archivos[category]=self.archivos[category]+1
        print counts

        return(counts2, counts, total) #retorna el diccionario de la clase,
y el total de palabras
```

```
    def classify(self, filename): #recibe la ruta del archivo, es independiente
de la clase
        results = {}
        for category in self.categories:
            results[category] = 1 #inicializa todas las clases en 0
        f = codecs.open(filename, 'r', 'utf-8') #abre el archivo
        #print filename
        #print f
        for line in f: #por cada linea
            tokens = line.split() #extrae las palabras
            for token in tokens: #por cada palabra
                #print(token)
                token = token.strip('\'".,?:-').lower() #quita los raros
                if token in self.vocabulary: #si esta en el vocabulario
                    for category in self.categories: #por cada clase calcula:
                        if self.probaFi[category][token] == 0: #si la prob es
cero para la clase
                            print("%s %s" % (category, token)) #imprime el
error
                        #print category, token, self.probaFi[category][token]
                        results[category] = results[category]*
self.probaFi[category][token] #el resultado para la clase es producto de las
probs de las palabras para la clase
        f.close()
        #print "results", results
        p={}
        for category1 in self.categories:
            p[category1]=0.0
            for category2 in self.categories:
                #print results[category1]

p[category1]=p[category1]+(self.probC[category2]*results[category2])/(self.pro
bC[category1]*results[category1])
        esUno=0.0
        for category in self.categories:
            p[category]=1/p[category]
            esUno=esUno+p[category]
        #tiene todos los resultados por clase, palabra
        probas=list(p.items())
        probas.sort(key=lambda tuple: tuple[1], reverse = True)
        return probas[0][0]


def testCategory(self, directory, category):
        print "Test", category
        print
        files = os.listdir(directory) #extrae los archivos
        print "N de archivos:", len(files)
        total = 0
        correct = 0
        resultadosC={}
        for category in self.categories:
            resultadosC[category]=0.0
        for file in files:
            total += 1 #suma los archivos
            #print total
            result = self.classify(directory + file) #llama al clasificador
con el archivo, recibe el mayor
            if result == category: #si coincide con la categoria
                correct += 1 #lo predijo bin
            resultadosC[result]=resultadosC[result]+1
        return (correct, total, resultadosC) #retorna la cantidad que predijo
bien, el total, y como clasifico
```

```
def test(self, testdir): #recibe el directorio del test
        print ""
        print "---------"
        print "TEST"
        print ""
        categories = os.listdir(testdir) #extrae las categorias
        print "Categorias"
        for ca in categories:
            print ca
        categories = [filename for filename in categories if
os.path.isdir(testdir + filename)]
        correct = 0.0
        total = 0.0
        print ""
        resultadosFinales={}
        for category in categories:
            (catCorrect, catTotal, catRes) = self.testCategory(testdir +
category + '/', category) #entrega la carpeta y la categoria
            correct += catCorrect
            total += catTotal
            resultadosFinales[category]=catRes
        print("Accuracy is  %f%%  (%i test instances)" % ((correct / total) *
100, total))
        print resultadosFinales

        sumC={}
        corrC={}
        for category in categories:
            sumC[category]=0.0
            corrC[category]=0.0
            for cat2 in categories:
                sumC[category]+=resultadosFinales[category][cat2]
                if cat2==category:
                    corrC[category]=resultadosFinales[category][cat2]
        for category in categories:
            print category, ((corrC[category]/sumC[category])*100)
```