PROYECTO DE GRADO
BÚSQUEDA DE OPINIONES EN MEDIOS DE PRENSA

Estudiantes: Rodrigo Stecanella (rodrigosteca@gmail.com); Jairo Bonanata (jbonanat@gmail.com)

Supervisores: Dina Wonsever; Mathías Etcheverry

Resumen

En este documento se presenta la construcción y evaluación de un sistema de recuperación de información, que permite, a partir de una base de texto extraída de medios de prensa uruguayos, buscar opiniones que hayan sido emitidas por diversas fuentes. Es decir, este sistema permite realizar búsquedas que contesten a la pregunta ¿Qué opinó X sobre el tema Y?

Este trabajo se basa en dos proyectos anteriores realizados por estudiantes y profesores de la Facultad de Ingeniería. El primer proyecto es un módulo de Identificación de Opiniones, que, a partir de un texto que toma como entrada, identifica y anota las opiniones que existan en el mismo. El segundo proyecto toma como entrada la salida del módulo anterior y soluciona el problema de las correferencias entre las fuentes de distintas opiniones para un mismo texto.

Por otro lado, aprovechando el corpus de noticias generado, se implementó una herramienta que permite saber cuáles fueron los temas de los que más habló la prensa en una semana determinada. Esto es, dada una semana, se presentan cuáles fueron las noticias más importantes en la misma.

Para el sistema de búsquedas de opiniones se consiguió obtener un 76% de precisión, mientras que para la herramienta de “los temas de la semana” se obtuvo un 86% de precisión y un 84% de recall.

Contenido

1 INTRODUCCIÓN

1.1 PLANTEO DEL PROBLEMA

Actualmente es muy común la lectura de prensa electrónica, por lo que resulta de utilidad contar con sistemas para la extracción de diferente tipo de información según las necesidades de los usuarios. Para esto es necesario conocer las características propias de cada tipo de texto de modo de lograr un análisis correcto de su contenido. En particular, en los textos de prensa es muy frecuente la mención por parte del autor de la noticia a opiniones de otras personas. Este sistema busca la información en los textos publicados en la web, pero es posible pensar en sistemas similares que hagan sus búsquedas en bases de textos.

Se propone el desarrollo de un sistema que, a través de la consulta a una base de textos de prensa, permita obtener y visualizar las diferentes opiniones emitidas por una fuente determinada (personas, instituciones, publicaciones), seleccionando además los temas de interés.

Se cuenta con dos módulos desarrollados por estudiantes de la facultad de ingeniería. Uno que permite la extracción de opiniones de textos en español, obteniendo la fuente, el asunto y el mensaje; y otro que permite realizar las correferencias entre distintas fuentes dentro de un mismo texto.

En este capítulo se presenta el problema planteado y su contexto. Se menciona posteriormente el área de motivación del trabajo y al final del capítulo los objetivos que se persiguen en este proyecto. Contiene además una guía del documento.

1.2 MOTIVACIÓN

En los últimos años, a medida que cada vez más personas tienen una computadora a su disposición y la información pasa de estar en el papel a estar almacenada en archivos en computadoras que a su vez son accesibles a través de la red, surge el problema de poder encontrar lo que se está buscando en la inmensidad de documentos disponibles. Se estima que al 2013 existen unas 45 mil millones de páginas web (1), esto es, alrededor de 6 páginas web por cada persona que vive en el planeta. Ante este problema, es de vital importancia tener herramientas que nos permitan rápidamente poder acceder a la información que estamos buscando.

Por otro lado, y siguiendo con esta tendencia, hoy en día la mayoría de los medios de prensa escrita tienen una presencia importante en la red de redes, siendo cada vez más común la lectura de noticias a través de las páginas web en detrimento del formato impreso.

Con esto en mente, uno de los problemas que se plantea es obtener información acerca de las opiniones emitidas por figuras relevantes de la sociedad. Planteándolo por ejemplo en un período de elecciones políticas, es importante obtener información acerca de qué es lo que piensa cada candidato sobre los diversos temas que se plantean en la campaña electoral.

En la actualidad no existe ningún sistema que resuelva este problema de una forma realmente efectiva y mucho menos para el idioma español.

1.3 OBJETIVOS

El objetivo principal de este proyecto es crear una herramienta que permita la recuperación de opiniones extraídas de artículos de prensa en español. Para esto se necesita realizar varias tareas y por lo tanto este objetivo se puede desglosar en los siguientes:

- Crear un corpus de artículos de prensa en español, los cuales deberán ser extraídos de forma automática de los sitios web de la prensa uruguaya.
- Extraer metadatos acerca de los artículos de prensa (título, etc).
- Utilizar un motor de búsqueda para poder realizar una correcta recuperación de las opiniones a partir de la fuente de la misma y palabras clave acerca del tema de la opinión.
- Crear una interfaz de usuario amigable para poder realizar las búsquedas fácilmente.

También se plantea como objetivo, directamente relacionado con lo anterior, la implementación de un sistema que permita determinar cuales son "los temas del momento" para períodos cortos de tiempo utilizando la información obtenida a través del corpus de artículos de prensa.

1.4 MARCO DE TRABAJO

Este proyecto se enmarca dentro del área de la Recuperación de Información (IR, por sus siglas en inglés: **I**nformation **R**etrieval).

La recuperación de información consiste en encontrar material de naturaleza desestructurada que cumple una necesidad de información a partir de una o varias colecciones. (2)

La expresión "datos desestructurados" se refiere a datos que no tienen una semántica clara, o a datos que no son fácilmente almacenables en una estructura de datos adecuada para una computadora. El opuesto a esto son los datos estructurados donde el ejemplo más simple es una base de datos relacional. Sin embargo, prácticamente no existen datos verdaderamente desestructurados, pues en general toda información tiene cierta estructura (hasta el texto escrito en lenguaje natural tiene una estructura), por eso la IR también cubre la recuperación en información "semi-estructurada".

El área de la recuperación de información también abarca el dar soporte a usuarios en la navegación y filtrado de colecciones de documentos, adicionalmente se puede dar un procesado a dichos documentos.

Ejemplos donde se aplica la Recuperación de Información pueden ser recuperar documentos electrónicos de cualquier tipo de colección documental digital, búsqueda de metadatos que describan documentos, o también la búsqueda en bases de datos relacionales. Como objetivo se puede realizar la recuperación en textos, imágenes, sonido o datos de otras características, de manera pertinente y relevante.

Dentro del área de recuperación de información se puede identificar a los motores de búsqueda, que permiten, mediante palabras clave, una búsqueda para recuperar información en base a documentos almacenados en un índice. El principal exponente dentro de esta área es sin lugar a dudas Google (3) que mediante su motor de búsqueda permite rápidamente

encontrar información en la web y que también ha desarrollado herramientas como Google Search Appliance (GSA) (4) que permite la búsqueda de archivos dentro de una intranet o en una única computadora.

Por otro lado, el IR se encuentra muy ligado al Procesamiento del Lenguaje Natural (PLN) (5). Este es un campo de las Ciencias de la Computación, Inteligencia Artificial (IA), y la lingüística el cual trata la interacción entre los lenguajes humanos y las computadoras. El principal desafío dentro del PLN es la comprensión del lenguaje natural, esto es, que una computadora puede entender el *significado* de una entrada de lenguaje humano.

El PLN es un área muy grande que comprende varias sub-disciplinas: la extracción de información, la traducción automática, la generación de resúmenes, sistemas de respuestas a preguntas, ayuda en preparación de textos y verificadores de gramática, búsqueda de información, gestión inteligente de documentos, tecnologías de reconocimiento de voz, etc.

1.5 ANÁLISIS DEL PROBLEMA

1.5.1 Extracción de información de los medios de prensa

Hoy en día es común que medios de prensa escrita publiquen sus contenidos en la web de forma gratuita y de libre acceso, existiendo medios de prensa que tienen formato impreso y otros que tienen solamente el formato web. Nuestro problema consiste en extraer información de prensa en forma automática de los medios que se publican en la web. Para ello se puede utilizar diversas técnicas y herramientas, entre las cuales destacan expresiones regulares y XPath.

En principio los medios de los cuales vamos a extraer la información para nuestro proyecto, son El País, La República y El Observador puesto que son los que tienen mayor presencia en internet en nuestro país. Sin embargo, nada descarta la posibilidad de utilizar otros medios o agregar medios nuevos en el futuro.

La cantidad de medios de los cuales se extrae información es acotada y conocida, por lo que se puede utilizar métodos exactos para extraer la información (lo cual no descarta el uso de métodos probabilísticos y heurísticas). En este documento se describe de cuáles páginas se extrae la información, y las distintas técnicas que se utilizan para lograrlo. Sin embargo, intentaremos que los métodos utilizados sean los más generales posibles con el objetivo de poder agregar nuevos medios de prensa en el futuro.

De los medios anteriormente citados se consideran los datos recientes, así como datos más antiguos. Cabe destacar que La República no tiene contenidos antiguos, pues son eliminados periódicamente, por lo que los datos son sólo los que se consiguen a partir de que se empezó a extraer por primera vez. Esto no ocurre para El Observador y El País, para los que es posible extraer un corpus histórico de prensa de varios años hacia atrás.

1.5.2 Buscador de opiniones

Una de las propuestas de este trabajo es la necesidad de buscar opiniones en nuestro corpus previamente procesado. Se debe de poder buscar por fuente y por tema. Además, se pueden añadir filtros adicionales opcionales como por ejemplo rangos de fechas y medio de prensa al cual pertenece el artículo de la opinión extraída. Por lo tanto, es necesaria una forma de búsqueda rápida y eficiente para las opiniones extraídas. Además, más allá de la velocidad con

la que se recuperan las opiniones, surge el problema de la relevancia de las opiniones recuperadas con la búsqueda, por lo que se deberá profundizar en la utilización de técnicas que sean adecuadas para este problema.

Entonces, para resolver el problema es necesaria una herramienta que nos permita buscar en forma eficiente y que se pueda adaptar a nuestras necesidades. Dentro de estas herramientas hay que destacar las búsquedas de texto completo, junto con otros agregados que nos permiten buscar por distintas características de las opiniones, dándoles mayor o menor relevancia según sea conveniente (dicha conveniencia se analizará más adelante en la propuesta de la solución).

1.5.3 Temas de la semana

Uno de los objetivos de este proyecto es poder realizar una búsqueda acerca de cuáles fueron los temas sobre los cuales más se habló durante el lapso de tiempo de una semana. El problema consiste entonces en, tomando todas las noticias extraídas de la web para dicho período de tiempo, determinar sobre qué se habló más en los medios de prensa.

El principal obstáculo en esta parte del proyecto será determinar cual es el "tema" de una noticia en particular, para luego ver cuáles son los temas que más se repiten a través de las diferentes noticias en una semana determinada.

Este problema puede verse como algo similar a lo que realiza google news, donde muestra cuáles son los temas sobre los que se está hablando en el momento en los medios de prensa en la web. Sin embargo google news no tiene la funcionalidad de permitir ver para un determinado período de tiempo cuáles fueron los temas más importantes.

1.5.4 Interfaz de usuario

Parte del objetivo de este proyecto es que un usuario final pueda utilizar la herramienta que desarrollaremos para realizar búsquedas de opiniones y ver cuales son los temas de una semana determinada. Por lo tanto, una de las problemáticas será desarrollar una interfaz que sea amigable con el usuario y que permita un uso eficaz de la herramienta. Por otro lado, también es esperable que los tiempos de carga sean relativamente bajos no teniendo que esperar más de unos pocos segundos luego de realizada una consulta.

1.6 GUÍA DEL DOCUMENTO

En el capítulo 2 del documento se da una descripción de los distintos módulos preexistentes que serán utilizados, así como varios conceptos que serán de utilidad a lo largo del trabajo. A su vez, también se describen algunas de las herramientas principales que serán utilizadas para el desarrollo del proyecto así como se mencionan otros proyectos relacionados.

En los capítulos 3, 4, 5, 6 y 7 se detalla el diseño y la implementación de los distintos módulos que fueron necesarios para el desarrollo del sistema.

En el capítulo 8 se realiza la evaluación del sistema desarrollado mediante distintas pruebas que allí se proponen.

Por último, en el capítulo 9 se desarrollan las conclusiones y el trabajo a futuro a realizar sobre el proyecto.

2 ANTECEDENTES

2.1 PROYECTOS ANTERIORES

2.1.1 Introducción

Este trabajo se basa en proyectos desarrollados por estudiantes de la Facultad de Ingeniería. El primer proyecto, Identificación de Opiniones de diferentes fuentes en textos en español (6), fue desarrollado por Aiala Rosá para su tesis de doctorado; el segundo, Resolución de Correferencias (7), fue desarrollado como proyecto de grado por Fernando Acerenza, Macarena Rabosto y Magdalena Zubizarreta.

2.1.2 Identificación de Opiniones

2.1.2.1 Descripción

En este proyecto se presenta un estudio de las expresiones que transmiten opiniones de diferentes fuentes en textos en español. El trabajo incluye la definición de un modelo para los predicados de opinión y sus argumentos (la fuente, el asunto y el mensaje), la creación de un léxico de predicados de opinión que tienen asociada información proveniente del modelo y la realización de tres sistemas informáticos.

Se desarrolló un primer sistema, basado en reglas contextuales, que obtiene valores de medida F parcial (incluyendo entre los elementos correctos los elementos reconocidos en forma parcial) satisfactorios: 92 % para el predicado, 81 % para la fuente, 75 % para el asunto, 89 % para el mensaje y 85 % para la opinión completa. En particular, para el reconocimiento de la fuente se obtuvo un 79 % de medida F exacta (sin incluir elementos reconocidos en forma parcial).

El segundo sistema desarrollado se basa en el modelo Conditionnal Random Fields (CRF) y se realizó sólo para el reconocimiento de las fuentes. El sistema alcanza un valor de medida F exacta de 76 %.

Un tercer sistema, que combina las dos técnicas anteriores incorporando la salida del sistema de reglas para el reconocimiento de fuentes como un nuevo atributo del sistema basado en CRF, mejora sensiblemente los resultados obtenidos por los dos sistemas anteriores: 83 % de medida F exacta.

En cuanto al reconocimiento de las fuentes de las opiniones, el sistema obtiene resultados muy satisfactorios (83 % de medida F exacta), si se toma como referencia trabajos realizados para otros idiomas que pueden considerarse similares a éste, si bien presentan varias diferencias en su enfoque y su alcance. Estos trabajos alcanzan valores de medida F (exacta o parcial) que se sitúan entre 63 % y 89,5 %.

2.1.2.2 Definición de una opinión

Para este trabajo se toma como definición que una opinión es un “segmento de texto que transmite las expresiones o posturas de alguna fuente sobre algún asunto”. De esta forma se define que una opinión consta de cuatro elementos:

1. **El predicado de opinión** - Son verbos como “opinó”, “rechazó”, etc. y sustantivos como “opinión”, “rechazo”, etc.

2. **La fuente** - El que emite la opinión.
3. **El tema** - Mención explícita del tema sobre el cual se emite la opinión.
4. **El mensaje** - El contenido en sí mismo de la opinión.

Por ejemplo, lo siguiente es una opinión extraída de un artículo de prensa con los cuatro elementos:

- Consultado *sobre la lentitud de los procesos judiciales uruguayos* Carranza **respondió**: "Hay una situación de un muy alto número de presos sin condena, hay que agilizar los procesos".

Donde el texto resaltado en cursiva es el tema, subrayada está la fuente, en negrita el predicado y en gris el mensaje de la opinión.

Cabe destacar que no en todas las opiniones que se encuentran en textos de prensa están los cuatro elementos. Los únicos que no pueden faltar son el predicado y el mensaje puesto que estos hacen a la opinión en sí misma. En el idioma español es bastante común que la fuente esté omitida por haberla mencionado anteriormente dentro del texto. Por otro lado, es raro que aparezca una mención explícita al tema de la opinión dentro de la misma.

2.1.2.3 Ejemplo de una salida del programa

Dada la siguiente entrada:

Mujica respaldó importante inversión minera

El siguiente es un ejemplo de la salida del programa para el reconocimiento de una opinión:

```
<opinion so="pos">
  <source so="neu">Mujica</source>
  <predicate so="pos">respaldó</predicate>
  <topic so="neu">importante inversión minera</topic>
</opinion>
```

2.1.3 Resolución de Correferencias

2.1.3.1 Descripción

En este trabajo se presenta la construcción y evaluación de un módulo de resolución de correferencias entre fuentes de opiniones en español, contenidas en textos de prensa. El módulo toma textos etiquetados con opiniones y sus componentes identificados, más información sintáctica, realizando la recuperación de fuentes y resolución de correferencias entre fuentes de las opiniones.

La entrada es generada por el módulo que realiza la identificación de opiniones. La resolución se basa en un algoritmo de manejo de puntajes para resolver el antecedente correcto de la lista de candidatos a correferir con una fuente.

Se genera una salida en XML que puede servir como entrada de otro componente y una salida en formato HTML para poder ser visualizada en cualquier navegador de manera amigable.

Se estudia el recall y precisión mediante pruebas sobre textos preparados. Los resultados obtenidos son del 81% en precisión y 80% en recall.

2.1.3.2 Ejemplo de la salida del programa

A continuación mostramos una salida del programa:

```
<texto>
     TEXTO 28 . Interpelarán a ministra de Salud María Julia Muñoz .
     Dra. Muñoz , presentarse en sala . 11.08.2009 14 : 11 .
    <opinion>
         El Partido Nacional interpelará a
        <fuenteRec id="1">
             la ministra de Salud Pública , María Julia Muñoz
        </fuenteRec>
        , para que dé
        <opinion>
            <fuente id="2" idRec="1"/>
             explicaciones sobre irregularidades en el manejo de ASSE
        </opinion>
         , según informó a Montevideo Portal
        <fuente id="1" idRec="0">
             el diputado nacionalista Jorge Gandini
        </fuente>
    </opinion>
     . El Partido Nacional acordó llamar a interpelación a la ministra de Salud Pública María Julia Muñoz
     por presuntas irregularidades en la Administración de los Servicios de Salud del Estado .
     En conversación con Montevideo Portal
    <opinion>
        <fuente id="3" idRec="0">
             el diputado nacionalista Jorge Gandini
        </fuente>
         señaló que la interpelación se convocará formalmente el
         miércoles y que se tomó la decisión porque la semana pasada
        <opinion>
            <fuente id="4" idRec="0">
                 se
            </fuente>
             quiso llamar a Comisión a las autoridades de ASSE
        </opinion>
         y
        <opinion>
             la iniciativa no contó con el apoyo de
            <fuente id="5" idRec="0">
                 los diputados del Frente Amplio
            </fuente>
        </opinion>
    </opinion>
     . El principal caso será el de la empresa de limpieza Clanider SA , que está siendo investigada
     por la Justicia, el gobierno central y el propio Hospital Maciel , donde se habría registrado
     un caso de sobrefacturación en el servicio que proporcionaba la empresa .

    <link id="4" idF1="1" idF2="3"/>
    <link id="5" idF1="3" idF2="4"/>

</texto>
```

Por un lado podemos ver cómo el programa taggea las opiniones utilizando la información obtenida por el módulo de extracción de opiniones, reconociendo una fuente que no había sido reconocida en un principio. Por otro lado, también podemos ver que al final del archivo de xml de salida, se genera una cadena de correferencias para todas las fuentes que aparecen dentro del texto.

2.2 SISTEMAS DE RECUPERACIÓN DE INFORMACIÓN

2.2.1 Introducción

Un sistema de recuperación de información es un sistema que se encarga de cualquiera de las tareas que se enmarcan en el área de la recuperación de información. Una breve descripción de la misma fue dada en la introducción de este documento. En general un sistema de

recuperación de información está compuesto por varios componentes para poder funcionar y más aún hacerlo en forma eficiente, por ejemplo índices, parsers de consulta, sistemas de puntaje, caché de documentos, etc.

En esta sección nos dedicamos a analizar algunos sistemas de recuperación de información que se encargan de: dado un conjunto de palabras clave introducidas por el usuario, que llamaremos consulta, recuperar documentos con información relevante. Este tipo de sistemas también se denominan motores de búsqueda. Es común que un motor de búsqueda sea parte de un sistema de recuperación de información.

A lo largo del documento utilizaremos “Sistema de Recuperación de Información”, “Motor de Búsqueda” y “Buscador” como sinónimos.

2.2.2 Lucene

Apache (8) Lucene (9) (10) es una biblioteca de software de recuperación de información libre y de código abierto, creada originalmente en Java por Doug Cutting. Es apoyado por la Apache Software Foundation y se distribuye bajo la licencia Apache Software.

Lucene ha sido portado a otros lenguajes de programación como Delphi, Perl, C#, C++, Python, Ruby y PHP.

Entre las características de esta herramienta se encuentran:

- Un índice invertido de almacenamiento persistente basado en texto para una recuperación eficiente de documentos con términos indexados.
- Un amplio conjunto de procesadores de texto para transformar una cadena de texto en una serie de términos (palabras), que son las unidades fundamentales a indexar y buscar.
- Un amplio lenguaje de consulta para poder realizar desde consultas por rango hasta consultas de aproximación de strings, también llamadas fuzzy queries, donde se utiliza la distancia de Levenshtein para calcular la proximidad entre la consulta realizada y los términos a buscar.
- Un buen algoritmo de relevancia basado en los principios de la Recuperación de Información para producir los más probables candidatos en primer lugar, con promedios flexibles para afectar a la puntuación de los resultados.
- Una característica de resaltado (highlighting) para mostrar palabras encontradas en el contexto.
- Un corrector ortográfico basado en la consulta sobre contenido indizado.

2.2.3 Solr

Apache Solr (11) es un motor de búsquedas de código abierto parte del proyecto Apache Lucene (2). Entre sus principales características se encuentra la posibilidad de efectuar búsquedas de texto completo. Solr al igual que Lucene está implementado en java y corre sobre un contenedor de servlets como puede ser Apache Tomcat o Jetty, de forma independiente al resto de las aplicaciones.

Solr, al estar basado en Lucene (de hecho comparten gran parte del código fuente), posee sus mismas funcionalidades básicas, siendo la mayor diferencia que Lucene es una API para Java, mientras que Solr se basa en archivos de configuración y se comunica por medio del protocolo HTTP con el resto de las aplicaciones.

La interfaz utilizada por otra aplicación para acceder a las búsquedas, es (como se mencionó anteriormente) por medio de HTTP, lo cual lo hace versátil, pues para consultar la información, solo se necesita tener acceso al servidor en el que corre Solr. Por otro lado, hay varias formas de cargar datos en Solr: Archivos XML, MySql, JSON. Esto último, más el agregado de la comunicación vía HTTP, lo hace independiente del lenguaje de programación utilizado para implementar otros módulos, haciendo fácil su integración en distintos proyectos.

Apache Solr se configura a través de archivos de configuración XML, entre los cuales destacan:

- Schema.xml que tiene información de cómo indizar los datos y todo lo respectivo a los campos a indizar. También se dan los criterios para poder buscar en dichos campos, y como deben ser tratados a la hora de añadir documentos al índice.
- SolrConf.xml que contiene información de Solr en sí mismo. Por ejemplo la utilización de bibliotecas y el puerto TCP en el que se ejecuta Solr.

Una extensa documentación sobre Solr puede ser encontrada en el siguiente libro: (12)

2.2.4 Comparación de Solr con un RDBMS

El índice de Lucene-Solr podría verse como algo similar a lo que sería una base de datos relacional con una única tabla que contiene toda la información. Sin embargo, existen diferencias sustanciales en la forma en las que se manejan las estructuras de datos internamente y a continuación presentamos algunas de ellas:

- Updates: En Lucene-Solr los documentos pueden ser agregados o borrados, pero no actualizados.
- Búsqueda de substrings vs búsqueda de texto: Utilizando una base de datos, una consulta para realizar una búsqueda sería por ejemplo: “SELECT * FROM mitabla WHERE nombre LIKE ‘%libros%’”. Esta consulta daría como resultado “libros de cocina” pero no“mi libro”. Por otro lado, Lucene fundamentalmente realiza búsquedas sobre términos (o palabras). Dependiendo de la configuración esto puede significar la búsqueda de las varias formas de una misma palabra, por ejemplo “libro” como singular tanto como “libros” en plural. A su vez, también pueden realizarse búsquedas de forma fonética por cómo suenan las palabras y no necesariamente por cómo están escritas junto con otros métodos de búsqueda.
- Resultados con puntaje: Gran parte del poder de Lucene es su capacidad para asignar un puntaje a cada documento según qué tan bien haya coincidido con los términos de búsqueda. A su vez, existe una variedad de otros factores y es posible ajustar las ponderaciones de los diferentes campos. En comparación, una base de datos no tiene este concepto, un registro o bien es devuelto en la consulta o no es devuelto. Por otro lado, Lucene también puede ordenar los documentos según algún otro campo (por ejemplo un valor de fecha).
- Commits lentos: Solr está altamente optimizado para las búsquedas, utilizando una gran cantidad de cachés. Por lo tanto los commit necesitan reconstruir estos últimos y esto puede insumir grandes cantidades de tiempo para índices de un tamaño considerable.

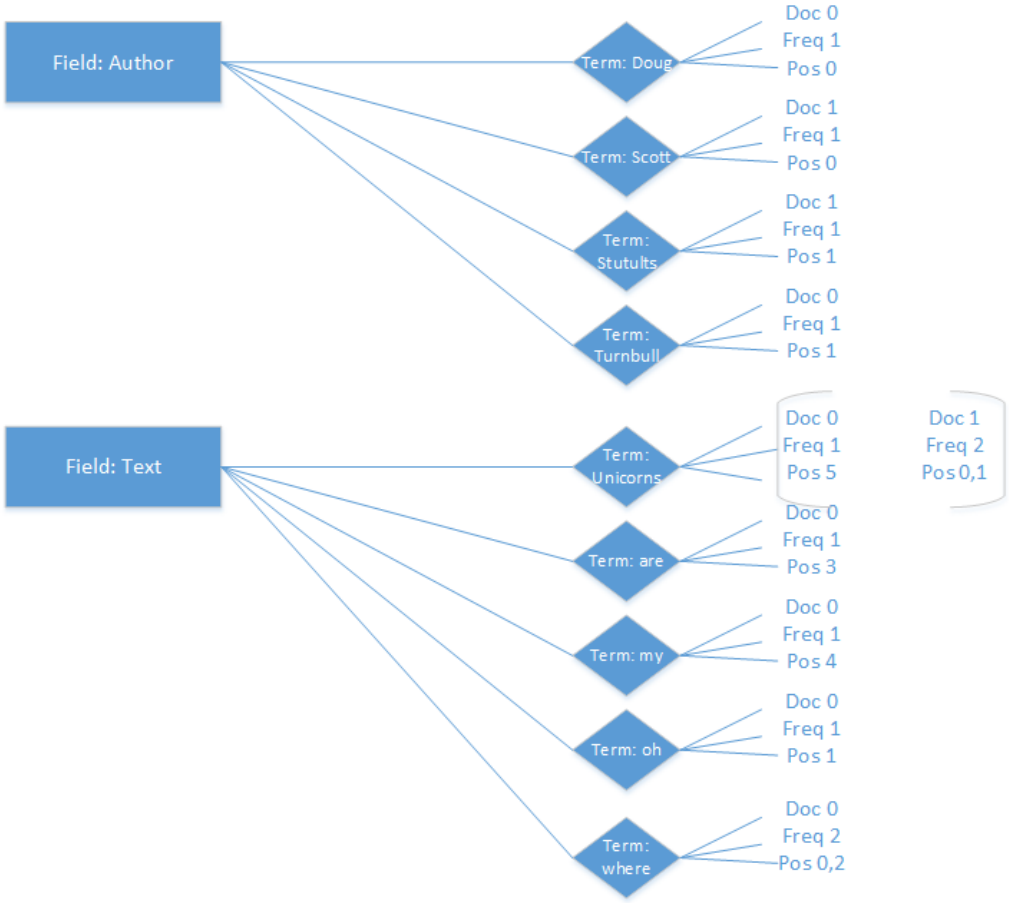
2.2.5 Cómo funciona una búsqueda con Lucene

Si bien Lucene se maneja con algoritmos complejos que implican mucho procesamiento sobre el lenguaje natural, la base de su funcionamiento es bastante sencilla y la mejor forma de entenderlo es a través de un ejemplo. Este ejemplo fue extraído de (13).

Supongamos que tenemos los dos siguientes documentos:

doc0:
  author: Doug Turnbull
  text: where oh where are my Unicorns!
doc1:
  author: Scott Stults
  text: Unicorns! Unicorns!

Si el único procesamiento que le decimos a Lucene que haga es partir las palabras por espacios en blanco para luego indexar los documentos, obtenemos el siguiente índice:



Este tipo de índice es llamado un índice invertido (14), que es algo similar a lo que tienen muchos libros al final con una lista de palabras y las páginas donde dichas palabras aparecieron. De esta forma, en vez de tener un índice donde hay una serie de documentos que apuntan a un conjunto de palabras, tenemos una serie de palabras que apuntan a un conjunto de documentos. Por lo tanto, si queremos hacer una búsqueda, simplemente tenemos que buscar en nuestro índice las palabras deseadas y retornar una lista con los documentos que la contienen.

Sin embargo, a diferencia de lo que ocurre con los índices al final de los libros, aquí también tenemos información acerca de cuántas veces aparece un término en un documento determinado y en qué posición aparece.

Por ejemplo, volviendo a nuestro ejemplo:

term Unicorns!
  doc 0
    freq 1
    pos 5
  doc 1
    freq 2
    pos 0
    pos 1

Se puede ver que en el documento 0, la palabra Unicorns! aparece una vez en la posición 5 y en el documento 1 aparece dos veces primero en la posición 0 y luego en la posición 1.

La frecuencia con la que aparece un término dentro de un documento determina qué tan relevante es ese documento cuando efectuemos la búsqueda de dicho término. Si en comparación, aparece más veces el término “Unicorns” en el documento 1, entonces ese documento obtendrá un puntaje más alto. Este modelo de búsqueda se conoce como el modelo TF-IDF (15), donde se tienen las siguientes ecuaciones:

tf(t,d) = la frecuencia del término t en el documento d

$$\operatorname{idf}(t, D) = \log \frac{|D|}{|\{d \in D : t \in d\}|}$$

Donde D es el conjunto de todos los documentos indexados y t es el término buscado. De esta forma, el puntaje del documento se calcula como el producto de tf (la frecuencia de un término de un documento) con idf (la frecuencia inversa del término dentro de todos los documentos indexados, o sea, qué tan raro es el término).

Para tomar como ejemplo y siguiendo con la analogía del índice de un libro, digamos que realizamos la búsqueda “Edgar Allen Poe” en un libro sobre la literatura del siglo XIX. Si supiéramos que a través del índice que hay más menciones de “Edgar Allen Poe” en la página 15 que en la página 5, probablemente iríamos a la página 15.

Por otro lado, la posición en la que aparecen los términos también es importante. Supongamos en nuestro ejemplo anterior que no listamos "Edgar Allen Poe" en el índice del libro. En su lugar, simplemente tenemos las palabras individuales "Edgar", "Allen" y "Poe" . Podríamos, sin embargo, averiguar si había una mención estricta de la frase "Edgar Allen Poe" si guardamos la posición de cada término en una página en el índice de nuestro libro. Por ejemplo, "Edgar" es la palabra del número 3 en la página 5 , "Poe" es la palabra número 17 en la página 5 , etc. Esto sería engorroso para nosotros los seres humanos, pero podríamos hacerlo. Sin embargo esto se puede implementar de forma eficiente en una computadora y es precisamente lo que hace Lucene. Lucene puede determinar rápidamente la distancia y la disposición de los términos de búsqueda de un documento en el momento de la consulta . Si los términos "Edgar", "Allen" , y "Poe" son encontrados en un documento y están uno al lado del otro en el orden correcto, Lucene puede determinar que ese es el documento que estábamos buscando.

2.3 EXTRACCIÓN DE INFORMACIÓN DE LA WEB (CRAWLERS Y SCRAPERS)

2.3.1 Introducción

De aquí en más y en el resto del documento se utilizan las palabras Crawler y Scraper para los programas que se describen a continuación. También se utilizan los términos crawling y scraping para referir a las tareas realizadas por estos programas.

Los llamados en inglés Web Crawler (o Arañas) son programas que inspeccionan las páginas web de forma metódica y automatizada. Uno de los usos más frecuentes que se les da consiste en crear una copia de todas las páginas web visitadas para su procesado posterior por un motor de búsqueda que indexa las páginas proporcionando un sistema de búsquedas rápido y eficiente.

Los Scraper son programas que utilizan diversos algoritmos para extraer datos a partir de páginas web. Por ejemplo el título de un documento embebido en la página, la fecha de publicación del mismo, un párrafo en particular, o incluso metadatos utilizados por redes sociales.

2.3.2 Nutch

2.3.2.1 Descripción

Apache Nutch (16) es un web crawler open-source altamente extensible y escalable. Su objetivo final es implementar un motor de búsqueda para la web (utilizando Lucene) que permita fácilmente crawlear e indexar páginas web para luego realizar búsquedas sobre las mismas.

Como comentaremos en el capítulo 3 se eligió no utilizar la integración que ofrece Nutch con Solr, sino que utilizamos Nutch únicamente para descargar todas las páginas de los sitios web. El procesamiento del HTML se realiza por separado mediante la implementación de un módulo específico para eso.

2.3.2.2 La arquitectura de Nutch

Nutch se divide naturalmente en dos partes, el crawler y el motor de búsqueda. El crawler lo que hace es descargar las páginas e introducirlas en el índice donde luego el motor de búsqueda realiza sus consultas. Por otro lado el motor de búsqueda es básicamente una instancia de Solr.

La principal ventaja de esta división es que se puede dividir en hardware distinto lo que es el crawling de la búsqueda siendo esto muy importante si se requiere un sistema con una alta performance. A su vez, tanto el crawler como la instancia Solr son altamente paralelizables para su ejecución en clústeres mediante Hadoop (17) o Solr Cloud (18).

2.3.2.3 El Crawler

El sistema de crawling de Nutch está manejado por una familia de herramientas que manejan diversas estructuras de datos. Entre estas se encuentran la base de datos web, un conjunto de segmentos y el índice que describiremos a continuación.

La base de datos web (**WebDB**) es una estructura de datos persistente que tiene como objetivo mantener la estructura y propiedades del grafo web que está siendo crawleado. La WebDB no cumple ningún rol durante la búsqueda de información, sino que guarda dos tipos de entidades: páginas y links.

Una página representa una página en la web y es indexada por su URL y un hash MD5 de sus contenidos. Además también se guarda otra información acerca de la página incluyendo la cantidad de links en la misma (outlinks), información acerca de cuándo la página fue crawleada (para saber cuándo re-crawlear), el puntaje de la página (basado en la cantidad de links que apuntan a la misma).

Un link representa una url encontrada dentro de una página que apunta hacia otra distinta. En el grafo web de la WebDB un nodo es una página y un vértice es un link.

Un **segmento** es un conjunto de páginas crawleadas e indexadas en una misma corrida del programa. La fetchlist de un segmento es la lista de de URLs de páginas que el crawler debe descargar y es generada a partir de la WebDB. La información de las páginas guardadas en un segmento se vuelve obsoleta una vez que todas las páginas han sido re-crawleadas. El período de tiempo para un re-crawleo es configurable.

El **índice** es un índice invertido de Lucene que ya describimos anteriormente. Este se crea a partir de la unión de todos los segmentos.

2.3.2.4 La herramienta de crawling

El crawling es un proceso cíclico que sigue los siguiente pasos: el crawler genera una fetchlist de urls a crawlear a partir de la WebDB, las páginas correspondientes a dichas urls se descargan y se guardan en segmentos, el crawler actualiza la WebDB con los nuevos links que fueron encontrados, el crawler genera una nueva fetchlist y el ciclo se repite.

2.3.2.5 Configuración de Nutch

Nutch puede ser fácilmente configurado a través de archivos XML y hablaremos en más profundidad acerca de esto en la sección de implementación. A su vez, el funcionamiento del mismo puede ser modificado a través de la creación de plugins, sobre todo enfocados en permitir hacer un scraping de los sitios web.

Por una descripción más detallada acerca del funcionamiento de Nutch pueden consultarse (19) y (20).

2.3.3 Scrapy vs Nutch

Scrapy es un framework para realizar web crawling de sitios web con un especial enfoque en realizar un scraping de los mismos. El mismo es un proyecto de código abierto escrito en Python.

A la hora de elegir una herramienta que nos ayudara a realizar el web crawling de los sitios web de los medios de prensa, analizamos varias alternativas, muchas de las cuales están incluidas en (21), y nos quedamos con dos de las cuales nos parecieron las más robustas (Nutch y Scrapy) para finalmente decidir cuál será utilizada.

Mientras que Scrapy es un framework, esto es, proporciona una serie de clases y funciones que pueden ser modificadas o extendidas con el objetivo de desarrollar un web crawler propio; Nutch es un programa ya desarrollado, el cual es configurable a través de archivos XML. Este último tiene la ventaja de que la mayor parte de los problemas de crawling ya están

resueltos sin la necesidad de escribir ningún programa, pero la desventaja de que si se quiere realizar alguna tarea adicional se debe escribir un plugin para extender las funcionalidades del mismo.

Utilizando Scrapy se vuelve realmente sencillo escribir un web crawler propio, pero en la experiencia que tuvimos fue difícil lograr que crawleara los sitios web sin entrar en loops debido a URLs mal formadas y a lograr distinguir entre páginas con contenido HTML y con contenido javascipt u otros cuando las páginas no tenían la extensión correcta (.html o .js).

En definitiva, la mayoría de los problemas que encontramos utilizando Scrapy es debido a que los sitios web no siempre están configurados o hechos de una forma "correcta" y si bien Scrapy brinda muchas herramientas a la hora de hacer scraping (que es en lo que realmente está enfocado) del HTML, inclusive de HTML mal formado, en nuestra experiencia requiere un esfuerzo elevado implementar un crawler que funcione en forma adecuada. Por otro lado, Nutch ya trae todos los problemas del crawling prácticamente solucionados y la parte engorrosa fue escribir el plugin para extender su funcionalidad.

Un punto a favor de Nutch es que tiene una documentación mucho más extensa y tiene a sus espaldas una comunidad muy amplia tanto de desarrolladores como de usuarios. Esto es debido a su larga trayectoria como proyecto puesto que existe desde hace más de 10 años mientras que Scrapy es un proyecto más reciente.

Sin embargo en el marco de este proyecto, el principal punto a favor de Nutch reside en una decisión de diseño. La misma es, como se discute más adelante, que tratamos por separado el problema del crawling y del scraping de las páginas web.

Por lo tanto se decidió utilizar Nutch para realizar el trabajo del crawling puesto que se adapta mejor a las necesidades que surgen de la solución propuesta y que además, se tenía alguna experiencia previa en su utilización.

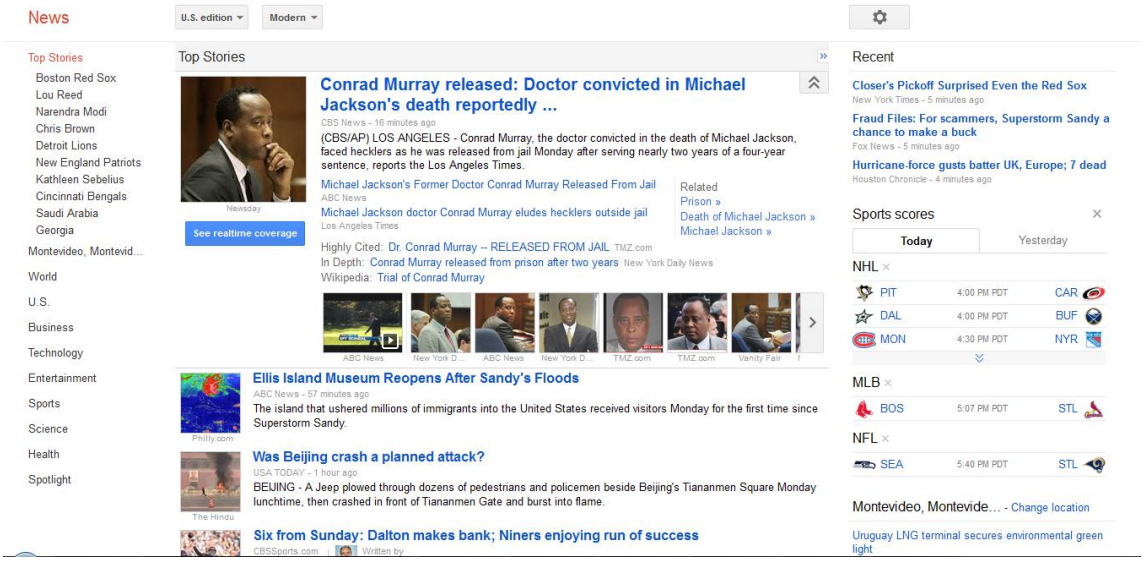
2.4 PROYECTOS SIMILARES

2.4.1 Google News

Nuestro proyecto está altamente relacionado a lo que son los motores de búsqueda, por lo tanto no podemos dejar de mencionar al trabajo que realiza el gigante de internet sobre la búsqueda de noticias. Google News (22) (23) es un agregador (24) y buscador de noticias automatizado que rastrea de forma constante la información de los principales medios de comunicación online (aunque aún no existe una versión para Uruguay).

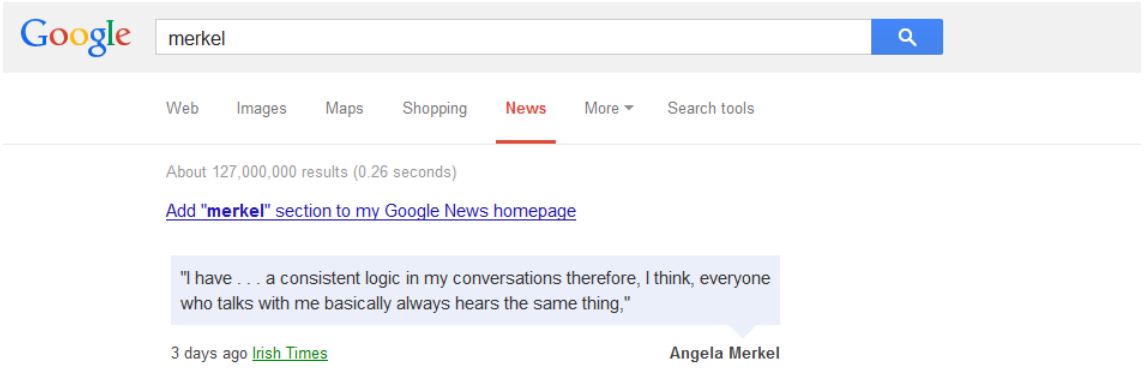
El sitio web de Google News, elaborado por Google Inc., fue lanzado en versión beta en abril de 2002. Existen más de 40 ediciones regionales de Google Noticias en diversos idiomas, entre los que se incluyen el alemán, el árabe, el chino, el coreano, el español, el francés, etc. Cada una de dichas ediciones está adaptada específicamente a los lectores de los respectivos países. Los artículos se seleccionan y se clasifican mediante un sistema informatizado que evalúa, entre otras cosas, la frecuencia y los sitios en los que aparece una noticia.

Si se entra a la página principal de Google News (25) se puede ver una serie de noticias con una imagen relacionada y un breve texto que la describe. Además, se pueden ver varios artículos relacionados a esa misma noticia junto con personas que fueron citadas en los artículos y a veces hasta un link al artículo de wikipedia sobre el tema:



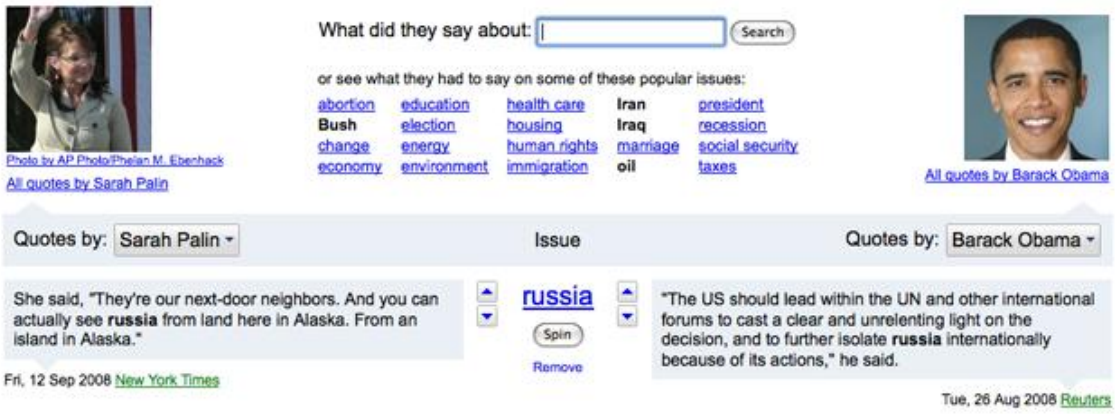
Esto puede verse como algo similar a uno de los objetivos de nuestro proyecto que es el de mostrar cuáles fueron los temas sobre los cuales más se habló en una semana determinada. La diferencia consiste en que Google News muestra los temas sobre los que más se está hablando en el momento pero no da la opción de ver cuáles fueron los temas candentes de días anteriores.

Por otro lado, Google News también permite realizar una búsqueda por el nombre de alguna persona y retorna una cita textual de dicha persona que haya sido encontrada recientemente en un medio de prensa. Sin embargo, esta funcionalidad sólo existe para el idioma inglés y no permite realizar un búsqueda por tema. Por lo tanto no es posible elegir ningún criterio acerca de la cita siendo una herramienta muy limitada.



2.4.2 Google "In Quotes"

En el 2008, con motivo de las elecciones en EE.UU., Google lanzó un proyecto en etapa beta que permitía comparar los dichos (citas textuales) de diferentes políticos acerca de un determinado tema:



Sin embargo, esta herramienta ya no está disponible.

3 CRAWLING DE LOS MEDIOS DE PRENSA

3.1 DISCUSIÓN

Antes de describir la solución propuesta, se analizan y comparan dos enfoques diferentes para poder descargar y procesar las páginas web correspondientes a artículos de prensa:

1- Descargar todas las páginas web del sitio, con el código HTML incluido, sin analizar si se trata de artículos o portadas, guardarlas por fecha de descarga, y luego procesarlas, descartando las páginas que no corresponden a artículos de prensa. En definitiva este enfoque separa las tareas de Crawling y la de Scraping.
2- Ir procesando a medida que se recorre el sitio, de esta forma descartamos todo lo que no sea de interés, el código HTML y JavaScript (de existir este último). Luego guardamos los artículos ya procesados.

Ambos enfoques tienen sus ventajas y desventajas y a continuación los compararemos.

3.1.1 Ventajas del primer enfoque

- Cambios en el formato de los sitios web no hacen que el programa deje de funcionar o funcione incorrectamente.
- Menos código necesario para descargar las páginas, por lo tanto menos probabilidad de introducir errores el proceso de descarga de las páginas.
- Se tienen en disco todas las páginas, sin necesidad de saber cómo procesarlas y de qué información es relevante. Esto nos permite ir almacenando información a lo largo del proceso de desarrollo, con lo cual una decisión incorrecta en el procesado de los datos no implica que sea necesario descargar nuevamente todo el material.

3.1.2 Ventajas del segundo enfoque

- Menos material descargado, lo que implica menos ancho de banda y espacio en disco requerido.
- Se descartan más fácilmente páginas que no corresponden a artículos de prensa.

3.1.3 Solución propuesta

Tomando en cuenta lo anterior se decidió optar por el enfoque número 1. Esto se debe a que los cambios en los formatos de los sitios son demasiado frecuentes, haciendo que los las arañas no funcionen como es debido, o directamente dejen de funcionar. Además en esta etapa del proyecto no se conoce a ciencia cierta cómo se van a utilizar las páginas, pudiendo ser el HTML de utilidad en un futuro, por ejemplo, en caso de que la información extraída no sea la que finalmente se va a utilizar.

Otro motivo para utilizar este enfoque, es el hecho de que el ancho de banda es un cuello de botella en el procesamiento. Si bien en un principio hay que descargar más material, al tener un error no es necesario descargar todo nuevamente. Por lo tanto, considerando que la probabilidad de tener errores de cualquier tipo es extremadamente alta, es casi seguro que con este enfoque se ahorra ancho de banda y tiempo.

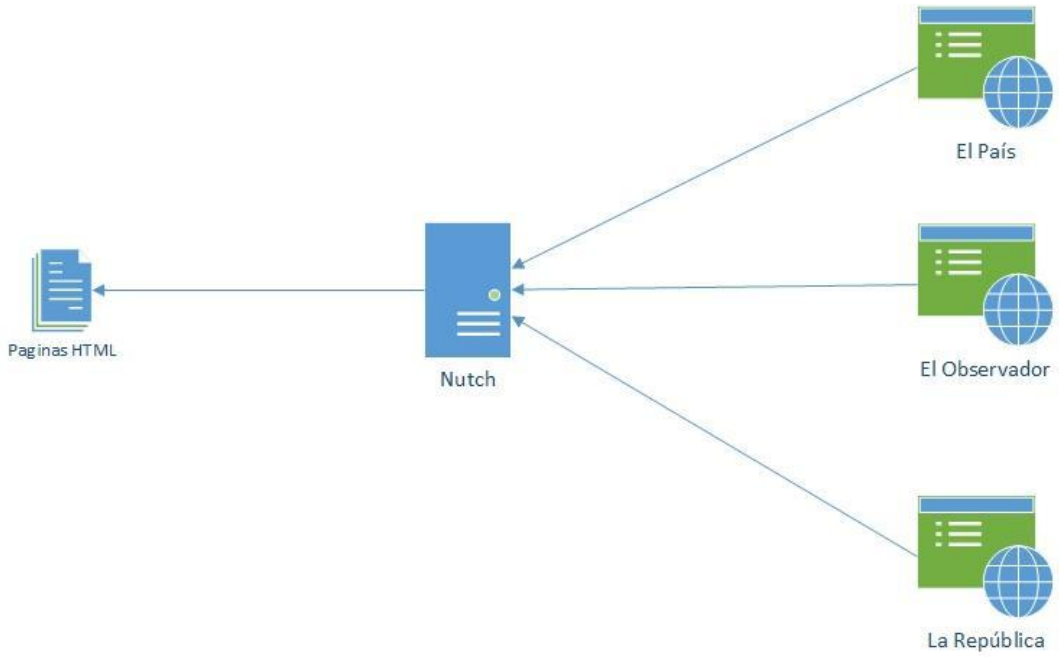
Ahora discutiremos la solución en sí misma. Como se utilizó el enfoque número 1 anteriormente descrito, el problema se divide en dos: Descargar las páginas y por otro lado procesarlas.

Para resolver el problema de *descargar* el sitio se utilizó un plugin para *Nutch* (al cual llamaremos Crawler), y para resolver el problema del procesamiento se creó un programa en Java (al cual llamaremos Scraper).

3.2 EL CRAWLER

3.2.1 Diseño del módulo

El crawler se encarga de descargar todas las páginas web de los sitios elegidos y almacenarlas junto con su URL ordenadas por la fecha de descarga. Para esto se utiliza un plugin escrito para Nutch. Los archivos HTML descargados servirán como entrada para el siguiente módulo, el Scraper.



Para un mejor entendimiento de Nutch se sugiere leer lo descrito en el capítulo 2.3.2, visitar la wiki (5) y en especial, para realizar una rápida configuración de Nutch, leer el tutorial (25). Nosotros en particular utilizamos la versión 1.2 del sistema, aunque entre las distintas versiones no hay grandes diferencias.

3.2.2 Un plugin para Nutch

Nutch está diseñado de una forma altamente modular donde cada funcionalidad se implementa en base a un plugin.

A la hora de crear un plugin, lo que se debe hacer es implementar la interfaz en el *punto de extensión* del cual se esté buscando extender la funcionalidad. En nuestro caso lo que queríamos era lograr que Nutch en vez de procesar las páginas HTML con sus parsers por defecto, las guardara en una carpeta para que luego nosotros pudiésemos procesarlas por nuestra cuenta. Por lo tanto, creamos un plugin que implementa la interfaz *HtmlParseFilter* (26).

Básicamente lo que hace el plugin que escribimos, al cual decidimos llamar *rawhtmlplugin*, es tomar el html que nos pasa Nutch y guardarlo en un archivo cuyo nombre es la URL de la página encodeado en Base64 (27). Este archivo se guarda dentro de una carpeta que tiene

como nombre la fecha en la cual la página fue descargada, la cual a su vez está dentro de una carpeta que tiene como nombre el nombre del medio del cual la página fue descargada.

Todo esto es a los efectos de tener bien organizadas las páginas que descargamos y poder detectar fácilmente si hubo algún error en el crawler mientras estemos haciendo el scraping. Por lo tanto, una ruta de ejemplo para un archivo HTML descargado sería la siguiente:

C:\elpais\2013_07_15\aHR0cDovL2hpc3Rvcmljby5lbHBhaXMuY29tLnV5L2NhcnRlbGVyYXMvRXNwZWN0YWN1bG9zLmFzcA==

3.2.3 Crawling con Nutch

El proceso que utiliza nutch para realizar el crawling de los sitios es el siguiente:

1. A través de los archivos de configuración XML, se le debe indicar a Nutch cuáles serán los patrones de URL a crawlear. Esto se hace mediante expresiones regulares.
2. Se debe insertar una "semilla" que será la primer página que visitará el crawler. En nuestro caso utilizamos como semillas las páginas Home de cada medio, donde está la portada principal.
3. Cada página que se visita, se recorre en busca de links (que están dentro del tag <a> de HTML) y los mismos se filtran según los archivos descritos en el paso 1 para posteriormente guardarlos en una base de datos.
4. Para elegir cuál será la siguiente página a visitar, se selecciona uno de los links que están guardados en la base de datos. Luego de visitada la página, se guarda, para dicho link, la fecha en la que se visitó. Esto último se hace en caso de que se desee luego poder re-crawlear páginas que anteriormente ya fueron visitadas por si se presentan cambios en las mismas. En nuestro caso esto es importante para poder crawlear varias veces la portada del diario, donde se colocan links a todas las noticias del día.

Aquí se puede encontrar una descripción más detallada de este proceso: (20)

3.2.4 Problemas encontrados

Al realizar la implementación de este módulo se encontraron las siguientes dificultades:

- Al descargar las páginas web, nos encontramos con el clásico problema del encoding (el cual se repitió a lo largo de todo nuestro proyecto) ya que cada sitio usa un encoding propio (UTF-8, ISO-8859-1, etc) mientras que los String de Java están en formato Unicode. Debido a esto, se debe tener el cuidado cuando se descarga el HTML de la página de hacer la conversión correcta entre diferentes encodings. Esto puede parecer sencillo, pero siempre son este tipo de problemas los que llevan más tiempo de resolver.
- Descargar todas las páginas de los distintos medios es un proceso que lleva mucho tiempo y se debe dejar corriendo un proceso durante varios meses de forma continua para obtener toda la información. Además, para tener los datos de forma actualizada (o sea, tener noticias recientes) se debe correr periódicamente dicho proceso.
- Si bien la documentación de Nutch en general es bastante buena (como en todos los proyectos de Apache), la documentación para escribir un plugin no lo es, especialmente por la falta de códigos de ejemplo.

4 CREACIÓN DE LA BASE DE TEXTO

4.1 INTRODUCCIÓN

Con el objetivo de poder obtener una base de texto con opiniones extraídas de las páginas web descargadas como se describió en el capítulo anterior, se debe realizar un procesamiento del HTML. Para esto se desarrolló un programa al cual llamamos Scraper.

Más adelante en este capítulo daremos una descripción acerca del volumen de datos obtenidos para nuestra base de texto así como algunos problemas que tuvimos con los diversos metadatos extraídos de los sitios web.

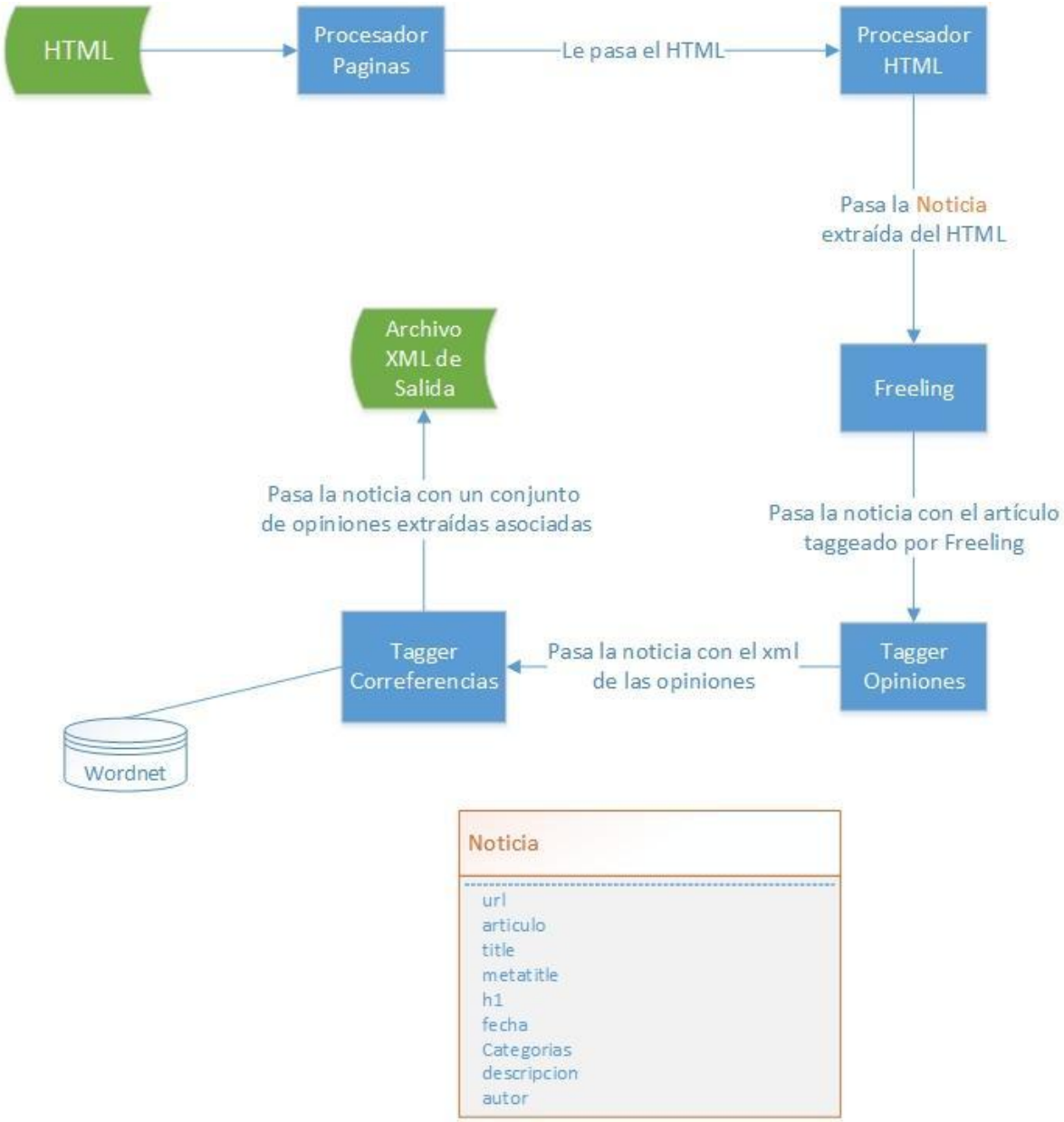
4.2 EL SCRAPER

4.2.1 Diseño del módulo

Este módulo fue implementado completamente en Java. El scraper toma como entrada cada uno de los archivos HTML descargados por el crawler y los procesa para extraer las opiniones de los textos. Para realizar esto, el módulo debe implementar las siguientes funcionalidades:

1. Distinguir entre: a) las páginas HTML descargadas que son portadas, blogs, rss, etc. que se encuentran dentro del dominio del medio y b) las páginas que son realmente artículos periodísticos. Las páginas que no son artículos periodísticos deben descartarse.
2. Extraer del HTML los diferentes atributos de los artículos periodísticos (título, el artículo de prensa en sí mismo, etc), descartando los elementos de la página que no sean importantes como el cabezal, publicidad, etc.
3. Procesar el artículo de prensa en sí mismo (extraído en la parte anterior) para extraer las expresiones de opinión del mismo. Para esto utilizaremos el módulo de extracción de opiniones descrito en el capítulo 2.1 previamente procesando el artículo con freeling.
4. Procesar la salida del módulo de extracción de opiniones para encontrar las correferencias entre las fuentes utilizando el módulo descrito en el capítulo 2.1.
5. Generar un XML con la información extraída para que pueda ser indexada por el motor de búsqueda.

A continuación se presenta un diagrama de flujo sobre cómo se procesan los datos que llegan al sistema:



4.2.2 Filtrado de artículos de prensa

4.2.2.1 Introducción

Dado que se decidió descargar todo el contenido de la web antes de hacer algún procesamiento de los datos, nos enfrentamos a la dificultad adicional de separar las páginas que corresponden a artículos de prensa, y las que son solo portadas o que cumplen alguna otra funcionalidad. Para los distintos medios de prensa se analiza cómo resolver este problema.

4.2.2.2 Análisis de distintos medios de prensa

Antes de resolver el problema, se analizan las características de cada medio de prensa.

4.2.2.2.1 El Observador

Para este medio de prensa resultó realmente sencillo, ya que todas las páginas que son artículos de prensa contienen la palabra “noticia” en la URL, por lo tanto para filtrar simplemente se descartan páginas web cuya URL no contenga dicha palabra. Este método no sólo descarta portadas, publicidad y aplicaciones, sino que también los blogs.

4.2.2.2.2 La República

Para La República el problema es igual de sencillo, dado que sólo se tienen unas pocas páginas que no corresponden a noticias, se puede hacer un listado y filtrarlas manualmente.

4.2.2.2.3 El País

Dado que para este medio no se puede deducir ninguna heurística sencilla y que tiene muchas portadas y páginas que no son artículos de prensa, se recurre a un método más complejo. Por lo tanto, se decidió utilizar un método de aprendizaje automático, con el uso de la biblioteca Weka (28). Para ello se guardó la información correspondiente a páginas web en formato CSV (comma-separated values), y luego se generó un clasificador a partir de dichos datos.

4.2.2.3 Resolución para El País

4.2.2.3.1 Conjunto de entrenamiento

Para poder tener datos sobre los cuales trabajar fue necesario descargar y clasificar 224 páginas para el país. Las mismas fueron clasificadas en “true” si es un artículo y en “false” en caso de páginas correspondientes a portadas y/o publicidad. Para esto se utilizó un módulo que se encarga de descargar la página web y almacenarla en un archivo con nombre igual a la codificación en BASE64 de su URL. Por otra parte se creó una tabla en un archivo llamado “resultados.csv”, donde a cada nombre de archivo se le agrega el resultado de “true” o “false”, según como se halla clasificado manualmente.

4.2.2.3.2 Extracción de características

Se eligieron las características que mejor parecieron representar los datos (en base solamente al código HTML). Las características elegidas para el diario El País son las siguientes: Tamaño del archivo HTML en bytes, largo de la URL, cantidad de tags H1, H2, H3, H4, H5, cantidad de tags TABLE, cantidad de tags DIV, cantidad de tags P y por último la cantidad total de tags (sin contar los de formato B y P).

Elegimos estos atributos debido a que en una primera visualización de los datos intuimos que en general las páginas que no son artículos periodísticos, contienen más datos y por lo tanto son más pesadas en bytes. Por otro lado, los artículos periodísticos en general tienen menos etiquetas que resaltan al texto como H1, H2, etc. Y, de forma más general, pudimos ver que la cantidad de tags que se utiliza en una portada es distinto a la cantidad utilizada en los artículos.

La clasificación de las páginas se realiza en dos clases, “true” si la página es un artículo y “false” en caso contrario.

4.2.2.3.3 Resultados obtenidos por distintos algoritmos

En esta parte se probaron varios algoritmos de aprendizaje automático supervisado. Para construir el clasificador se utilizó Weka como biblioteca y para verificar los resultados se utilizó validación cruzada. Los clasificadores que se utilizaron fueron los siguientes: C4.5, el mismo se basa en árboles de decisión es una extensión del algoritmo ID3 ; Clasificador Bayesiano Simple; 1NN, es el algoritmo de los K-Vecinos pero tomando en cuenta solo el vecino más cercano.

A continuación se muestra un cuadro de resultados con los resultados de los distintos clasificadores, los mismos se evaluaron con validación cruzada de diez iteraciones.

Algoritmo	Porcentaje de acierto
Bayesiano	82.5893 %
C4.5	95.5357 %
1NN	97.3214 %

Por lo tanto a la vista de estos resultados se optó por el clasificador 1NN. No se tomaron en cuenta algoritmos más complejos dado que a partir de estos se obtuvo muy buenos resultados.

A continuación se muestra la matriz de confusión del clasificador 1-NN:

	false	true
false	82	6
true	0	136

Se puede observar que se logra filtrar el 92,68% de las páginas que no son artículos de prensa y por el otro lado se conserva el 100% de los artículos periodísticos sin descartar ninguno.

4.2.3 Elementos a extraer de los artículos de prensa

4.2.3.1 Descripción

Con el objetivo de mejorar las búsquedas sobre las opiniones en noticias, se trata de extraer la mayor cantidad de información posible de los artículos de prensa. Cada uno de estos elementos o características aparecen de una forma distinta en cada página web, e incluso tiene varios formatos dependiendo de la fecha en que se creó el artículo, por lo tanto es un desafío de crear algoritmos que puedan extraerlos en forma correcta en la mayoría de los casos. Las siguientes son las características elegidas:

- **Título -** Es el título de la noticia, muchas veces está entre tags HTML "<h1>" y tiene una presencia destacada.
- **Subtítulo -** En general las noticias tienen un texto corto que agrega información al título. Por ejemplo, la noticia con título "Timerman salió al cruce de dichos de Danilo Astori" publicada por El País el 08/05/13 tiene como subtítulo "Canciller Argentino".
- **Copete -** Es un resumen de la noticia que por lo general aparece debajo del título y con un destacado intermedio.
- **Fecha de Publicación -** Es la fecha en la que se publicó la noticia. No en todas las páginas aparece con el mismo formato ni en el mismo lugar. Para un programa que

reconoce las fechas automáticamente es difícil distinguir la fecha de publicación de alguna otra fecha mencionada en la noticia o en el boilerplate de la página.

- **Categoría -** Las noticias en general están categorizadas, cada medio utiliza distintas categorías. Por ejemplo: Música, Mundo, Economía, etc.
- **Etiquetas(o tags)-** Algunos medios colocan etiquetas a sus artículos que son palabras claves que describen el contenido de la noticia.
- **Artículo -** Es la noticia en sí misma
- **Noticias relacionadas -** Prácticamente todas las versiones en línea de los medios ofrecen, dada una noticia, varias noticias que están relacionadas. A veces simplemente se muestran otras noticias en la misma categoría pero también pueden usar métodos más sofisticados.
- **Autor -** En algunos casos como por ejemplo en columnas y editoriales, es interesante saber el autor del artículo, sin embargo, no siempre se puede obtener el mismo.

4.2.3.2 Implementación

Esta fue una de las partes más complejas de implementar del sistema puesto que todas las páginas web de noticias utilizan distintas formas de distinguir los distintos atributos como el título, etc. Además, para un mismo sitio web, pueden utilizarse distintas formas de realizar la distinción entre estos elementos. Por ejemplo, la página web de El País existe desde el año 2002 y a lo largo de su historia a sufrido diversas actualizaciones y modificaciones; sin embargo, si se va a leer una noticia del 2002, se verá la interfaz de la página como era en aquella época y si se visita una noticia actual, se verá la interfaz como es en la actualidad. Esto implica que dentro del mismo sitio web existen decenas de formas en las que aparecen los elementos del artículo de prensa.

Por otro lado, hay cambios frecuentes en los formatos de las páginas web de los medios de prensa, lo cual trae la dificultad de adaptar el programa para que se puedan procesar estos nuevos formatos. Por lo tanto, será necesario revisar periódicamente en busca de cambios y se tratará de diseñar algoritmos de extracción lo más generales posibles para mitigar este problema.

Todo esto hace que haya que revisar a mano las diferentes formas en las que aparece el html en las diversas páginas para crear un algoritmo que extraiga los elementos que definimos anteriormente.

Para implementar la extracción de de los diferentes elementos nos planteamos dos opciones diferentes: utilizar XPath o expresiones regulares.

XPath posee la ventaja de que está especialmente pensado para la extracción de información sobre XML ó XHTML (HTML bien formado). Sin embargo, surge el problema de que la mayoría de las páginas con las que nos encontramos están mal formadas, teniendo etiquetas que abren pero nunca cierran o otro tipo de anomalías. Para solucionar esto experimentamos con la utilización de herramientas que transforman en forma automática el HTML mal formado en uno bien formado como *HTMLCleaner* (10).

Sin embargo, ni siquiera de esta forma obtuvimos buenos resultados pues utilizar XPath implicaba tener que escribir una expresión para cada una de las posibles formas en las que apareciera la información. Por lo tanto, finalmente decidimos utilizar una herramienta más versátil como las expresiones regulares.

4.2.3.3 Extracción de la fecha

La extracción de la fecha, si bien también se realizó mediante expresiones regulares, merece un apartado especial. Para nuestro sistema es fundamental que la fecha esté extraída de forma correcta puesto que luego las opiniones se ordenarán según la fecha en la que se hayan emitido.

Cada medio utiliza una forma distinta de mostrar la fecha en la cual se publicó la noticia y dentro de un mismo medio de prensa se utilizan varias formas distintas de mostrarla. Además, dentro de una página pueden aparecer distintas fechas tanto dentro de la noticia como en el boilerplate y se debe tener especial cuidado de tomar la fecha correcta. Por ejemplo, el cabezal de la página de El País incluye la fecha del día actual, que es distinta al dia en el que se publicó la noticia y existen algunas páginas que tienen la fecha dentro de la url.

A continuación mostramos algunas de las expresiones regulares que utilizamos para la extracción de la fecha:

(?i)([0-1][0-9]).?([0-1][0-9]).?([0-3][0-9])
(?i)(20[0-1][0-9]).?([0-1][0-9]).?([0-3][0-9])
(?i)([0-3]?[0-
9]).de.(enero|febrero|marzo|abril|mayo|junio|julio|agosto|septiembre|octubre|noviembre
|diciembre)(.de|,| ,).?(20[0-1][0-9])
(?i)([0-3][0-9]).?([0-1]?[0-9]).?(20[0-1][0-9])
(?i)([0-3][0-9]).?([0-1][0-9]).?([0-1][0-9])
(?i)([0-3][0-9])\\.([0-1][0-9])\\.(20[0-1][0-9])
(?i)Publicado el ([0-3]?[0-9])/([0-1]?[0-9])/(20[0-1][0-9])

4.2.3.4 Extracción del artículo

El artículo, que configura a la noticia en sí misma, es el único atributo que no extrajimos mediante la utilización de expresiones regulares. Si bien en un principio también pensamos en implementarlo mediante expresiones regulares, encontramos que existe un algoritmo que puede extraer la información principal de dentro de una página (en este caso el artículo periodístico) descartando todo lo que es accesorio, también llamado el boilerplate.

El funcionamiento del algoritmo es el descrito en el paper (12) de Christian Kohlschütter. A grandes rasgos, lo que hace el algoritmo es ir contando los tags HTML que se van abriendo y cerrando determinando cuáles son los textos que aparecen más anidados dentro de estos tags. Por ejemplo, una página de ejemplo puede ser la siguiente:

```
<html>
    <cabezal>
        <titulo>
            titulo
        </titulo>
    <cabezal>
    <cuerpo>
        <noticia>
            texto muy largo
        </noticia>
    </cuerpo>
    <publicidad>
        coca-cola
    </publicidad>
</html>
```

Se puede ver que tanto la noticia como el título están al mismo nivel de profundidad dentro del árbol, pero el texto de la noticia es más largo y por lo tanto se considera que este es el contenido principal.

Si bien el algoritmo descrito en el paper hace algo bastante más complejo, este ejemplo describe el funcionamiento básico.

4.2.3.5 Bibliotecas utilizadas

A la hora de escribir el scraper para procesar el HTML y extraer las características que nos interesan, se utilizaron varias bibliotecas que nos facilitan esta tarea.

- Jsoup (29)- Proporciona una API muy conveniente para la extracción y manipulación de datos desde HTML, utilizando métodos DOM, CSS, jQuery y similar.
- HTMLCleaner (30)- Es una herramienta que nos permite trabajar fácilmente con HTML mal formado.
- BoilerPipe (31)- Esta librería proporciona algoritmos para detectar y eliminar el excedente (boilerplate) en torno al principal contenido de texto de una página web. Están especialmente diseñados para trabajar sobre páginas con artículos de prensa y blogs donde el contenido principal de la página es un texto medianamente largo. Para entender cómo funciona este algoritmo es recomendable leer el paper (32) de Christian Kohlschütter quien desarrolló el mismo.

4.2.3.6 Problemas de extracción

Como comentamos anteriormente, esta fue una de las partes más complejas de implementar del sistema debido al gran trabajo manual que la misma requirió. Intentamos implementar esta parte para que fuese lo más general posible aunque no siempre lo logramos. Es importante que estos algoritmos de extracción de características sean reutilizables ya que pensando en el futuro de este proyecto, sería bueno poder agregar nuevos medios de prensa con relativamente poco esfuerzo.

El mayor problema que tuvimos es que no todas las noticias utilizan el mismo estilo y muchas veces no todos los atributos están presentes. Por ejemplo, en El País, la mayoría de las noticias no incluyen quién fue el periodista autor de las mismas. Por otro lado, todos los diarios utilizan diferentes categorías para describir a sus noticias, muchas de estas incompatibles entre sí. Un caso especial es, nuevamente, el de El País, donde a lo largo del tiempo fueron cambiando las categorías utilizadas y se da la situación de que en el 2012 utilizaban un sistema de categorías que es incompatible con las utilizadas para el año 2013.

A su vez, hay categorías que son muy poco descriptivas o demasiado generales. Por ejemplo, en El País existe una categoría llamada “información”. El Observador tiene una categoría llamada “estilo” donde se pueden encontrar noticias tanto nacionales como internacionales sin que podamos haber encontrado exactamente qué criterio utilizan para clasificarlas, ya que también tienen categorías llamadas “nacional” y “mundo”.

Por otro lado, en La República es imposible conocer la categoría de una noticia desde la página de la noticia en sí misma. Sin embargo, tiene distintas portadas para las distintas categorías.

Con respecto a las noticias relacionadas, el único medio de los analizados que implementa un sistema de sugerencias de noticias realmente relevantes es El Observador. Los otros dos si bien realizan sugerencias al lector sobre otras noticias para leer, las mismas no se basan en qué tan

relacionadas estén a la noticia que se está leyendo, sino que se sugieren las noticias más leídas del día o criterios similares.

4.2.4 Utilización de los proyectos anteriores dentro del Scraper

4.2.4.1 Descripción

Los módulos correspondientes al tagger de opiniones y al tagger de correferencias se utilizaron conectados entre sí con el objetivo de procesar los artículos de prensa y extraer las expresiones de opinión de los mismos.

Primero se procesa el html del artículo con nuestro scraper, luego se procesa la salida obtenida con Freeling para luego pasarla al módulo de identificación de opiniones y a su vez, se pasa la salida de este último al módulo de resolución de correferencias.

De la salida del módulo de correferencias se utiliza un xml (salidaFinal.xml) en el cual figuran las opiniones encontradas en el texto con sus respectivas fuentes. También figuran las fuentes reconocidas por el módulo que no habían sido reconocidas en un principio por el módulo de identificación de opiniones. Por otro lado, también se incluye la correlación entre las distintas fuentes encontradas. Por ejemplo se correlaciona “Mujica” con “El presidente” y “José Mujica”.

4.2.4.2 Problemas encontrados con la utilización de los módulos

A la hora de utilizar los módulos anteriormente mencionados, se tuvieron varios problemas. Esto era esperable ya que son proyectos de estudio científico y no pensados para que los utilice un usuario final que no participó de la elaboración de los mismos. A continuación detallamos algunos de los obstáculos con los que nos encontramos:

- El módulo de identificación de opiniones está desarrollado en Prolog y el de correferencias en Python. Nosotros utilizamos Java para desarrollar nuestro programa y por lo tanto tuvimos que conectar Java con los otros lenguajes. En principio pensamos en utilizar librerías específicas para poder ejecutar ambos lenguajes en Java. Para Python existe una librería llamada Jython (33) que permite la ejecución de código Python dentro de Java. El problema que nos encontramos con esta librería es que no pudimos hacer que nuestro programa se conecte a una base de datos MySql a través de Python, algo requerido por el módulo de correferencias para su correcto funcionamiento. En el caso de Prolog, intentamos utilizar la librería JPL (34) para Java que viene incluida en la instalación de SWI-Prolog. El problema que nos encontramos con esta librería es que no permite resetear la máquina virtual donde se ejecuta el código de prolog y el proyecto de identificación de opiniones posee una limitación que consiste en que no se puede ejecutar dos veces seguidas el módulo en el mismo intérprete de Prolog (el mismo debe reiniciarse primero).
  En conclusión, no pudimos utilizar ninguna de las dos librerías y terminamos utilizando la clase *ProcessBuilder* de Java que permite la ejecución de procesos externos a la aplicación, para de esta forma ejecutar Python y Prolog externamente desde la instalación en la propia máquina y obtener la salida que generan los programas.
- El módulo de identificación de opiniones genera como salida un xml mal formado, pero el módulo de correferencias espera un xml estándar como entrada. Para solucionar este problema tuvimos que utilizar la librería HTMLCleaner que si bien en

principio la habíamos utilizado para limpiar el HTML mal formado de las páginas descargadas de internet, también sirvió para arreglar el XML mal formado.

- En el módulo de correferencias se encontraron varios bugs que hacían que el programa diese un error en tiempo de ejecución. Para solucionar esto tuvimos que ponernos a revisar el código Python en el que está escrito el proyecto teniendo en cuenta que no teníamos ninguna clase de experiencia con este lenguaje.
- Para poder utilizar el módulo de identificación de opiniones se tiene primero que pasar el texto por el tagger Freeling (35). El problema que tuvimos fue que el módulo estaba desarrollado con una versión de Freeling más antigua a la que se puede descargar actualmente y tuvimos que hacer un procesamiento de la salida del mismo para que coincidiera con la entrada que espera el módulo.
- Los dos módulos junto con Freeling utilizan archivos para escribir su salida y volvimos a tener los clásicos problemas de encoding que derivan de esto. La salida de Freeling es un archivo en UTF-8 pero el módulo de identificación de opiniones espera como entrada un archivo en el formato Windows-1252. A su vez, este módulo genera un archivo de salida en ese mismo formato, pero el módulo de correferencias espera como entrada un archivo en UTF-8.

4.2.5 Archivo de salida del Scraper

Luego de procesar un artículo, el Scraper guarda cada una de las opiniones en un archivo xml de salida, el cual servirá para luego indexar los datos en Solr. Aquí se puede ver cómo a partir de una página web, se obtiene un xml con los datos de la misma:


```
<add>
<doc>
    <field name="url">http://historico.elpais.com.uy/101125/pinter-
531027/internacional/EE-UU-se-moviliza-tras-escalada-belica-en-Corea-
y-da-apoyo-a-Seul/</field>
```

```
    <field name="articulo">Internacional EE.UU. se moviliza tras
escalada bélica en Corea ...</field>
    <field name="title">EE.UU. se moviliza tras escalada bélica en
Corea y da apoyo a Seúl - Diario EL PAIS - Montevideo -
Uruguay</field>
    <field name="metatitle">Crisis. Dos civiles murieron en la isla
atacada por norcoreanos Surcoreanos y estadounidenses realizarán
ejercicios ..,Hallaron los cadáveres de dos isleños muertos por el
ataque de artillería norcoreano a una isla de Corea del Sur, que
..</field>
    <field name="h1">EE.UU. se moviliza tras escalada bélica en Corea
y da apoyo a Seúl</field>
    <field name="fecha">2010-11-25T00:00:00Z</field>
    <field name="categorias"></field>
    <field name="descripcion">Crisis. Dos civiles murieron en la isla
atacada por norcoreanos Surcoreanos y estadounidenses realizarán
ejercicios ..,Hallaron los cadáveres de dos isleños muertos por el
ataque de artillería norcoreano a una isla de Corea del Sur, que
..</field>
    <field name="autor"></field>
    <field name="fuente">La radio estatal</field>
    <field name="fuente_corref"></field>
    <field name="opinion">La radio estatal señaló que &quot; nuestra
artillería no está todavía tranquila &quot;</field>
    <field name="id">http://historico.elpais.com.uy/101125/pinter-
531027/internacional/EE-UU-se-moviliza-tras-escalada-belica-en-Corea-
y-da-apoyo-a-Seul//20</field>
</doc>
<doc>
    …
</doc>
</add>
```

Cada elemento `doc` del xml es lo que luego Solr va a considerar como un documento, que en este caso corresponde a una opinión. Cada uno de los `field` corresponde a un campo definido en el schema de Solr, sobre el cual hablaremos más adelante.

Como se procesan cientos de miles de noticias de las cuales se extraen varias opiniones, el archivo contiene muchas entradas y pesa varios gigas. A su vez, considerando la enorme cantidad de información que se está procesando, el Scraper puede demorar mucho tiempo en terminar de correr. Por esta razón, se decidió que una vez que se pone a correr el programa este va guardando en un archivo cuál es la noticia que se está procesando para cada medio de prensa. Si se corta la ejecución en algún momento, cuando se vuelva a correr, el programa se continúa ejecutando desde la última noticia procesada para cada medio de prensa.

4.3 BASE DE TEXTO GENERADA

4.3.1 Descripción

A partir de los diferentes elementos extraídos de los artículos de prensa que fueron mencionados anteriormente, se logró generar un corpus de opiniones extraídas de los distintos artículos de prensa. Sin embargo, se determinó que existían algunos elementos que no eran realmente relevantes para realizar las búsquedas de opiniones y por otro lado, se agregó algunos atributos nuevos que nos pueden ayudar en las mismas.

Los atributos que finalmente componen la base de texto generada son:

ID	Es artificial creado por nosotros al concatenar la url con un número, para darle la identidad a la opinión (la url no se puede utilizar pues de un mismo artículo se puede extraer más de una opinión)
URL	La url de la noticia original desde la cual se extrajo la opinión.
Artículo	El texto completo del artículo de prensa del que es extraída la opinión.
Title	El título del artículo.
Metatitle	Es una meta etiqueta que funciona algunas veces como subtítulo, aportando alguna información.
H1	Lo que se encuentra dentro de las etiquetas H1 del HTML descargado.
Categorías	Las categorías a las que (según el medio de prensa) pertenece el artículo.
Descripción	Una breve descripción del artículo.
Autor	El autor del artículo, nótese que difiere de la fuente de la opinión.
Fuente	La fuente de la opinión, extraída a partir del Módulo de Identificación de Opiniones.
Fuentes Correfenciadas	Una lista que contiene todas las fuentes que están en la misma cadena de correferencias que la fuente de la opinión. La cadena de correferencias es la generada por el Módulo de Correferencias.
Opinión	La opinión en sí misma, extraída a partir del Módulo de Identificación de Opiniones.
Fecha	La fecha en la que fue publicado el artículo.

4.3.2 Dimensiones de la base

Al momento de escribir este informe, se tienen descargadas más de 570 mil páginas de las cuales más de 400 mil son artículos de prensa (170 mil son portadas u otras páginas de los diarios). Las páginas descargadas ocupan un espacio alrededor de los 50 GB y se tardó unos 6 meses en descargarlas. Este intervalo de tiempo fue tan extenso debido a que no dispusimos de un servidor dedicado donde pudiésemos dejar corriendo el programa, teniendo que utilizar una máquina que sólo corría por las noches.

Por otro lado, el procesamiento de las 570 mil páginas para la extracción de sus datos también es algo que lleva una cantidad considerable de tiempo. En nuestro caso, y repartiendo el trabajo entre varias computadoras, se necesitaron 2 semanas para procesar todos los archivos HTML.

A su vez, midiendo la cantidad de opiniones extraídas de los diferentes artículos de prensa, se obtuvieron 1.865.037 opiniones.

En la práctica se verificó que fue una decisión acertada la de separar el problema del crawling de las páginas web con la del procesamiento de su contenido, ya que a medida que fuimos desarrollando el sistema, fuimos encontrando errores que fuimos corrigiendo en el Scraper. Si hubiésemos integrado los dos módulos (el Crawler y el Scraper), cada vez que encontrábamos un error, hubiésemos tenido que descargar nuevamente todas las páginas, aumentando exponencialmente el tiempo que nos tomó descargar las páginas, que ya de por sí fue bastante alto.

5 BUSCADOR DE OPINIONES

5.1 INTRODUCCIÓN

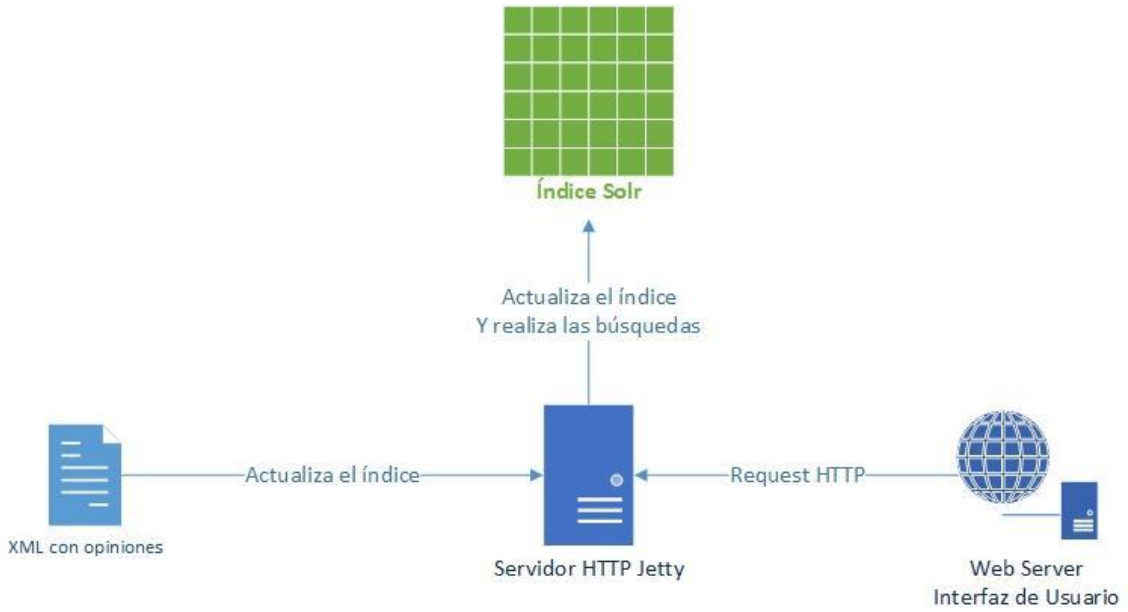
Como se vio en el análisis del problema, es necesario poder realizar búsquedas rápidas y eficientes para la recuperación de las opiniones. El camino elegido para solucionar este problema es utilizar herramientas de búsqueda de texto completo. Para esto se recurrió al motor de búsqueda Apache Solr, que ya describimos en el capítulo 2.2.3.

Cada opinión se puede ver como un conjunto de atributos de texto y fechas. Entre los atributos destacan la opinión propiamente dicha, los títulos y el resto del artículo (los campos son los mencionados en el análisis del problema). Las opiniones así definidas serán los "documentos" que se guarden en el índice de Solr y sobre los cuales luego se realizarán las búsquedas.

Cabe destacar que, finalmente las correferencias extraídas mediante el Módulo de Correferencias, no fueron utilizadas en las búsquedas debido a que introducían demasiados errores en las búsquedas, disminuyendo drásticamente la efectividad del sistema. La única fuente utilizada al realizar una búsqueda es la detectada por el Módulo de Identificación de Opiniones.

5.2 DISEÑO DEL MÓDULO

La salida generada por el Scraper se utiliza para llenar el índice de Solr, donde cada elemento es una opinión junto con todos sus atributos. El buscador de opiniones funciona con una arquitectura cliente-servidor donde la respuesta a las consultas HTTP es un xml que luego será procesado por la interfaz de usuario para una vista más "amigable" del mismo.



5.3 CONFIGURACIÓN DE SOLR PARA EL BUSCADOR DE OPINIONES

Para poder hacer funcionar el motor de búsqueda se partió de uno de los ejemplos que vienen por defecto con la distribución de Solr, modificándose el esquema schema.xml para definir los campos sobre los cuales vamos a realizar las búsquedas.

El archivo schema.xml, volviendo a la comparación de Solr con un RDBMS vista en el capítulo 2.2.4, puede verse como la definición de la única tabla que compone al índice. En este archivo se definen tanto los campos (o columnas) que componen la tabla así como los tipos de datos de dichos campos.

Los elementos que definimos formarán parte de este schema son los definidos en el capítulo 4.3.1. En el capítulo 1 del anexo de Detalles de Implementación, puede encontrarse un Extracto del XML con una explicación detallada sobre cómo fue configurado.

El procesamiento que se realiza sobre los diferentes campos de texto para ser indizados por Solr es el siguiente:

1. Se tokenizan las palabras, dejando de lado signos de puntuación, espacios, etc. Esto se hace ya que lo que se quiere guardar en el índice son palabras, como se describió en el capítulo 2.2.5.
2. Se pasan todas las palabras a minúsculas. Esto se hace con el objetivo de no diferenciar la forma en las que aparecen escritas las palabras. Luego en las búsquedas se tomará como equivalente la búsqueda “Educación” y “educación”.
3. Se quitan las stopwords. Las stopwords son palabras que no serán relevantes a la hora de realizar las búsquedas. Son palabras como “la”, “el”, etc.
4. Se procesan las palabras con *Stemming* para obtener el término raíz. Palabras como “educación” y “educó” pasan a ser “educ”.
5. Se quita cualquier caracter que no pertenezca al encoding ASCII (ASCII tiene 128 caracteres). Es es para evitar problemas con posibles caracteres “extraños” que pueden aparecer en el texto original. Las letras con tildes se traducen a sus respectivas sin tildes, por ejemplo “josé” a “jose”.

5.4 HERRAMIENTA EDISMAX

5.4.1 Descripción

EDismax (36) es un query parser. Un query parser se encarga de transformar una cadena de caracteres en un objeto interno de Lucene, además nos permiten utilizar parámetros adicionales que son locales al string de la consulta. Por defecto Solr utiliza un parser homónimo a Lucene.

El cambiar el parser por defecto permite utilizar una sintaxis especial cuando se realiza una consulta a Solr, esta sintaxis resulta ser, en muchas ocasiones, muy útil y versátil.

Es importante mencionar el tipo de consulta *DisjunctionMaxQuery* que forma parte de Lucene. Este tipo de consulta nos permite buscar en distintos campos asignandoles distinta relevancia a cada uno; por ejemplo, podríamos decir que si una palabra aparece en el título de una noticia, es más relevante que si aparece dentro del artículo. Este tipo de consulta en general genera mejores resultados cuando buscamos en muchos campos el mismo término.

5.4.2 Principales características de EDismax

- Busca en diferentes campos según diferentes pesos (asignados manualmente) utilizando la consulta *DisjunctionMaxQuery* de Lucene.
- Da más puntuación a las frases automáticamente. Por ejemplo, si estamos realizando la búsqueda "conflicto de la salud" y en un artículo aparece exactamente esa frase, tendrá más relevancia ese documento que uno donde las palabras también aparecen pero en otro orden, como podría ser "existe un conflicto cuando se habla de la salud del presidente".
- Tiene su propia forma de asignar diferentes pesos a los parámetros de la consulta. Esto resulta útil cuando trabajamos con *function queries*.
- Se puede especificar un número mínimo de palabras a coincidir con los términos de búsqueda, que pueden depender de la cantidad de palabras en la consulta.
- En el caso de que la consulta sea vacía, se puede especificar una consulta por defecto.

5.4.3 Ventajas del parser EDismax frente al parser por defecto

El parser por defecto exige una sintaxis rígida, con paréntesis y comillas bien balanceadas; sin embargo es común que se quiera pasar los parámetros de la búsqueda de una forma más sencilla. En particular nos interesa que la relevancia sea la adecuada dependiendo de cantidad de palabras que coincidieron con los términos de búsqueda, para esto utilizaremos el parámetro *mm* de el parser EDismax que nos brinda dicha funcionalidad.

5.4.4 Parámetros EDismax

- mm o *Min-should-match*, este parámetro nos permite decirle al parser cuántas palabras de la consulta queremos que coincidan para recuperar un documento, es decir cuántas son opcionales y cuantas son obligatorias. Por ejemplo si buscamos "milanesa en dos panes" (suponiendo que no tenemos stopwords) y mm=2, vamos a recuperar todos los documentos que contengan al menos dos de las palabras, es decir dos palabras son obligatorias. En este ejemplo vamos a recuperar un documento que tenga las palabras "milanesa" y "panes", pero no un documento que tenga solo la palabra "milanesa". También podemos poner un número negativo que lo que nos dice es cuantas palabras son opcionales (la cantidad restante van a ser obligatorias), por ejemplo si mm=-3 tendríamos que con solo una palabra es necesaria, es decir que una consulta va a recuperar documentos con tan solo la palabra "milanesa". Además se pueden definir porcentajes que pueden ser tanto positivos como negativos (porcentaje de obligatorias y porcentaje de opcionales) redondeados hacia abajo. Aquí es importante aclarar que hay una diferencia entre 50% y -50%, podríamos pensar que es lo mismo pero el hecho de el redondeo hace que sea distinto, por ejemplo si tenemos 3 palabras en el primer caso vamos a tener una obligatoria y dos opcionales, en el segundo vamos a tener dos obligatorias y una opcional. Pero lo más interesante del parámetro mm es la capacidad de, para una consulta con más de n palabras, poder fijarle un porcentaje de palabras obligatorias (u opcionales). Por ejemplo 2<75% se lee: "Se recupera el documento si al haber más de de dos palabras en la consulta, coinciden el 75%, en caso de que haya una o dos son ambas las que tienen que coincidir". Además de lo anterior, podemos hacer combinaciones del estilo 2<75% 5<50% (si hay más de dos palabras tienen que coincidir el 75%, pero si hay más de 5 alcanza con solo el 50% para recuperar el documento).
- qf o *Query Fields*, es el parámetro donde se ingresan los campos sobre los cuales se realiza la consulta junto con los pesos de cada campo.

- pf o *Phrase Fields*, este parámetro se utiliza para potenciar la relevancia de los documentos que contengan frases exactas o que se aproximan. Por medio de este parámetro le decimos cuáles son los campos en los que queremos potenciar dichas frases o aproximaciones.
- qs o *Query Phrase Slop*, este parámetro nos permite configurar la proximidad entre las palabras de una consulta explícita por una frase, es decir cuántas palabras pueden haber entre las que componen la frase.
- ps o Phrase Slop, este parámetro nos permite configurar la proximidad por defecto entre las palabras de una frase (las configuradas en el parámetro pf), para ,de esta forma, potenciar su relevancia.
- Luego existen más parámetros como ps2, ps3, pf2 y pf3. Estos cumplen la misma función que los campos pf y ps, solo que ahora se consideran bigramas y trigramas para 2 y 3 respectivamente.

5.5 AGRUPAMIENTO DE RESULTADOS

Para este proyecto se tomó la URL como identificador del artículo de prensa; pero puede pasar que dos páginas iguales tengan distinta URL (ya que hay muchas páginas web que están mal hechas). Por lo tanto surge la necesidad de evitar que aparezcan noticias y opiniones duplicadas cuando el usuario realice una búsqueda, para esto se utilizó el parámetro *group* que provee Solr.

La agrupación por resultados, es la capacidad de agrupar los resultados de la búsqueda por el valor de un campo u otro posible criterio a definir.

El colapso de campos por otro lado nos permite remover de los resultados de la búsqueda documentos que contengan el mismo valor en un campo que otro documento.

Se agrupan todas las opiniones en que el texto del mensaje sea exactamente el mismo antes de aplicar el stemmer. Para esto se configura el parámetro de Solar *group.field* y se establece el valor correspondiente en dicho parámetro. Lo que se utiliza en este caso es el colapso de campos, pues para los resultados se muestra únicamente una opinión para cada mensaje.

5.6 FILTRADO DE RESULTADOS

Otra funcionalidad importante para nuestro proyecto es el filtrado de resultados según fechas, medios de prensa, fuente de la opinión y cantidad de resultados (si bien esta última utiliza otras funcionalidades para implementarse).

Para filtrar resultados se utiliza el parámetro fq (*filter query*), que nos permite utilizarlo varias veces, de esta forma para cada filtro que queremos aplicar asignamos un nuevo parámetro fq con el campo y el valor a filtrar.

Para el parámetro fq se pueden utilizar rangos al igual que para todos los valores numéricos, esto resulta útil a la hora de filtrar por fechas pues podemos establecer un rango de fechas de la siguiente forma: "fecha:[fecha1 TO fecha2]".

Por otro lado para filtrar por cantidad de resultados se utiliza un parámetro especial llamado rows, que nos permite limitar directamente el número de resultados.

5.7 BÚSQUEDAS SOBRE SOLR

Para buscar sobre Solr se utiliza la interfaz que se brinda por medio del protocolo HTTP con el método GET. Para esto es necesario construir una URL con los parámetros recibidos de la página web.

La URL mencionada anteriormente contienen la dirección del servidor de Solr, a qué colección se va acceder e indica que se va a utilizar el método select, es decir, se va a realizar una consulta.

Con esto se realiza una consulta sobre la base de texto. La consulta tiene ciertas características que se definen mediante los siguientes parámetros: Asunto, fuente, rango de fechas, medio de prensa, cantidad de resultados, agrupamiento de resultados y EDismax.

El asunto es el que determina qué opinión se va a recuperar, por lo tanto el mismo se utiliza como principal motivo de la consulta. El mismo se utiliza como consulta en los distintos campos del documento, con diversos valores de relevancia que se definen mediante EDismax.

La fuente se decidió utilizar como filtro en la búsqueda sobre el campo "*Fuente"*, dado que se requiere filtrar por quién emite la opinión. En caso de que el usuario no ingrese una fuente, el filtro se deja vacío, por lo que recupera todos los documentos que se

Rango de fechas, este parámetro es opcional e introducido por el usuario. El mismo es también un filtro, dado que lo que se requiere es que no se recuperen documentos en que el campo fecha no esté dentro de este rango de fechas.

El medio de prensa en el que queremos buscar, este parámetro también opcional e introducido por el usuario. El mismo sirve para filtrar documentos que no pertenezcan a un medio de prensa en particular, por lo tanto el mismo se utiliza como filtro sobre el campo de Solr "*medio de prensa*".

Cantidad de resultados que queremos desplegar, este valor puede ser introducido por el usuario, sirve para limitar la cantidad de resultados que queremos que solr despliegue. Este parámetro utiliza el parámetro rows de Solr, el mismo está especialmente diseñado para limitar la cantidad de resultados. Además de esto, en caso de que el usuario no introduzca un valor, se da uno por defecto calculado en base a la cantidad de resultados recuperados con la siguiente expresión: $\log_{1.5}(\text{cantidadDeResultadosRecuperados} + 1)$. Esto permite evitar mostrar resultados que no tienen mucha relevancia, pues siempre se despliegan los resultados más relevantes primero.

Agrupamiento de resultados, este parámetro es utilizado internamente por el sistema, es decir es invisible al usuario, y sirve, como se explicó anteriormente, para solucionar el problema que surge de tener noticias repetidas en el índice.

Herramienta EDismax, con esto se le comunica a Solr como se quiere utilizar esta herramienta, configurándose relevancia, cantidad de palabras que deben aparecer y el efecto que tienen las frases exactas. Analizando los datos se tomó como criterio que si la consulta tiene entre tres y cinco palabras, el 75% de las mismas deben aparecer en el documento para que éste sea recuperado, mientras que si tiene más de cinco palabras, alcanza con que el 50% de las mismas aparezcan en el documento para que el mismo sea recuperado.

Todos los campos se decidieron relevantes, pues puede pasar que una opinión haga referencia a un asunto que no se menciona explícitamente en la misma, pero que se obtiene a partir del

contexto. Sin embargo se le da seis veces más relevancia a lo que aparece en el campo opinión, pues es más probable que la misma sea algo concreto sobre el tema.

Por otro lado también es importante configurar la regla de proximidad para una frase exacta, este valor se setea en uno, es decir que se toman como frases exactas aproximaciones que tengan hasta una palabra en medio. En este caso, las frases exactas no afectan cuando un documento debe ser recuperado, sin embargo, aumentan de manera notoria la relevancia del mismo.

En resumen la consulta queda definida mediante una consulta del campo *asunto* filtrando con el resto de parámetros de búsqueda, y configurada mediante los parámetros especiales de EDismax que permiten descartar opiniones y darles distinta relevancia según la cantidad y distribución de las palabras de la consulta que aparecen en un documento.

5.8 PERSONAS QUE TAMBIÉN OPINARON SOBRE UN TEMA

5.8.1 Introducción

Con el objetivo de enriquecer la experiencia del usuario, se decidió crear una herramienta que, dada una búsqueda de opiniones emitidas por una persona acerca de un tema determinado, devuelve una lista de personas que también opinaron acerca de ese tema.

Para lograr esto se utilizó una de las tantas características que provee Solr que es la búsqueda facetada (37). El facetado de una búsqueda se trata de ofrecer más información acerca de la búsqueda realizada utilizando algunas campos específicos dentro de nuestro schema. Esto se utiliza mucho dentro del e-commerce donde páginas como Amazon, luego de realizada una búsqueda nos permite filtrarla según distintos criterios. Por ejemplo, si buscamos “camiseta” aparecen distintas opciones para filtrar resultados como: “colores: rojo, verde, azul”, “modelo: sin mangas, con mangas”, etc.

En nuestro caso, haremos un facetado del campo “fuente” de nuestro schema para que dada una búsqueda sobre un tema, nos diga cuáles fueron las fuentes que más opinaron sobre ese tema. Para esto, cada vez que un usuario realice una consulta en el sistema, se realizarán dos búsquedas, la primera para encontrar opiniones como se describió en la parte anterior, y la segunda utilizando facetado para encontrar otras fuentes que opinaron sobre el tema.

La consulta que se realiza sobre Solr es prácticamente igual a la de la parte anterior pero sin incluir el parámetro de filtrado “fq” e incluyendo los parámetros de facetado “facet” y “facet.field”. El parámetro “facet” se setea en true, indicando que deseamos realizar facetado.El parámetro “facet.field” se setea en “fuente_facetado” que es un campo especial de nuestro schema sobre el cual haremos el facetado.

5.8.2 Configuración de Solr

Para poder realizar el facetado sobre las fuentes, se creó un campo adicional en el schema de Solr llamado fuente_facetado donde se guarda la fuente de la opinión repetida, pero con algunos cambios. Estos cambios surgen a partir del siguiente procesamiento que se realiza sobre el texto:

1. Se tokeniza el texto.

2. Se pasa el texto a minúsculas.
3. Sólo se deja en el campo las palabras que estén en el archivo apellidos.txt. En este archivo se encuentra una lista de apellidos comunes en Uruguay. Por lo tanto se descartan los nombres de pila y cualquier otra palabra que no sea un apellido.

Un extracto del XML de configuración se puede ver en el anexo de Detalles de Implementación, junto a una descripción del mismo.

5.8.3 Archivo de apellidos

Todo el procesamiento hecho sobre la fuente de las opiniones es con el objetivo de dejar únicamente el texto que corresponde a un apellido. Esto es ya que si no se filtraba el resto de las palabras, pueden aparecer como fuentes sugeridas dos veces la misma persona, esto es por ejemplo “Mujica” y, “José Mujica” o “El presidente Mujica”. Para evitar esto, en el campo “fuente_facetado” se deja únicamente (para este ejemplo) la palabra “Mujica”.

Con el objetivo de tener una lista de apellidos por la cual pudiésemos filtrar, buscamos en internet y encontramos que wikipedia mantiene una lista de nombres de políticos uruguayos (38) y también una lista de nombres de futbolistas uruguayos (39). A partir de estas páginas web, hicimos un procesamiento “a mano” para eliminar todo el html y los nombres de pila para dejar únicamente los apellidos que allí aparecen. De esta forma pudimos generar un archivo que contiene alrededor de 1500 apellidos que sabemos se utilizan en Uruguay.

5.9 HIGHLIGHTING

Highlighting se le llama al resaltado en negrita que hacen los buscadores sobre los resultados de las búsquedas. Por ejemplo si en un buscador cualquiera (como por ejemplo Google) se realiza la búsqueda “regasificadora de ancap”, en los resultados de la misma, aparecen porciones de texto donde aparecen los términos de búsqueda como “regasificadora”, “ancap” y también “gas”.

Esta funcionalidad en Solr se implementa mediante la característica de Highlighting que ya viene implementada. Para utilizarla se debe setear en la consulta a Solr el parámetro “hl” en “true” y en el parámetro “hl.fl” el nombre del campo de donde se extraerán las palabras resaltadas. En nuestro caso elegimos para este último parámetro el valor “articulo” ya que es el campo donde se guarda todo el artículo periodístico.

Aquí se pueden ver más detalles acerca de esta característica (40).

5.10 SPELLCHECK

5.10.1 Introducción

A la hora de realizar una búsqueda uno se puede equivocar al escribir las palabras ya sea por un falta de ortografía o un error de tipeo. Para esto, los motores de búsqueda en general incluyen una herramienta “Quizás quiso decir...” que sugiere, al detectar un error, una búsqueda con las palabras corregidas. Solr realiza la implementación de esta herramienta mediante el Spellcheck.

5.10.2 Configuración en Solr

Para poder utilizar esta herramienta se tuvo que agregar algunas cosas al schema.xml. Se tuvo que crear un nuevo campo donde se mantiene un diccionario con todas las palabras de cada

artículo de prensa. Es decir, se mantiene en un campo, todas las palabras que han sido indexadas hasta el momento. Esto es para luego poder realizar una consulta sobre Solr y que usando la distancia de Levenshtein (41) pueda corregir si alguna palabra está mal escrita.

A su vez, también se tuvo que modificar el archivo de configuración solrconfig.xml para que el query handler "/spell" que viene configurado por defecto, utilizara el campo spellcheck_es recién definido.

En el anexo de Detalles de Implementación, capítulo 4, se puede ver una explicación más detallada acerca de esta herramienta.

6 TEMAS DE LA SEMANA

6.1 INTRODUCCIÓN

Como se comentó anteriormente, uno de los objetivos del proyecto es permitir ver cuáles fueron los temas de los que más habló la prensa dentro de una semana determinada. Para esto, el procedimiento elegido fue extraer para cada noticia de la semana, sus términos clave, para luego agrupar las noticias utilizando un algoritmo de clustering; donde cada cluster representará un "tema" de esa semana.

6.2 DISEÑO DEL MÓDULO

Este módulo en realidad funciona dentro del webserver de la interfaz de usuario que se explicará en el próximo capítulo.

Cada una determinada cantidad de tiempo, se corre un proceso ejecuta un algoritmo que identifica para cada semana, cuáles fueron los temas más importantes que aparecieron en la prensa escrita. Estos temas se guardan en un archivo que luego serán consultados por la interfaz de usuario para desplegarlos en pantalla.

6.3 EXTRACCIÓN DE LOS TÉRMINOS CLAVE

Cada artículo de prensa puede verse como el conjunto de las palabras que lo componen. Sin embargo, existe un subconjunto de dichas palabras que son las que mejor describen sobre qué es lo que se está hablando en dicho artículo. Esto puede verse fácilmente si se considera el título de una noticia, que mediante unas pocas palabras describe sobre qué trata la misma. Por ejemplo, en un artículo donde se habla acerca de un préstamo del FMI, las siguientes palabras podrían estar incluidas dentro del conjunto de palabras clave: "fmi", "préstamo", "economía", "astori" (antiguo ministro de economía), etc.

Para realizar la extracción de las palabras clave, utilizamos el módulo *MoreLikeThis* (42) provisto por Solr, que utiliza la librería de Lucene con el mismo nombre. A continuación comentamos cómo es que funciona este algoritmo.

Primero se toman todas las palabras de un artículo y se ponen en un vector junto con su frecuencia de aparición en el documento, quitando las stopwords y realizando stemming sobre las palabras restantes. A modo de ejemplo, utilizaremos el siguiente vector: mujic[4], pluna[5], lorenzo[2], mes[1], negoci[3], extrapol[2]. De aquí en más, nos referiremos a estas palabras procesadas con stemming como "términos".

Luego, se descartan los términos que no aparecen más de una determinada cantidad de veces dentro del documento. En nuestro caso, configuramos para que este número sea 1, por lo tanto, se descarta el término "mes" de nuestro vector de ejemplo. A su vez, también se descartan los términos que no aparecen en al menos 3 documentos dentro del corpus, buscando con esto eliminar términos que sean demasiado raros. En nuestro ejemplo, tenemos que quitar el término "extrapol" (por extrapolación).

Continuando con el algoritmo, se realiza el cálculo para cada una de los términos, acerca de la frecuencia inversa de aparición en el corpus (Inverse Document Frequency o IDF) con el objetivo de determinar cuáles términos son los que menos aparecen en el resto de los documento y por lo tanto, que mejor describen al mismo. El IDF se calcula de la siguiente forma:

$$\operatorname{idf}(t, D) = \log \frac{|D|}{|\{d \in D : t \in d\}|}$$

Donde D es el conjunto de todos los documentos y t es un término dentro de un documento.

Multiplicando el IDF de un término por la cantidad de veces que aparece en el documento, se puede obtener un puntaje acerca de qué tan "importante" es el término para ese documento. En nuestro caso, para obtener el subconjunto de los términos que describen a un artículo de prensa, nos quedamos con los 15 términos que mejor quedan en el ranking según este cálculo.

Una descripción más detallada del algoritmo aplicado por Solr puede encontrarse en (43).

6.4 CLUSTERING

El agrupamiento o clustering es la tarea de agrupar un conjunto de objetos de tal manera que los objetos en el mismo grupo (llamado cluster) son más similares (en uno u otro sentido) entre sí que a las de otros grupos (clusters) .

Para aplicar este concepto, primero se armó un conjunto con todas los términos clave extraídas en la parte anterior. Llamaremos a este conjunto "diccionario".

Representaremos a cada artículo como un vector de largo igual al tamaño de nuestro diccionario, donde cada posición en el vector representa un término y tendrá un cero si el artículo no tenía ese término como término clave y el puntaje obtenido en la parte anterior en caso contrario. Un posible vector, por ejemplo, sería: <0,0,0.8,0,1,0>.

Con el objetivo de obtener varios clusters, agrupando estos vectores en conjuntos, utilizaremos el algoritmo de Expectation-Maximization. Este es un método iterativo de aprendizaje no supervisado para encontrar una estimación de la máxima verosimilitud de los

parámetros en un modelo estadístico que depende de variables latentes no observables. Una descripción completa de este algoritmo puede encontrarse en (44).

Utilizamos nuevamente la librería Weka que ya trae implementado el algoritmo de EM, tratando de obtener 15 clusters distintos haciendo 100 iteraciones del algoritmo. De esta forma, obtenemos los vectores (que cada uno representa un artículo periodístico) agrupados en 15 conjuntos distintos.

Se revisa cada clúster, dentro del grupo de 15, y si tiene menos de 5 noticias se descarta. Esto es ya que consideramos que un tema representado por un clúster con menos de 5 noticias no es lo suficientemente relevante para estar entre los temas de la semana.

6.5 OBTENCIÓN DEL TEMA DE LA SEMANA

Como paso final para determinar cuál es el tema principal de cada clúster, vamos a elegir una noticia que sea la más representativa de cada conjunto. Para esto, tomamos nuevamente los términos clave de cada noticia, pero esta vez considerándolos dentro de cada clúster. Armamos un nuevo diccionario de términos y sumamos los puntajes obtenidos en cada noticia por cada término para obtener un único puntaje por clúster. Por ejemplo, si dentro de un clúster un término apareció en tres artículos distintos primero con un puntaje de 0.8, luego con 0.3 y finalmente con 0.9, el puntaje final sería 2.

Teniendo para cada clúster una lista de términos ordenados por su puntaje total, podemos elegir una noticia representativa del mismo realizando una búsqueda en Solr con los 7 términos con el puntaje más alto.

6.6 CONFIGURACIÓN DE SOLR

Para realizar la parte de la obtención de los temas de la semana, se creó una instancia separada de Solr donde sólo se indexan los datos de una noticia y no de las opiniones. Es decir, se mantienen los campos: url, artículo, title, metatitle, h1, categorías, descripción, autor y fecha.

Por otro lado, también hubo que modificar la configuración de esta nueva instancia para que permita la utilización de la herramienta *MoreLikeThis* the Solr. Esto se detalla un poco más en el anexo de Detalles de Implementación, en el capítulo 5.

7 INTERFAZ DE USUARIO

7.1 INTRODUCCIÓN

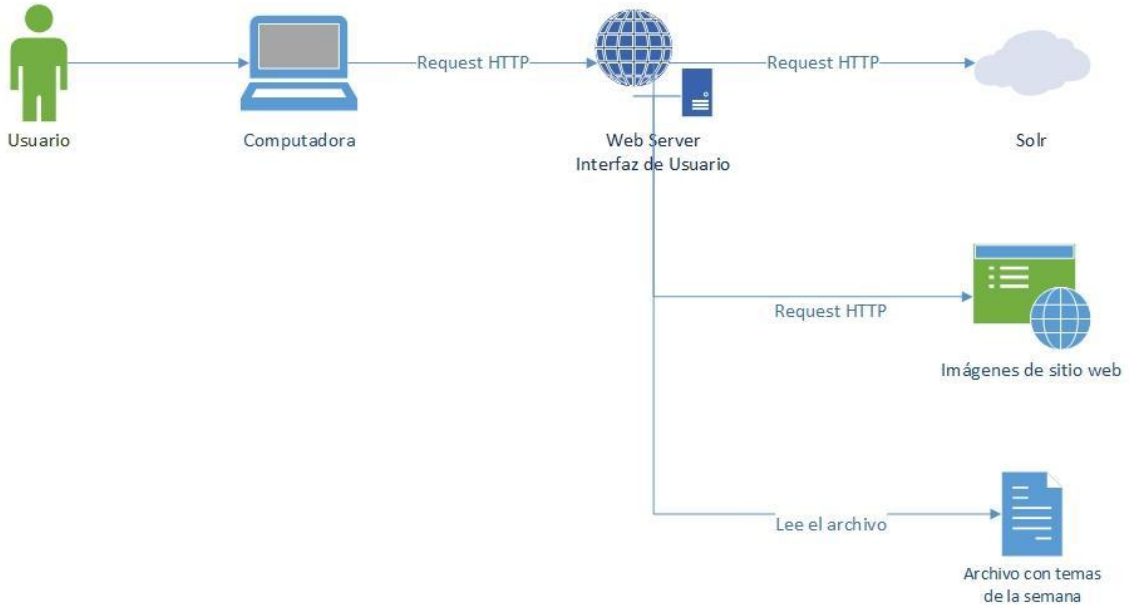
La interfaz de usuario es el medio mediante el cual los usuarios interactúan con nuestro sistema. La implementación de la misma se hará de forma web utilizando Java EE. Si bien no utilizamos ningún framework para el desarrollo de la interfaz, seguimos el patrón de diseño MVC (Model View Controller) usando los servlets de Java y JSP (Java Server Pages).

Uno de los principales objetivos del diseño de la interfaz es que sea atractiva para el usuario y que la información se presente de una forma clara y concisa al realizar las búsquedas.

7.2 DISEÑO DEL MÓDULO

La interfaz no es solo una forma de presentar la información de forma visualmente atractiva al usuario, sino que se encarga de ejecutar los algoritmos de búsqueda que permiten traer correctamente la información de las opiniones y de los temas de la semana.

El web server además de comunicarse con Solr para realizar las búsquedas, se conecta con las páginas web de los medios para extraer las imágenes de las noticias ya que las mismas no se encuentran almacenadas localmente.

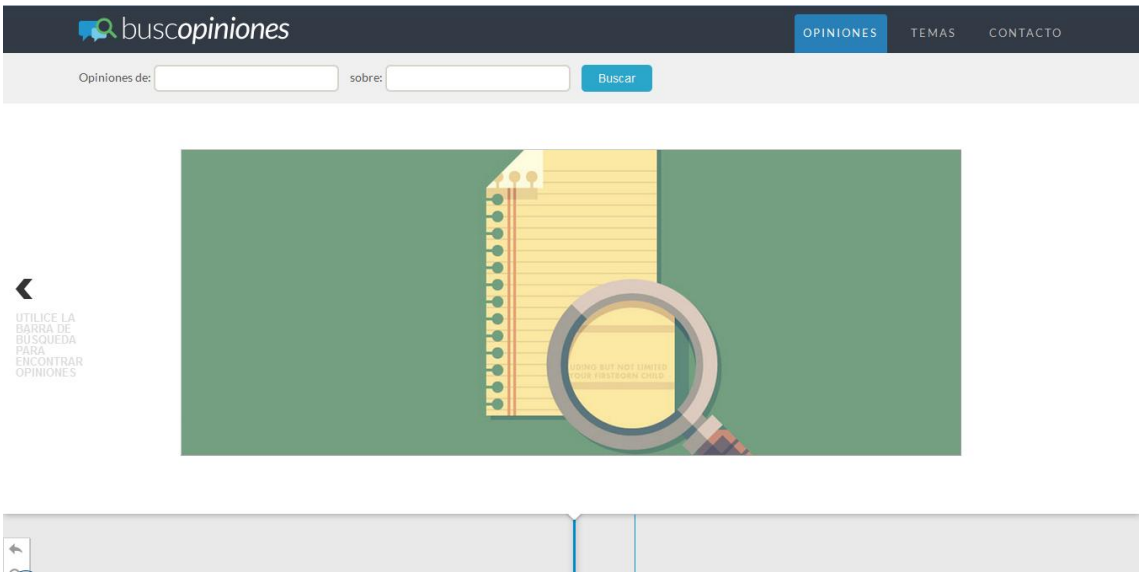


7.3 INTERFAZ HTML

Para el desarrollo de la interfaz se utilizó un template llamado Flati (45) que está hecho con Bootstrap (46) el cual es una serie de librerías CSS y HTML para desarrollar interfaces minimalistas. Decidimos realizarlo de esta forma para que el usuario pueda entender

fácilmente la información que se le presenta en la pantalla, pudiendo interactuar con el sistema cómodamente como lo hace en su día a día en internet.

En la primera pantalla que se le presenta puede realizar una búsqueda acerca de opiniones que tuvo una persona sobre un determinado tema como se muestra en la siguiente captura:

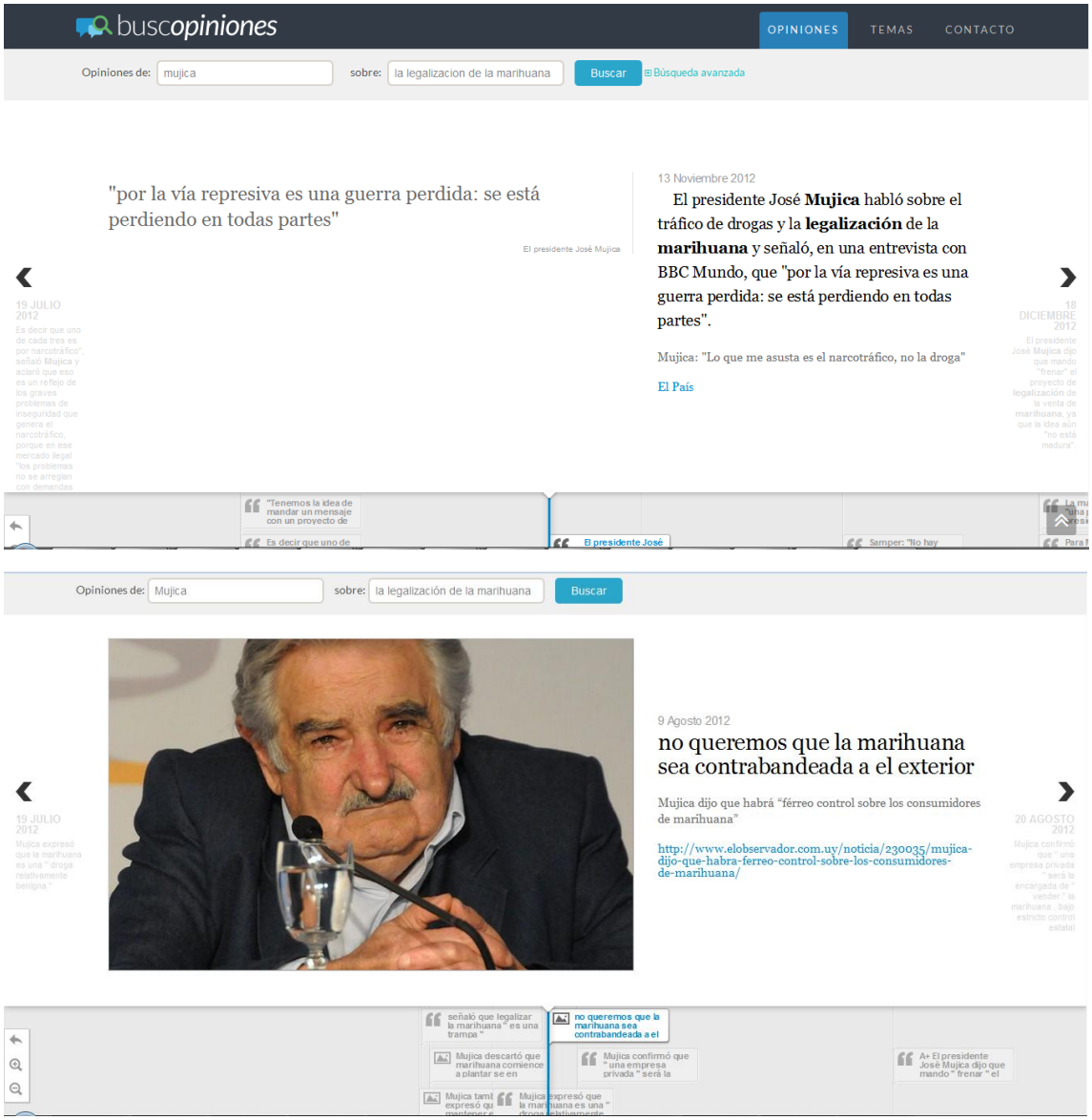


7.4 LÍNEA DE TIEMPO

Con el objetivo de presentar los datos que retorna la búsqueda de una forma amigable y entendible para el usuario, se utilizó un GUI Widget llamado TimelineJS (47), open-source, que está desarrollado en javascript, html y css que permite la creación de líneas de tiempo para presentar información acerca de hechos que sucedieron con un cierto orden cronológico. Este widget permite la incorporación de imágenes y citas de una forma visualmente atractiva, además de permitir una fácil navegación entre una serie de eventos.

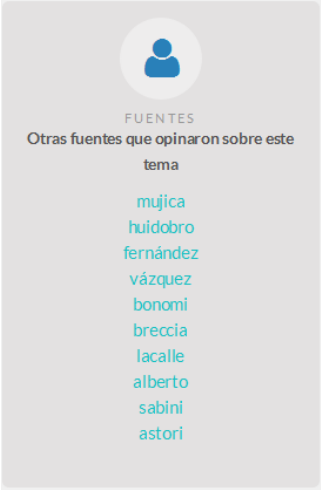
La idea es a partir de una búsqueda, por ejemplo, opiniones de “Mujica” sobre “la legalización de la marihuana”, generar una línea de tiempo que permite recorrer de forma cronológica, empezando por la opinión emitida más recientemente, y retrocediendo hacia el pasado.

En el caso de que la opinión cuente con una cita directa, esto es una frase entre comillas, esta se muestra por separado y más grande para mejorar la lectura. Si esto no sucede, se trata de traer una imagen de la noticia desde la web desde donde esta haya sido extraída. En las siguientes capturas se muestra un ejemplo de esto:

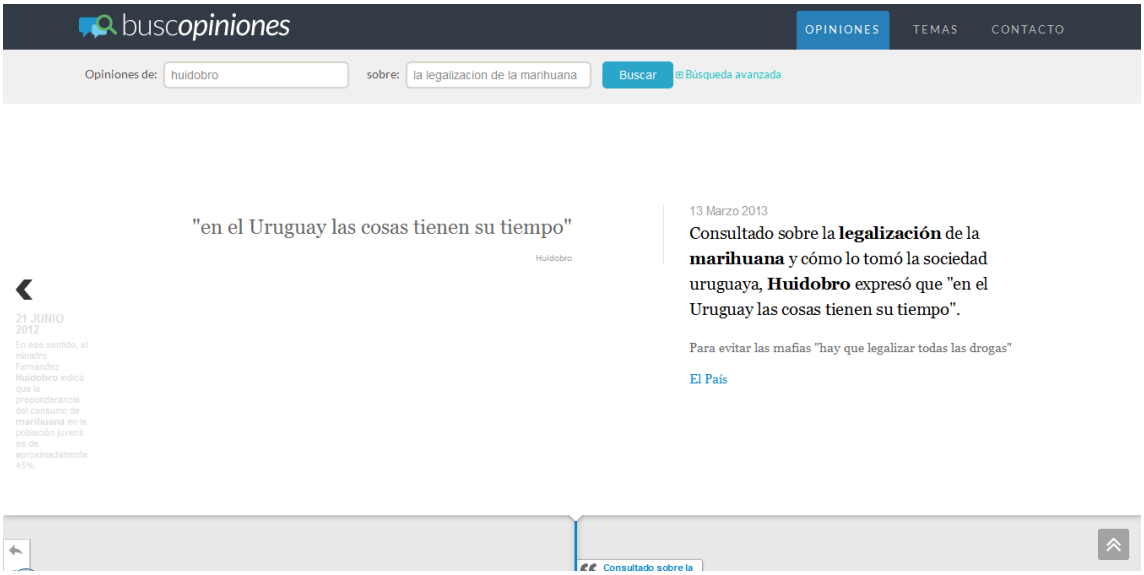


7.5 FUENTES SUGERIDAS

Con el objetivo de mejorar la experiencia del usuario al realizar la búsqueda de la información, se le proporciona una herramienta donde se le sugiere otras fuentes que opinaron sobre el tema buscado. En el caso del ejemplo anterior, la siguiente es una lista de fuentes sugeridas:



Haciendo clic en alguna de las fuentes, automáticamente se realiza una nueva búsqueda sobre el mismo tema pero con la fuente seleccionada. Por ejemplo, seleccionando “Huidobro”:



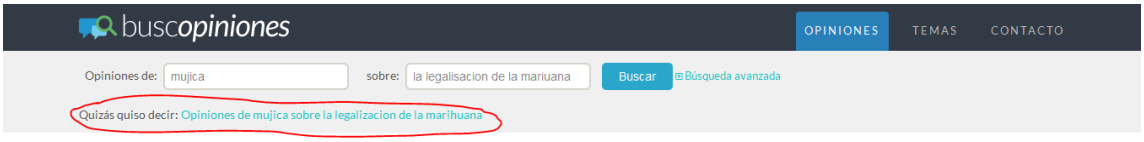
7.6 HIGHLIGHTING

Como se puede ver en las capturas anteriores, se utilizó la herramienta de highlighting de Solr para obtener qué palabras deben resaltarse (poner en negrita) en el texto que coincidan con términos que se utilizaron en la búsqueda.

El resaltado de estas palabras es importante a la hora de explicar al usuario por qué el resultado de una búsqueda está relacionada a los términos ingresados.

7.7 SPELLCHECK

Con el objetivo de mejorar la experiencia del usuario en su interacción con el sistema, utilizamos la herramienta de SpellCheck que proporciona Solr como se muestra a continuación:



Esto nos permite corregir la búsqueda del usuario si el mismo se equivoca al realizarla cometiendo un error de tipeo o una falta de ortografía.

7.8 EXTRACCIÓN DE IMÁGENES

Siguiendo con la línea de tratar de hacer la interfaz de usuario lo más amigable posible, decidimos extraer las imágenes de las páginas de los medios de prensa. En principio habíamos descartado extraer las imágenes de los artículos de prensa en la etapa de scraping puesto que las imágenes pueden ocupar muchísimo espacio en disco y nos podía traer el problema de quedarnos sin espacio para seguir guardando artículos.

Por lo tanto, en esta etapa lo que vamos a hacer es descargar las imágenes "on demand", esto es, cuando se realiza una búsqueda obtenemos un conjunto de opiniones y de cada opinión sabemos la url de la página de donde la misma se extrajo. De esta forma, realizamos un GET a la página y analizamos dentro del html de la misma, todas las urls de imágenes que están embebidas. Consideramos como imágenes aquellas urls que terminan en .jpg, .gif o .png.

Teniendo para cada una de las opiniones un conjunto de imágenes posibles para mostrar, dentro de las cuales se encuentran por ejemplo, el logo del medio de prensa y otras imágenes propias del diseño de la página web en sí, es necesario tener un criterio para elegir cuál es la imagen correspondiente al artículo de prensa (si es que esta existe). El siguiente es un pseudocódigo del algoritmo que elige la imagen:

```
maxImagen = 0
maxHeight = 200;
maxWidth = 200;
maxProporcion = 2.0;
minProporcion = 0.3;
mejorImagen = null
Para cada imagen en la página:
   height = alto de la imagen
   width = ancho de la imagen
   proporcion = height / width
   si (maxImagen <= (height * width) and (height * width) >= (maxHeight * maxWidth) and
proporcion < maxProporcion and proporcion > minProporcion) entonces
      mejorImagen = imagen;
      maxImagen = height * width;
   finsi
finpara
```

Analizando el algoritmo, nos quedamos con la imagen más grande que respete las proporciones deseadas. Si la imagen es más chica que un tamaño determinado, entonces no la tomamos. Los números que se ven el pseudocódigo surgieron de analizar las imágenes que aparecen en los distintos medios de prensa y de concluir que en general la imagen más grande de proporciones más o menos "cuadradas" es la imagen que corresponde al artículo.

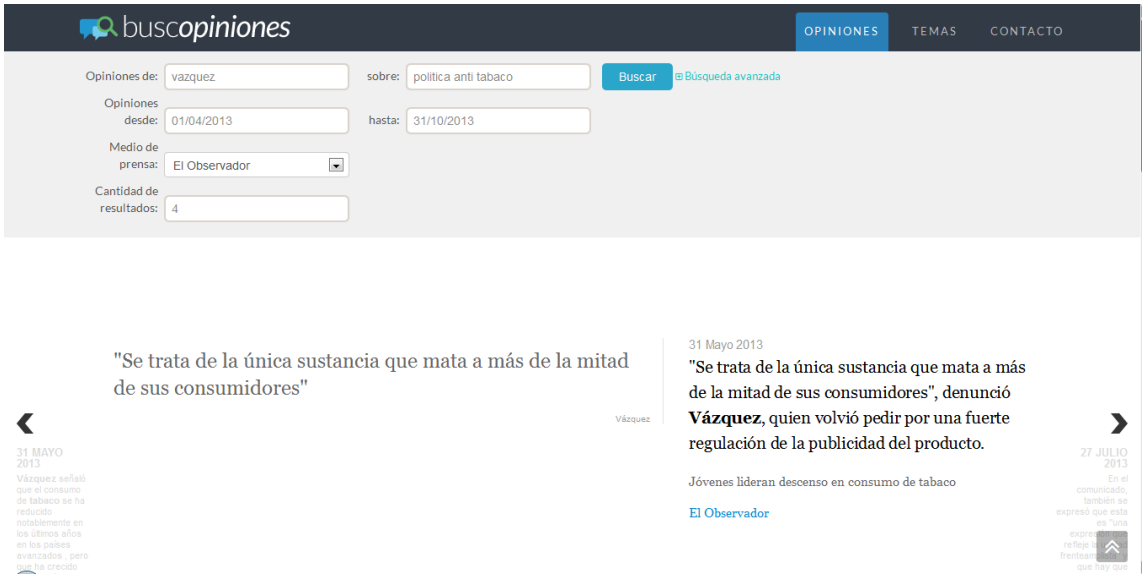
Por otro lado, también surgió el problema de los tiempos de carga, ya que para cada una de las opiniones que aparecen en el resultado, se descargan y analizan cada una de las imágenes del artículo de donde fueron extraídas. Por lo tanto, se decidió utilizar la clase *ExecutorService* de Java que nos permite crear varios threads de ejecución para descargar y analizar en paralelo las imágenes de los artículos para de esta forma reducir el tiempo de carga.

7.9 BÚSQUEDA AVANZADA

Con el objetivo de darle más control al usuario sobre los resultados de búsqueda que se muestran, decidimos incluir algunos opciones de búsqueda "avanzadas" para que el usuario pueda filtrar los resultados. Estas opciones son básicamente 3: filtrar por el rango de fecha en el cual se emitieron las opiniones, filtrar por medio de prensa en el cual aparecieron las opiniones y elegir la cantidad de resultados a mostrar en la línea de tiempo.

La implementación de estos filtros está descrito en la sección sobre el Motor de Búsqueda.

Las opciones de búsqueda avanzada se encuentran disponibles a través de un menú desplegable al cual se puede acceder haciendo click en donde dice "Búsqueda avanzada":



7.10 CONEXIÓN DE LA INTERFAZ CON LA BÚSQUEDA EN SOLR

La interfaz de usuario está estrechamente relacionada con el motor de búsquedas ya que es la que se encarga de tomar el xml devuelto por Solr y transformarlo en algo más comprensible para el usuario final.

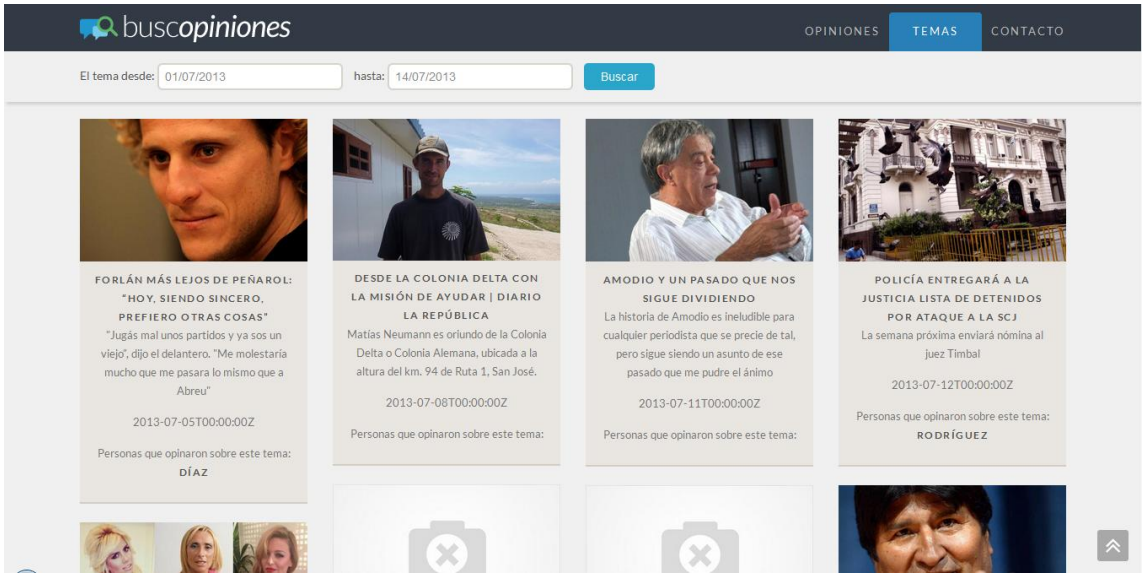
Una vez realizada una búsqueda a través de la interfaz, se pasa a través de un GET HTTP los parámetros de la misma: fuente de la opinión, tema (o asunto) de la opinión, rango de fechas en el cual se está buscando las opiniones, medio de prensa en el cual se emitió la opinión, cantidad de resultados a mostrar (en caso de que este número supere a la cantidad de resultados devueltos por Solr, se muestran la totalidad de los resultados).

El rango de fechas, el medio de prensa y la cantidad de resultados son parámetros opcionales que están disponibles a través de la búsqueda avanzada. En el caso del rango de fechas y el medio de prensa, al quedar vacíos se interpreta que deben retornarse las opiniones en cualquier rango de fecha y/o cualquier medio de prensa. En el caso de la cantidad de

resultados, si esta se deja vacía, se retorna la cantidad calculada por el algoritmo descrito en el capítulo 5.7.

7.11 TEMAS DE LA SEMANA

Para acceder a la visualización de la parte de nuestro proyecto del tema de la semana, hay que hacer click en la pestaña “Temas”. En esta pantalla se puede seleccionar un período de fechas del cual se desea obtener cuáles fueron los principales temas de los cuales hablaron los medios. Para ello se selecciona una fecha de inicio y una de fin del período donde haciendo click en buscar se despliega la información requerida:



El algoritmo utilizado para encontrar las noticias representantes de cada tema de la semana es el que fue descrito en la sección del Tema de la Semana.

Al igual que en la búsqueda de opiniones se decidió utilizar una interfaz amigable con el usuario, que contenga tanto el título como la descripción de la noticia así como una imagen de la misma extraída con el algoritmo presentado en la sección previa.

8 EVALUACIÓN DE LOS RESULTADOS

8.1 INTRODUCCIÓN

La evaluación de los resultados del proyecto se va a dividir en cinco partes:

- Evaluación de los resultados obtenidos en el crawling y scraping de los medios de prensa.
- Evaluación de los resultados obtenidos en la búsqueda de opiniones.
- Evaluación del sistema de fuentes que también opinaron sobre un tema.
- Evaluación de la correctitud de los temas de la semana.
- Realización de un test de usuario para la evaluación del sistema desde un punto de vista más general.

8.2 CRAWLING Y SCRAPING DE LOS MEDIOS DE PRENSA

8.2.1 Introducción

En esta parte se realizará una evaluación de los resultados que obtuvimos comentando algunos de los problemas que tiene esta implementación y posibles trabajos a futuro para mejorarlo.

8.2.2 Crawling de la prensa uruguaya

El primer objetivo de nuestro proyecto era lograr crear un corpus de prensa uruguaya. Para esto, como se comentó en la parte de implementación, se utilizó nutch para intentar descargar todas las páginas de tres medios de prensa uruguayos: El País, El Observador y La República.

Es difícil saber si logramos descargar todas las páginas de un sitio, ya que estos no publican en ningun lado una lista de todas sus páginas existentes. Lo que utilizaremos para realizar una aproximación es fijarnos en Google la cantidad de páginas con las que cuenta un medio y medir la cantidad que nosotros tenemos descargadas, verificando que más o menos son comparables. Tomamos como un supuesto para la evaluación de estos resultados que Google tiene absolutamente todas las páginas crawleadas para los tres medios.

En total, al momento de escribir este informe, se tienen descargadas 422.039 páginas para El País, 86.156 páginas para El Observador y 64.351 páginas para La República. Comparando, Google tiene 2.840.000, 284.000 y 66.400 páginas para El País, El Observador y La República respectivamente.

Analizando ahora estos resultados, podemos ver que la cantidad de páginas crawleadas por nosotros para La República es prácticamente la misma que las que tiene Google, pero para El País y El Observador es significativamente inferior.

En principio, nuestro objetivo no era descargar la totalidad de las páginas de los medios sino generar un corpus lo suficientemente grande como para que se pudiera realizar búsquedas sobre opiniones. Sin embargo, podríamos haber logrado crawlear la totalidad de los medios si hubiésemos contado con un servidor dedicado para correr nutch en vez de utilizar una máquina de bajos recursos que sólo corría por las noches.

8.2.3 Scraping de las páginas HTML

Otro de los objetivos de este trabajo es extraer metadatos de los artículos de prensa. Como se comentó anteriormente, los datos a extraer son los siguientes: título, subtítulo, copete, fecha de publicación, categoría, etiquetas, artículo, noticias relacionadas y autor.

En retrospectiva y teniendo ahora todo el sistema ya implementado, podemos ver que los atributos más importantes, puesto que son los que más se utilizaron en el buscador de opiniones, fueron el título, el copete, la fecha de publicación y el artículo de prensa en sí mismo. Por lo tanto será sobre estas características que evaluaremos los resultados.

Para realizar la evaluación tomaremos 25 noticias al azar de cada uno de los medios de nuestro corpus y verificaremos a mano si estos atributos fueron correctamente extraídos. Las URLs de estas 75 noticias se pueden encontrar en el anexo de evaluación. A continuación se muestran los resultados medidos como cantidad de éxitos sobre casos totales.

	Título	Copete	Fecha	Artículo
El País	22/25	18/25	21/25	21/25
El Observador	25/25	25/25	25/25	25/25
La República	25/25	23/25	25/25	25/25

En el caso de El País, cabe destacar que en 3 de los 4 casos donde no se pudo extraer correctamente las fechas fue porque en la página la misma no estaba presente. En los casos donde no se pudo realizar correctamente la extracción del artículo fue porque se agregó información perteneciente al boilerplate de la página y no porque se haya descartado al artículo en sí mismo, por lo tanto se puede considerar como un error menor.

A raíz de estos números podemos concluir que el scraper tuvo un buen rendimiento, sobre todo para El Observador y La República donde se obtuvo prácticamente un 100% de efectividad. En el caso de el país, los errores se pueden atribuir al hecho de que tiene muchas más páginas que los otros dos medios y además utiliza muchos formatos distintos para presentarlas, haciendo mucho más difícil la extracción de los atributos.

8.3 BUSCADOR DE OPINIONES

8.3.1 Introducción

En esta parte primero definiremos algunas métricas para la evaluación del sistema para luego realizar algunas búsquedas sobre temas que creamos que han sido relevantes en el último año, consultando opiniones sobre diferentes actores políticos.

8.3.2 Medidas de rendimiento y correctitud

8.3.2.1 Introducción

A la hora de evaluar el correcto funcionamiento de un motor de búsqueda, se utilizan diferentes medidas. Estas necesitan de una colección de documentos y una consulta. A continuación serán descritas algunas medidas comunes, las cuales asumen que: de cada documento se sabe si este es relevante o no para una consulta en particular.

8.3.2.2 Precisión

La precisión es la fracción de documentos recuperados que son relevantes para las necesidades de información del usuario y se calcula como:

$$\text{Precisión} = \frac{|\{\text{documentos relevantes}\} \cap \{\text{documentos recuperados}\}|}{|\{\text{documentos recuperados}\}|}$$

8.3.2.3 Recobrado (Recall)

El recall es la fracción de documentos relevantes para una consulta que fueron recuperados.

$$\text{Recobrado} = \frac{|\{\text{documentos relevantes}\} \cap \{\text{documentos recuperados}\}|}{|\{\text{documentos relevantes}\}|}$$

8.3.2.4 Medida F (f-measure)

La medida F es un balance de la precisión y el recobrado:

$$F = \frac{2 \cdot \text{Precision} \cdot \text{Recobrado}}{(\text{Precision} + \text{Recobrado})}.$$

8.3.2.5 ¿Qué se va a utilizar?

Es importante entender que para una búsqueda es difícil saber cuál es la cantidad de documentos relevantes que existen en el corpus, ya que el mismo puede contar no solo con millones de entradas sino que además no existe una medida que nos diga qué tan relevante es un documento para una búsqueda en particular más allá de la proporcionada por el propio motor de búsqueda. Por lo tanto, en general sólo se utiliza la medida de precisión donde se determina si un documento es o no relevante “a ojo” o comparándolo con los resultados devueltos por algún otro motor de búsqueda.

8.3.3 Búsquedas a realizar

Para poder medir la precisión de nuestro buscador, definimos búsquedas sobre 20 temas distintos que han sido de relevancia en el 2013 y 5 nombres de políticos para poder realizar las búsquedas de opiniones. Todas las búsquedas se pueden ver en el anexo de evaluación.

Con el objetivo de realizar una evaluación más pertinente y no tomar solamente en cuenta la precisión incluimos cuatro reglas:

> **1-** Para los resultados, sólo se considerará aquellas búsquedas donde realmente había algún resultado pertinente para mostrar. Esto es, por ejemplo, en el caso de que busquemos qué opinó Astori sobre “muerte de Chávez” donde sabemos que en realidad Astori no emitió ninguna opinión en la prensa escrita sobre este tema. Para determinar esto, realizaremos una búsqueda en google, que si bien no tiene la

capacidad de buscar opiniones, nos permite ver si existen noticias donde aparezcan todos los términos buscados, en este ejemplo la búsqueda sería 'astori muerte de Chávez' y en caso de que existan revisaremos si se emitió alguna opinión sobre el tema.

Podríamos considerar que en realidad si Astori no emitió ninguna opinión sobre la muerte de Chávez, nuestro buscador no debería mostrar ningún resultado. El problema es que se configuró el motor de búsqueda para que no sea tan estricto en las búsquedas con el objetivo de mejorar el recall en detrimento de la precisión.

2- Se agregará una variable "cantidad de resultados obtenidos mayor que 5" que nos ayudará, luego de obtener los datos acerca de la precisión, a visualizar mejor los resultados.

3- Se agregará una variable "cantidad de resultados obtenidos mayor que 8".

4- Se agregará una variable que sea "el primer resultado fue relevante" que nos ayudará a determinar qué sucedería con la precisión si para cada búsqueda se mostrase solamente un único resultado (el más relevante).

Creemos importante volver a recordar que en esta evaluación no se está midiendo el recall pues no poseemos ninguna otra herramienta de búsqueda de opiniones con la cual comparar y por esta razón se agregan estas reglas que nos ayudan a tener una visión más global de los resultados.

Recordamos aquí también que existe una diferencia fundamental entre nuestro buscador de opiniones y otros buscadores. Mientras que en el buscador de opiniones ordenamos los resultados de búsqueda por la fecha en la cual fue emitida la opinión, en la mayoría de los buscadores los resultados se ordenan por relevancia con lo cual se le da un orden de importancia a los resultados que aparecen en las búsquedas.

Por ejemplo, cuando realizamos una búsqueda en Google, en general este devuelve cientos de miles de resultados, pero para evaluar al buscador tendría más sentido fijarse solamente si los primeros resultados obtenidos (principalmente los primeros 2 o 3 resultados) fueron realmente relevantes.

Por otro lado, en el buscador de opiniones que implementamos, al mostrar los resultados al usuario ordenados por fecha, le damos una relevancia equitativa a todos los resultados, a pesar de que al realizar la búsqueda en Solr estos sí estaban ordenados por relevancia. Por lo tanto, para mejorar la precisión de nuestro buscador, mostramos una cantidad $\log_{1.5}(\text{cantidadDeResultadosRecuperados} + 1)$ de resultados. Esto es, de todos los resultados recuperados por el motor de búsqueda, solamente mostramos una fracción de los mismos, donde dicha fracción se calcula de una forma logarítmica.

Si se quiere obtener más resultados, se puede utilizar la búsqueda avanzada para establecer la cantidad exacta. Sin embargo, realizaremos nuestra evaluación del buscador sobre la fracción de resultados anteriormente mencionada, pues son los resultados devueltos por defecto.

8.3.4 Resultados obtenidos

Se realizaron las 100 búsquedas (20 temas distintos para 5 políticos distintos), obteniendo más de 400 resultados para las mismas. Cada uno de estos resultados fue analizado con el objetivo de determinar si era o no relevante a la búsqueda que estábamos realizando. En el anexo de

evaluación se puede encontrar una tabla donde se indica la precisión obtenida para cada una de las búsquedas. Además, también se incluyen dos tablas más donde para cada una de las búsquedas se indica si hubo más de 5 resultados y si el primer resultado fue relevante.

La precisión obtenida para estas búsquedas fue de un 76% en promedio. Por otro lado, la precisión promedio obtenida para aquellas búsquedas donde se habían obtenido más de 5 resultados fue de un 79%. Además, también se realizó el cálculo sobre aquellas búsquedas que retornaron más de 8 resultados, obteniendo un 85% de precisión en promedio.

A la vista de estos resultados, podemos afirmar que cuantas más opiniones haya emitido una persona sobre un determinado tema, más baja será la probabilidad de que el buscador de opiniones devuelva resultados irrelevantes.

Si estudiamos ahora la cantidad de búsquedas cuyo primer resultado fue relevante, podemos ver que lo fueron el 91% de las mismas. Esto es, en las búsquedas donde existía alguna opinión relevante para mostrar, la primer opinión retornada fue relevante. Esto es importante ya que en el caso de que quisiéramos mejorar la precisión de nuestro sistema podríamos llegar hasta a un 91% de precisión retornando únicamente la opinión más relevante en detrimento de buscar un mayor recall.

Por otro lado, solamente 2 de las búsquedas (un 2%) retornaron un resultado vacío cuando existía una opinión al respecto del tema buscado sobre la persona buscada. Esto es, en Google encontramos una noticia donde la persona emitía una opinión sobre el tema pero nuestro buscador no retornó nada. En ambos casos, luego de analizarlos, determinamos que la razón fue que la noticia donde esa opinión aparecía no había sido crawleada por nuestro sistema y por lo tanto no se encontraba en el corpus. Esto puede verse como una evaluación parcial acerca del recall.

8.3.5 Análisis de los problemas observados

Al analizar los resultados obtenidos en la evaluación de las búsquedas, pudimos observar que son básicamente dos los problemas que llevan al buscador a mostrar opiniones irrelevantes.

El primero es que arrastramos el error que venía del proyecto de reconocimiento de opiniones, donde se tenía una precisión en el reconocimiento de la fuente de un 81%. Principalmente el problema se da cuando el reconocedor de opiniones le asigna una fuente incorrecta a la opinión. Esto es un problema importante para nosotros ya que, por así decirlo, el buscador está poniendo palabras en boca de alguien que en realidad no emitió esa opinión.

El segundo problema que detectamos es que en muchos artículos de prensa en la web se habla sobre más de un tema dentro de la misma página. La mayoría de las veces estos temas poseen alguna relación entre sí pero son independientes. Esto genera que cuando se realiza la búsqueda de opiniones de, por ejemplo, “mujica” sobre “la legalización del aborto” aparece una opinión de mujica acerca de la legalización de la marihuana. Si se va a la noticia original donde apareció la opinión, se puede ver que se habla tanto de la ley sobre la legalización del aborto como de la ley sobre la legalización de la marihuana.

Recordamos que este último problema se da ya que el motor de búsqueda no sólo se fija las palabras que aparecen en la opinión sino que también el contexto en el cual esta se da, es decir las palabras que aparecen en el artículo de la noticia de la cual se extrajo la opinión.

Otro problema, que si bien apareció en menor medida en las búsquedas de prueba realizadas, es el de la ambigüedad. Este problema siempre aparece en los sistemas de Procesamiento del

Lenguaje Natural y se da debido a que el lenguaje humano es ambiguo, una misma palabra puede tener diferentes significados en diferentes contextos. En nuestro caso, esto nos afecta a la hora de realizar las búsquedas ya que se pueden encontrar resultados que no son los que se esperaba en un principio al realizar la búsqueda. Por ejemplo, al realizar la búsqueda de opiniones de "Mujica" sobre "inflación", una de las opiniones encontradas fue "Cuanto más inflación periodística le demos al tema , en lugar de mejorar lo agravamos". Nosotros en realidad estábamos buscando una opinión sobre la inflación en la economía y por lo tanto tenemos que catalogar este resultado como incorrecto.

8.3.6 Posibles soluciones a los problemas observados

Para mejorar la precisión obtenida por el reconocedor de opiniones al reconocer la fuente de la opinión, podría usarse el sistema de aprendizaje automático basado en CRF propuesto en (6). Utilizar esto nos permitiría mejorar la precisión del reconocimiento de la fuente hasta alcanzar un 93% según lo indicado en el citado texto.

Con respecto al segundo problema (más de un tema por artículo), creemos que podría solucionarse al menos parcialmente utilizando alguna técnica de Topic Segmentation como la sugerida en (48). Si utilizando esto pudiésemos tener el artículo periodístico partido en partes según su tema, podríamos realizar las búsquedas del contexto de la opinión sin tener en cuenta el resto del artículo que habla sobre un tema distinto, lo cual aumentaría la precisión.

8.4 FUENTES SUGERIDAS

8.4.1 Introducción

Si bien no realizaremos una evaluación formal sobre esta parte de nuestro sistema, sí comentaremos algunos detalles acerca de los resultados obtenidos y posibles mejoras.

8.4.2 Resultados obtenidos

En general los resultados obtenidos con esta herramienta creemos que fueron buenos aunque no desarrollamos ninguna métrica para evaluarlos más allá del test de usuario que comentaremos más adelante.

Uno de los resultados que creemos más interesantes es que las fuentes sugeridas nos permiten saber qué fuente fue la que recibió más atención de la prensa sobre un determinado tema. Por ejemplo, si buscamos el tema "inflación" veremos que la primer fuente sugerida es "Lorenzo", el actual ministro de economía. Por otro lado, si buscamos por ejemplo opiniones sobre "Patrice Evra" veremos que la primer fuente sugerida es Suárez debido a las acusaciones de racismo que recibió este último. Dentro de las fuentes sugeridas para este último ejemplo también podemos encontrar a Lugano y Bauzá (presidente de la AUF).

8.4.3 Problemas encontrados y posibles mejoras

El principal problema de nuestra implementación es que con el objetivo de unificar las fuentes, sólo dejamos los apellidos. Esto lleva a que las personas que tienen apellidos comunes se mezclen. Por ejemplo, si se hace una búsqueda sobre el tema "Malvinas" una de las sugerencias sobre las fuentes es "Fernández", pero al realizar la búsqueda con ese apellido encontramos opiniones tanto de la presidente argentina Cristina Fernández como del ministro de defensa Fernández Huidobro.

Para poder solucionar este problema deberíamos encontrar algún algoritmo que nos permita unificar las fuentes pero sin perder los nombres. Por ejemplo que “Mujica”, “el presidente Mujica” y “José Mujica” unifiquen a “José Mujica”; y que por otro lado “Fernández” unifique a “Cristina Fernández” cuando en la noticia se hace referencia a ella y a “Fernández Huidobro” cuando la noticia hace referencia al ministro.

8.5 TEMAS DE LA SEMANA

8.5.1 Introducción

Para evaluar esta parte de nuestro sistema utilizamos el juicio experto de un periodista (Gabriel Mordecki) que evaluó a mano los temas de la semana retornados para 20 semanas del 2013. Para esto se tuvo en cuenta tres puntos:

1. La cantidad de temas mostrados que no eran realmente relevantes en la semana.
2. La cantidad de temas que no fueron mostrados pero que eran relevantes en esa semana.
3. La cantidad total de temas mostrados por el sistema.

Con estos parámetros vamos a poder calcular tanto la precisión como el recall de nuestra implementación.

8.5.2 Resultados obtenidos

La tabla completa de la evaluación del sistema puede verse en el anexo de evaluación.

Se obtuvo un 86% de precisión y un 84% de recall en promedio para los resultados de las 20 semanas evaluadas. Si bien no nos habíamos impuesto un objetivo acerca de qué precisión o recall debía alcanzar nuestro sistema, creemos que los resultados alcanzados son más que satisfactorios.

Por otro lado, no contamos con otro sistema que realice una tarea similar y mucho menos para las noticias de Uruguay. Por lo tanto no podemos realizar una comparación de los resultados obtenidos con otros sistemas.

8.5.3 Problemas encontrados

Al estar utilizando un algoritmo de clustering, es complicado lograr entender por qué una noticia quedó en uno u otro cluster o realizar un análisis profundo acerca de por qué apareció un tema y otro no. Sin embargo, algo que pudimos notar a partir de la evaluación es que en aquellas semanas donde se obtuvieron los peores resultados, también eran las que tenían la menor cantidad de noticias indexadas.

Por lo tanto, el hecho de no tener el 100% de las páginas crawleadas probablemente está afectando negativamente el funcionamiento del sistema, ya que para realmente saber cuáles fueron los temas más importantes de una semana, es necesario tener todas las noticias que fueron publicadas durante esa semana.

8.6 TEST DE USUARIO

8.6.1 Introducción

Con el objetivo de tener una evaluación más general acerca del sistema, decidimos realizar un test de usuario donde se le pide al usuario que utilice y puntúe del 1 al 10 cada una de las características del sistema.

Esta encuesta consiste de 4 partes. Primero se le pide al usuario que realice 3 búsquedas predefinidas por nosotros y que evalúe los resultados obtenidos (las búsquedas realizadas se encuentran en el anexo de evaluación). Luego se le pide al usuario que realice varias búsquedas a su antojo y que evalúe los resultados que obtuvo. A continuación se le pide al usuario que evalúe el sistema de otras fuentes que opinaron sobre un tema con las búsquedas recién realizadas. Por último, se le pide al usuario que utilice la herramienta para ver los temas de la semana y la evalúe.

8.6.2 Resultados obtenidos

La encuesta fue realizada a 10 personas distintas y la tabla con los resultados obtenidos se encuentran en el anexo de evaluación.

En promedio, el buscador de opiniones obtuvo una puntuación de 8.1, las fuentes relacionadas 9.3 y los temas de la semana 8.7. Por otro lado, los resultados para las búsquedas predefinidas #1, #2 y #3, obtuvieron un puntaje de 8.7, 7.5 y 8.5.

Las precisiones, estimadas por nosotros a priori, de las búsquedas #1, #2 y #3 son de 92%, 73% y 83% respectivamente.

Se eligió para la búsqueda #2, una búsqueda que daba resultados no tan buenos, según nuestros cálculos, como en las otras dos búsquedas. Esto lo hicimos con la intención de observar si realmente los usuarios se estaban dando cuenta de si los resultados eran o no buenos.

Comparando con los puntajes otorgados por los usuarios, pudimos verificar que su criterio fue “similar”, por decir de alguna forma, al nuestro al realizar la evaluación.

Más allá de los puntajes obtenidos, creemos que la parte más interesante del test fue ver a los usuarios utilizar el sistema. Muchos de ellos son ajenos al campo de la ingeniería y por lo tanto pueden considerarse como usuarios “comunes”.

Lo primero que nos llamó la atención fue la dificultad que tuvieron los usuarios en comprender qué era una opinión. Por ejemplo, uno de ellos, al decirle que lo que iba a utilizar es un buscador de opiniones, creyó que al realizar la búsqueda el sistema daría un resumen claro y preciso acerca de la opinión que tuvo una persona sobre un determinado tema. Mientras que lo que obtuvo con la búsqueda fue una serie de opiniones extraídas de medios de prensa.

Por otro lado, notamos que al realizar búsquedas se frustraron con mucha rapidez sin intentar realizar otras búsquedas similares, por ejemplo cambiando algún término de la búsqueda o utilizando la búsqueda avanzada. Intuimos que esto se debe a que el único buscador que están acostumbrados a utilizar es el de Google, donde en general no es necesario realizar más de una o dos búsquedas para encontrar resultados relevantes.

Otro aspecto a destacar es que la herramienta de “fuentes sugeridas” fue la mejor puntuada con respecto a las demás.

9 CONCLUSIONES

9.1 ANÁLISIS

Se construyó un sistema que permite realizar búsquedas de opiniones en textos de prensa. Estos textos residen dentro de un corpus que fue extraído de los sitios web de diversos medios de prensa uruguayos.

A partir de la evaluación realizada se pudo determinar que se alcanzó un 76% de precisión en las búsquedas en esta primera aproximación al problema. Es destacable que no existe ningún sistema que realice una tarea similar al que se construyó y, por lo tanto, no podemos comparar los resultados que obtuvimos con otros buscadores.

Para mostrar los resultados de las búsquedas se creó una interfaz web. En donde, a partir de la búsqueda de un usuario por una fuente y un tema, se despliega en pantalla una línea de tiempo con las opiniones de dicha fuente sobre el tema ingresado. Es posible navegar en la línea de tiempo generada para ver las distintas opiniones que emitió una persona a través del tiempo, siendo interesante ver si existen contradicciones entre las mismas.

Al construir la interfaz se puso especial énfasis en que fuese atractiva para el usuario, utilizando tecnologías recientes para la creación de páginas web como HTML5 y CSS3.

Al realizar una búsqueda, también se le ofrece al usuario sugerencias sobre otras fuentes que opinaron sobre el tema ingresado. Esto, según pudimos constatar en los test de usuario realizados, les pareció de gran utilidad a los usuarios.

Por otro lado, también se construyó una herramienta que permite saber cuáles fueron los temas de los que más se habló en la prensa en una semana determinada. Para esta parte del sistema, según una evaluación realizada por un periodista, se obtuvo un 86% de precisión y un 84% de recall.

9.2 APORTES DEL PROYECTO

Los resultados obtenidos permiten disponer de una herramienta para la búsqueda de opiniones en los medios de prensa uruguayos. En un futuro, esta herramienta puede extenderse fácilmente para abarcar a otros medios de prensa de otros países de habla hispana. A partir de la evaluación realizada fue posible demostrar que cualquier usuario puede utilizar el sistema desarrollado sin ningún tipo de problemas, logrando en gran medida encontrar lo que estaban buscando.

Por otro lado, utilizando la herramienta para ver los temas más importantes de una semana determinada, se puede realizar una recorrida acerca de las noticias más relevantes en un determinado período de tiempo. Algo que no permite realizar ninguna otra herramienta para la prensa uruguaya.

Dentro del aporte académico a los participantes del proyecto, se puede destacar la gran cantidad de herramientas que se tuvieron que aprender a utilizar, las cuales se encuentran en gran parte enumeradas en el capítulo 10. Por otro lado, el área en el que se basa este proyecto, la Recuperación de Información, no es un área en la que se tuviera experiencia anteriormente puesto que no es parte de la formación en la Facultad de Ingeniería. Por lo

tanto, se tuvo que realizar cierta investigación sobre el estado del arte de esta área para interiorizarse en el tema.

Debido al hecho de que no existe otra herramienta que permita realizar búsquedas de opiniones para el idioma español, se considera que este proyecto es de alta relevancia para el área de investigación en el cual se desarrolla.

Para obtener una evaluación más objetiva sobre los aportes de este proyecto, se consultó al periodista Sebastián Cabrera, el cual trabaja para el diario El País. Según Cabrera, "Así como está creo que ya sería útil para un periodista. Si yo puedo saber todo lo que dijo Tabaré Vázquez sobre la marihuana, por ejemplo, me parece muy útil en el caso de que vaya a armar una nota sobre el tema". Por otro lado, también dijo: "también estaría bueno que incluyera otros sitios como montevideo.com, 180, Espectador, Brecha y Búsqueda en lo que es de acceso gratuito". Esto último se dejará como trabajo a futuro para una posible mejora del proyecto.

9.3 POSIBLES MEJORAS Y TRABAJOS A FUTURO

9.3.1 Crawling de la prensa uruguaya

Si bien se logró generar un amplio corpus de noticias extraídas de los tres medios de prensa de los que se propuso descargar información desde un principio; según la evaluación realizada todavía falta crawlear parte de estos medios.

Por otro lado, como trabajo a futuro nos podemos proponer integrar al sistema otros medios de prensa tanto uruguayos como internacionales. En principio descartamos esta posibilidad para acotar el área de desarrollo de proyecto, pero no debería implicar grandes dificultades agregar nuevos medios debido a que se diseñó el sistema de forma de poder realizar esta tarea de una forma relativamente sencilla.

Para realizar esta tarea habría que realizar los siguientes tres pasos:

1. Agregar la URL del nuevo medio a nuestro crawler para que comience a descargar las páginas.
2. Agregar expresiones regulares para obtener la fecha de publicación de las noticias para este medio.
3. Entrenar el clasificador que separa los artículos periodísticos del resto de las páginas (portadas, etc).

9.3.2 Interfaz de usuario

A partir de los test de usuario pudimos observar que las personas están muy acostumbradas a utilizar Google para buscar información y se sentían desorientadas al recibir los resultados de búsqueda ordenados por fecha en vez de por relevancia. Como trabajo a futuro podemos proponer crear una segunda interfaz donde los resultados de búsqueda estén ordenados por relevancia y no por fecha.

Por otro lado, los usuarios se frustran rápidamente cuando no encuentran lo que están buscando con la primera búsqueda realizada y no saben cómo hacer para mejorarla cambiando las palabras. Atribuimos nuevamente esto a que los usuarios están acostumbrados a utilizar herramientas de búsqueda como Google donde en general la primer búsqueda ya arroja resultados relevantes. Para paliar esta situación podemos proponer que el sistema

retorne únicamente la opinión más relevante, que como se indicó en el capítulo de evaluación, tiene un 90% de precisión.

Otra mejora que podemos proponer es realizar un estudio más profundo de cómo los usuarios utilizan el sistema para ubicar mejor las diferentes herramientas que se le presentan. Por ejemplo, en el test de usuario notamos que la mayoría de las personas no había visto las sugerencias de "otras fuentes que opinaron sobre este tema" por aparecer más abajo en la pantalla.

9.3.3 Buscador de opiniones

A partir de los resultados obtenidos en la evaluación, pudimos ver que los errores en las búsquedas tienen tres causas principales:

1. No existe lo que se está buscando y el buscador retorna malos resultados.
2. Problemas en el reconocimiento del tema sobre el cual se emite una opinión.
3. Problemas en el reconocimiento de la fuente de una opinión.

El **problema 1** es un problema de precisión, ya que si no existe ningún resultado relevante, el buscador no debería retornar ningún resultado. El problema es que con el objetivo de mejorar el recall en las búsquedas a veces se terminan trayendo resultados que no son pertinentes. Para mejorar esto debería buscarse alguna forma de restringir más los resultados que aparecen pero sin dañar las búsquedas donde sí existen más resultados para mostrar.

El **problema 2** es principalmente que en un mismo artículo periodístico muchas veces se habla sobre más de un tema. Nuestro buscador, para identificar el tema de una opinión, se fija en las palabras que aparecen en todo el artículo de prensa. De aquí surge que cuando se habla de dos temas distintos en una misma noticia, por ejemplo, en una noticia donde se habla acerca de la legalización de la marihuana y el matrimonio igualitario; si se realiza una búsqueda de opiniones de Mujica sobre el matrimonio igualitario, también pueden aparecer opiniones de Mujica sobre la legalización de la marihuana.

Para solucionar este problema, proponemos tratar de analizar el espacio de texto donde aparece una opinión y tomar en cuenta únicamente las palabras que aparecen dentro de una determinada ventana alrededor de la expresión de opinión.

Otra aproximación al problema sería intentar separar dentro de un mismo artículo de prensa, las secciones donde se habla de uno u otro tema y considerar únicamente la opinión dentro de la sección del tema donde aparece. Para esto creemos que podría utilizarse alguna técnica de Topic Segmentation como la sugerida en (57).

El **problema 3** es que debido a que nuestro proyecto se basa en la utilización de otros módulos desarrollados anteriormente, arrastra los errores que estos tenían. En particular, en el módulo de identificación de opiniones, se tenía una precisión en el reconocimiento de la fuente de un 81%. El problema se da cuando el reconocedor de opiniones le asigna una fuente incorrecta a la opinión. Esto es un problema importante para nosotros ya que, por así decirlo, el buscador está poniendo palabras en boca de alguien que en realidad no emitió esa opinión.

Para mejorar la precisión obtenida por el reconocedor de opiniones al reconocer la fuente de la opinión, podría usarse el sistema de aprendizaje automático basado en CRF propuesto en (6). Utilizar esto nos permitiría mejorar la precisión del reconocimiento de la fuente hasta alcanzar un 93% según lo indicado en el citado texto.

9.3.4 Fuentes sugeridas

El principal problema de nuestra implementación es que con el objetivo de unificar las fuentes, sólo dejamos los apellidos. Esto lleva a que las personas que tienen apellidos comunes se mezclen. Por ejemplo, si se hace una búsqueda sobre el tema "Malvinas" una de las sugerencias sobre las fuentes es "Fernández", pero al realizar la búsqueda con ese apellido encontramos opiniones tanto de la presidente argentina Cristina Fernández como del ministro de defensa Fernández Huidobro.

Para poder solucionar este problema deberíamos encontrar algún algoritmo que nos permita unificar las fuentes pero sin perder los nombres. Por ejemplo que "Mujica", "el presidente Mujica" y "José Mujica" unifiquen a "José Mujica"; y que por otro lado "Fernández" unifique a "Cristina Fernández" cuando en la noticia se hace referencia a ella y a "Fernández Huidobro" cuando la noticia hace referencia al ministro.

9.3.5 Temas de la semana

Esta parte de nuestro sistema, si bien es la que creemos que tuvo un mejor rendimiento comparada con el resto, también es la más difícil de analizar. Al estar utilizando un algoritmo de clustering, es difícil entender por qué una noticia quedó en uno u otro cluster o por qué una noticia que debería ser claramente un tema de la semana, no aparece.

Lo que podemos sugerir para mejorar esta parte del sistema, más allá del análisis caso por caso, es utilizar otros algoritmos de clustering distintos al EM y tratar de comparar "a ojo" si los resultados que se obtienen son mejores.

Aclaramos aquí que al elaborar el algoritmo que encuentra los temas de la semana ya intentamos utilizar otros algoritmos de clustering: SimpleKMeans, DBSCAN y Hierarchical Clustering pero los descartamos debido a que los resultados con esos dichos algoritmos fueron pobres.

Por otro lado, como se mencionó en el capítulo de evaluación, tener crawleado el 100% de las páginas web de los medios de prensa probablemente mejoraría el funcionamiento del algoritmo que determina cuáles fueron los temas de la semana.

9.4 SISTEMA EN RÉGIMEN

Al momento de redactar este informe, se dejó corriendo todo el sistema implementado en un servidor de la Facultad de Ingeniería. Todos los días el servidor corre el siguiente proceso para tener las opiniones sobre las noticias actualizadas:

1. Se crawlean los sitios de El País, La República y El Observador para encontrar las noticias que hayan sido publicadas en el dia.
2. Se procesan las noticias para extraer las opiniones y el resto de los elementos.
3. Se indexan las opiniones para que las mismas puedan ser buscadas.

Por otro lado, todas las semanas se corre un proceso que genera los "temas de la semana" para los últimos 7 días.

A su vez, se tiene registrado el dominio buscopiniones.com desde donde se puede acceder a la implementación del sistema. Sin embargo, los datos accesibles desde esta versión, no están actualizados y solo se puede encontrar noticias hasta el 30 de setiembre del 2013. Esta instancia está corriendo en un servidor externo a la facultad.

10 HERRAMIENTAS UTILIZADAS

Módulo de identificación de opiniones - Es un proyecto desarrollado por una docente de la Facultad de Ingeniería que permite, dado una entrada de texto, identificar diferentes expresiones de opinión en el mismo. Está especialmente pensado para analizar artículos periodísticos.

Módulo de correferencias - Es un proyecto desarrollado por estudiantes de la Facultad de Ingeniería que, dada como entrada la salida del módulo de identificación de opiniones, retorna como salida un XML donde se resulven las diferentes correferencias que existen dentro de las fuentes de distintas opiniones. Por ejemplo, correferencia la fuente "Mujica" con "El presidente Mujica".

Lucene - Es una biblioteca de software de recuperación de información de código abierto, creada originalmente en Java por Doug Cutting. Es apoyada por la Fundación de Software Apache y se distribuye bajo la licencia Apache Software.

Solr - Es una plataforma de búsqueda empresarial de código abierto del proyecto Apache Lucene. Sus principales características incluyen la búsqueda de texto completo, highlighting, búsqueda facetada, clustering dinámico, la integración con bases de datos, y manipulación de documentos como pdf o word.

Nutch - Es un proyecto de código abierto altamente extensible y escalable que implementa un software de web crawling. El proyecto es mantenido por Apache y está basado en Lucene y Solr.

Boilerpipe - Esta es una librería que proporciona algoritmos para detectar y eliminar el excedente (boilerplate) en torno al principal contenido de texto de una página web. Están especialmente diseñados para trabajar sobre páginas con artículos de prensa y blogs donde el contenido principal de la página es un texto medianamente largo.

HTMLCleaner - Es una herramienta que nos permite trabajar fácilmente con HTML mal formado. Esto fue de especial utilidad cuando utilizamos XPath para extraer la fecha de publicación de un artículo.

JSoup - Proporciona una API muy conveniente para la extracción y manipulación de datos desde HTML, utilizando métodos DOM, CSS, jQuery y similar.

Weka - Es una suite de software para el aprendizaje automático escrita en Java, desarrollada en la Universidad de Waikato, Nueva Zelanda. Weka es un software gratuito disponible bajo GNU General Public License.

Freeling - Es un conjunto de herramientas de análisis para el lenguaje natural, de código abierto, liberada bajo GNU General Public License de la Free Software Foundation.

Bootstrap - Es un conjunto de herramientas gratuitas para crear sitios y aplicaciones web. Contiene HTML y basados en plantillas de diseño CSS para tipografía, formas, botones, navegación y otros componentes de la interfaz, así como las extensiones de JavaScript opcionales.

Flati - Es una plantilla de páginas web diseñadas en HTML5 que fueron construidas utilizando el framework Twitter Bootstrap.

Wordnet - Es una base de datos léxica del Inglés (auque también existen una versión para el español). Sustantivos, verbos, adjetivos y adverbios se agrupan en conjuntos de sinónimos cognitivos (synsets), cada una expresando un concepto distinto. Los Synsets están vinculados entre sí por medio de las relaciones conceptuales semánticas y léxicas.

TimelineJS - Es una herramienta open-source que permite fácilmente crear una línea de tiempo para mostrar en una interfaz de usuario basada en HTML y Javascript. Se utilizó en la creación de la interfaz de usuario para mostrar los resultados de una búsqueda de opiniones ordenada por fecha.

Tomcat - Es un web server que funciona como un contenedor de servlets desarrollado bajo el proyecto Jakarta en la Apache Software Foundation. Tomcat implementa las especificaciones de los servlets y de JavaServer Pages (JSP) de Sun Microsystems. Este servidor fue utilizado para correr la interfaz web.

Jetty - Es un servidor HTTP y un contenedor de Servlets totalmente basado en Java. Jetty es un proyecto de software libre bajo la licencia Apache 2.0. Este servidor fue utilizado para correr las instancias de Solr.

Java - Es un lenguaje de programación de propósito general, concurrente, basado en clases y orientado a objetos que está diseñado específicamente para tener el menor número de dependencias de ejecución posible. Su objetivo es permitir a los desarrolladores de aplicaciones "escribir una vez, ejecutar en cualquier parte" (WORA), lo que significa que el código que se ejecuta en una plataforma no necesita ser recompilado para funcionar en otra.

Python - Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis muy limpia y que favorezca un código legible. Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, usa tipado dinámico y es multiplataforma. Es el lenguaje en el que está implementado el módulo de correferencias.

Prolog - Es un lenguaje para programar artefactos electrónicos mediante el paradigma lógico con técnicas de producción final interpretada. Es bastante conocido en el área de la Ingeniería Informática para investigación en Inteligencia Artificial y PLN. Es el lenguaje en el que está implementado el módulo de identificación de opiniones.

HTML - Es un lenguaje de marcas, muy utilizado en internet, que, junto con el texto, incorpora etiquetas o marcas que contienen información adicional acerca de la estructura del texto o su presentación.

XPath - Es un lenguaje que permite construir expresiones que recorren y procesan un documento XML teniendo en cuenta su estructura. También puede aplicarse a HTML.

Javascript - Es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.

CSS - Es un lenguaje de hojas de estilo utilizado para describir la semántica de presentación (la apariencia y formato) de un documento escrito en un lenguaje de marcas. Su aplicación más

común es en las páginas web de estilo escrito en HTML y XHTML, pero el lenguaje también se puede aplicar a cualquier tipo de documento XML, incluyendo sin formato XML, SVG y XUL.

Netbeans - Es un entorno de desarrollo integrado libre, hecho principalmente para el lenguaje de programación Java. Este fue utilizado para el desarrollo de todo el proyecto.

11 BIBLIOGRAFÍA

1. **Cantidad de páginas web. [En línea] http://www.worldwidewebsize.com/.**

2. ***Introduction to Information Retrieval.*** **Christopher D. Manning, Prabhakar Raghavan, Hinrich Schütze. s.l. : Cambridge University Press. ISBN 978-0-521-86571-5.**

3. Google. [En línea] http://www.google.com/.

4. Google Search Appliance. [En línea] http://www.google.com/enterprise/search/products/gsa.html.

5. PLN. [En línea] http://en.wikipedia.org/wiki/Natural_language_processing.

6. ***Identificación de opiniones de diferentes fuentes en textos en español.*** **Rosá, Aiala. Montevideo, Uruguay : s.n., 2011. ISSN 0797-6410.**

7. ***Resolución de correferencias en expresiones de opinión.*** **Acerenza, Fernando, Rabosto, Macarena y Zubizarreta, Magdalena. Montevideo, Uruguay : s.n., 2012.**

8. Apache Sofware Fundation. [En línea] http://apache.org/.

9. Apache Lucene. [En línea] http://lucene.apache.org/.

10. Lucene en Wikipedia. [En línea] http://en.wikipedia.org/wiki/Lucene.

11. Apache Solr. [En línea] http://lucene.apache.org/solr/.

12. ***Solr 1.4 Enterprise Search Server. Birmingham.*** **Smiley, David y Pugh, Eric. Birmingham, B27 6PA, UK : Packt Publishing Ltd, 2009. ISBN 978-1-847195-88-3.**

13. Cómo funciona Lucene. [En línea] http://www.opensourceconnections.com/2013/05/20/how-does-a-search-engine-work-an-educational-trek-through-a-lucene-postings-format/.

14. Índice invertido. [En línea] http://en.wikipedia.org/wiki/Inverted_index.

15. TF-IDF. [En línea] http://en.wikipedia.org/wiki/Tf%E2%80%93idf.

16. Apache Nutch. [En línea] http://nutch.apache.org/.

17. Apache Hadoop. [En línea] http://hadoop.apache.org/.

18. Solr Cloud. [En línea] http://wiki.apache.org/solr/SolrCloud.

19. Nutch Wiki. [En línea] http://wiki.apache.org/nutch/.

20. java.net. [En línea] https://today.java.net/pub/a/today/2006/01/10/introduction-to-nutch-1.html.

21. ***Técnicas de extracción de información desde fuentes Web.*** **Miguel Ángel, Jaén Gómez. 2007. D1.2-05/07-DEF.**

22. Google news. [En línea] https://news.google.com/.

23. Google News wikipedia. [En línea] http://es.wikipedia.org/wiki/Google_News.

24. News Aggregator. [En línea] http://en.wikipedia.org/wiki/News_aggregator.

25. Nutch Tutorial. [En línea] http://wiki.apache.org/nutch/NutchTutorial.

26. HtmlParseFilter API. [En línea] http://nutch.apache.org/apidocs-1.2/org/apache/nutch/parse/HtmlParseFilter.html.

27. Base64. [En línea] http://en.wikipedia.org/wiki/Base64.

28. Weka. [En línea] http://www.cs.waikato.ac.nz/ml/weka/.

29. JSoup. [En línea] http://jsoup.org/.

30. HTMLCleaner. [En línea] http://htmlcleaner.sourceforge.net/.

31. Kohlschütter, Christian. BoilerPipe. [En línea] http://code.google.com/p/boilerpipe/.

32. *Boilerplate Detection using Shallow Text Features.* Christian Kohlschütter, Peter Fankhauser, Wolfgang Nejdl. Hannover - Germany : L3S Research Center / Leibniz Universität Hannover, 2010. s.n., 2010. WSDM.

33. Librería Jython para conectar Java con Python. [En línea] http://www.jython.org/.

34. JPL, librería para conectar Java con Prolog. [En línea] http://www.swi-prolog.org/packages/jpl/java_api/.

35. Freeling, analizador del lenguaje natural. [En línea] http://nlp.lsi.upc.edu/freeling/.

36. Edismax. [En línea] http://wiki.apache.org/solr/ExtendedDisMax.

37. Facetado. [En línea] http://wiki.apache.org/solr/SolrFacetingOverview.

38. Lista de políticos uruguayos. [En línea] http://es.wikipedia.org/wiki/Categor%C3%ADa:Pol%C3%ADticos_de_Uruguay.

39. Lista de futbolistas uruguayos. [En línea] http://es.wikipedia.org/wiki/Categor%C3%ADa:Futbolistas_de_Uruguay.

40. Highlighting. [En línea] http://wiki.apache.org/solr/HighlightingParameters.

41. Distancia de Levenshtein. [En línea] http://en.wikipedia.org/wiki/Levenshtein_distance.

42. Solr More Like This. [En línea] http://wiki.apache.org/solr/MoreLikeThis.

43. Algoritmo para obtener las palabras claves mediante MLT. [En línea] http://cephas.net/blog/2008/03/30/how-morelikethis-works-in-lucene/.

44. Algoritmo de Expectation-Maximization. [En línea] http://en.wikipedia.org/wiki/Expectation%E2%80%93maximization_algorithm.

45. Flati. [En línea] http://themeforest.net/item/flati-responsive-flat-design-bootstrap-template/4690890.

46. Bootstrap. [En línea] http://getbootstrap.com/.

47. TimelineJS. [En línea] http://timeline.verite.co/.

48. [En línea] 1997. http://link.springer.com/chapter/10.1007/BFb0026725. ISBN 978-3-540-63554-3.