Escuela Politécnica Superior de Jaén

Grado en Ingeniería Informática

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Jaén

Trabajo Fin de Grado

SEGUIMIENTO DE TEMAS EN PRENSA

Alumno: Ismael Garrido Muñoz

Tutor: Prof. D. Arturo Montejo Ráez
Dpto: Informática

Junio, 2016

Universidad de Jaén
Escuela Politécnica Superior de Jaén
Departamento de Informática

Don ARTURO MONTEJO RÁEZ , tutor del Proyecto Fin de Carrera titulado: NOMBRE_PROYECTO, que presenta ISMAEL GARRIDO MUÑOZ, autoriza su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, Junio de 2016

El alumno:

ISMAEL GARRIDO MUÑOZ

Los tutores:

ARTURO MONTEJO RÁEZ

Índice

1. Introducción

En este capítulo se introducirá la motivación del proyecto desarrollado y se detallarán los objetivos y resultados del proyecto y que metodología se aplicará para llegar a estos.

1.1. Propósito y motivación del proyecto

En este proyecto propuesto por Arturo Montejo Ráez se pretende crear un sistema que permita realizar un seguimiento temporal de temas destacados en prensa online aplicando algoritmos de modelado de temas.

Ante la existencia de una cantidad tan grande de medios online es interesante recibir un resumen de los principales temas que se han tratado a lo largo de la semana y poder visualizar la evolución de estos temas a lo largo del tiempo para cada una de las secciones principales.

1.2. Objetivos principales

Los objetivos que principales serán:

- Crear un extractor de información de prensa lo suficientemente flexible como para poder responder ante cambios en los sitios web de los que se extraerá la información de forma sencilla y sin tener que recodificar el proyecto. Debe extraer la información automáticamente.
- Buscar la mejor configuración del algoritmo de modelado de temas para cada sección, obteniendo así los temas más descriptivos posibles
- Aplicar estos algoritmos a las distintas secciones una vez por semana automáticamente.
- Visualizar la información recogida en una interfaz que permita observar la evolución de los temas, visualizando que términos aparecen y desaparecen y midiendo la magnitud del cambio.
- Inferir si un tema ha cambiado lo suficiente como para considerarlo un tema distinto a partir de la magnitud del cambio.

- Crear una plataforma sencilla que facilite el estudio tanto de los algoritmos como de la información resultante de su ejecución.
- Evaluar los requisitos del proyecto para su despliegue en un servidor web.
- Crear una interfaz web que esté disponible desde cualquier dispositivo y se visualice de forma acorde al dispositivo.
- Redacción de la memoria del proyecto.

1.3. Metodología a desarrollar

La metodología que se usará consta de los siguientes elementos:

- Se estudiará cada una de las tecnologías usadas para la ejecución del proyecto así como sus alternativas.
- Se ejecutará el proyecto siguiendo la metodología ágil SCRUM para lo cual el proyecto si dividirá en múltiples fases, teniendo al final de cada fase una de las principales funcionalidades completadas. En el apartado 1.5 se detallará la división de estas fases y la funcionalidad requerida en cada una.
- Se usará el patrón de arquitectura software Modelo-Vista-Controlador, el cual separa datos, lógica e interfaz, facilitando el desarrollo de la aplicación y su mantenimiento.

1.4. Resultados esperados

El resultado deseado es una plataforma que recoja noticias de las fuentes especificadas de forma autónoma, las clasifique y semanalmente resuma la información de cada sección en un conjunto de temas destacados, permitiendo al usuario visualizar la progresión temporal de los temas.

Esta plataforma debe de poder ser administrada de forma sencilla pero flexible para responder a cambios estructurales en los sitios web de los cuales se recogen las noticias.

1.5. Estructura del proyecto

El proyecto se dividirá en **3 fases** principales que reflejaran los objetivos principales del proyecto, estando **cada fase dividida en 2 partes**, una **teórica** en la cual se estudiará cómo enfrentar el problema y se realizarán diversas pruebas para elegir la mejor tecnología y comprobar con pruebas de concepto que cumple en todos los apartados y otra **práctica** en la cual se desarrollará el proyecto usando las herramientas estudiadas en la fase anterior.

Para cada problema concreto se realizará una pequeña prueba de concepto aislada para explorar potenciales soluciones, a estas pruebas se les conoce como spikes. El objetivo de desarrollar estas pruebas aisladas es reducir el riesgo a la hora de integrar una nueva tecnología o afrontar un problema, ya que la solución propuesta puede no cumplir con los objetivos o no encajar con la historia de usuario de SCRUM. Si detectamos a tiempo estos problemas con estas sencillas pruebas podemos buscar una solución a tiempo.

1.5.1. Fase de extracción de datos

Para el desarrollo de la parte teórica de esta fase tendremos que encontrar la mejor forma de localizar los documentos que contienen las noticias y cómo tratar estos documentos para localizar el cuerpo de la noticia que es la parte que nos interesa.

Además se estudiará cómo hacer que el sistema recoja estas noticias de forma autónoma. Se realizarán spikes para comprobar que las soluciones estudiadas son válidas.

Para la parte práctica de la frase se implementará el extractor usando las tecnologías estudiadas, permitiendo que el usuario administre las fuentes de datos, los filtros que se deben aplicar a dichas fuentes para obtener el cuerpo de la noticia y las secciones mediante interfaces CRUD.

1.5.2. Fase de obtención de información

La parte teórica de esta fase consistirá en el estudio de los diversos algoritmos, buscando la configuración que obtenga los mejores resultados para cada sección.

Se estudiará el impacto de los principales parámetros sobre los resultados para posteriormente aplicar este conocimiento y conseguir los mejores resultados en las distintas secciones.

En la parte práctica de esta fase se implementará un sistema que nos permita ejecutar los algoritmos con los parámetros dados y que recoja la información resultante generando informes en los cuales se pueda estudiar el algoritmo y visualizar sus resultados.

Los informes que se generarán tras dos modalidades, la manual y la semanal. Los informes manuales se podrán generar gráficamente desde la interfaz web, con el fin exclusivo del estudio de los algoritmos o de la información de los mismos. Los informes semanales los ejecutará automáticamente el sistema, para lo cual se creará una tarea acorde a este propósito.

1.5.3. Fase de visualización de la información

En la parte teórica de esta fase se estudiará cual es la mejor forma de representar la evolución temporal de los temas y como detectar la evolución de los temas.

Para la parte práctica se desarrollara una interfaz para ver cómo evolucionan los temas y un comparador que nos diga como de similares son dos temas dados para poder enlazar los temas de dos semanas consecutivas.

1.6. Antecedentes

En 2002 Andrew Kachites McCallum crea la herramienta para el procesamiento de lenguaje natural MALLET, la cual implementa el Muestreo de Gibbs que más tarde será necesario para LDA, además ofrecer soporte para pre-procesar, clasificar y marcar información. (McCallum A. K., 2002)

En 2003 David M. Blei publica el artículo “Modeling Annotated Data” donde se crea la que será la base del algoritmo LDA conocido como Correspondence-LDA. (Jordan D. B., 2003)

Ese mismo año David M. Blei publica una implementación del algoritmo LDA en C en su sitio web http://www.cs.princeton.edu/~blei/lda-c/ y publica un artículo en el que explica el algoritmo.

Durante los siguientes años se aplicará el modelado de temas y estos algoritmos a diferentes contextos como en el artículo publicado en 2004, “Probabilistic Topic Decomposition of an Eighteenth-Century American Newspaper” en el cual David J. Newman y Sharon Block descomponen en temas unos 80000 artículos pertenecientes el periódico colonial de EE.UU entre los años 1728 y 1800. (Block, 2006)

En 2008 David Mimno implementará una versión mejorada de LDA en MALLET que generará desarrollada en el artículo “Topic models conditioned on arbitrary features with Dirichlet-multinomial regression”. (McCallum D. M., 2008)

En una conferencia en 2012 David Minmo en la universidad de Princeton definirá los problemas del modelado de temas y como evaluar la calidad de los mismos. En esto basaremos la evaluación de los temas en este TFG. (Mimno, Topic Modeling Workshop: Mimno, 2012)

1.7. Introducción al modelado de temas

El modelado de temas (topic modeling) es una forma concreta de aplicar Minería de textos. Consiste en un conjunto de algoritmos capaces de extraer las relaciones semánticas latentes de un amplio corpus de documentos. Estas relaciones se conocen como temas, y son un conjunto de palabras que suelen aparecer juntas en los mismos contextos.

Si dos palabras co-ocurren normalmente (aparecen normalmente en los mismos contextos), se asume que hacen referencia a los mismos temas. Solo nos quedaremos con aquellas que tengan una tendencia muy marcada, basándose en la distribución de probabilidad de Dirichlet.

El principal algoritmo es conocido como LDA que hace referencia a la distribución y a las relaciones latentes que se buscan con el nombre Latent Dirichlet Allocation.

Un símil sería coger un artículo periodístico, identificar los términos más importantes del artículo y marcar con un color diferente estos términos en base a su temática. Al finalizar, cada uno de los conjuntos representaría un tema. (Blei, Probabilistic Topic Models, 2011)

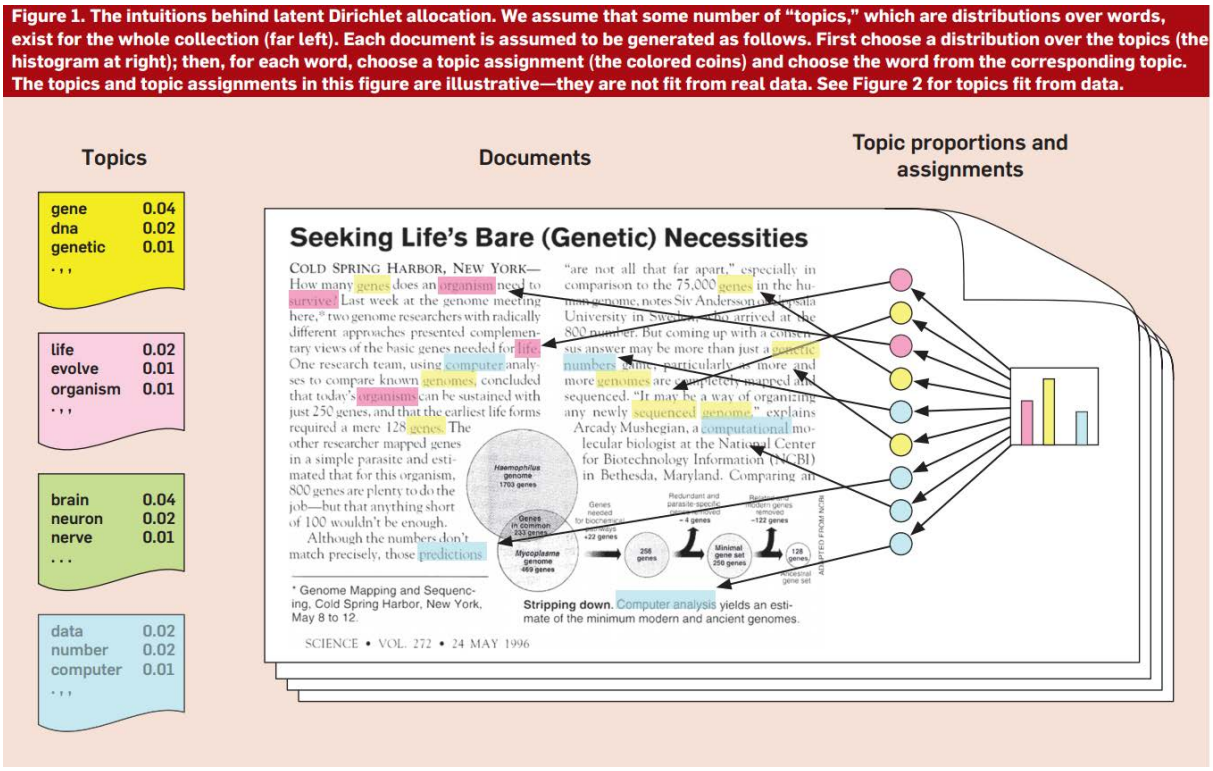

Ilustración 1.1 (McCallum D. M., 2008)

Lo interesante del modelado de temas es que se limitan a ofrecer un simple resumen sino que identifica temas que se tratan a lo largo del corpus uniendo la información de varios documentos. Ofreciendo información que a priori no está a simple vista.

En este proyecto concreto el algoritmo tendrá que identificar los principales temas que se tratan en el corpus de noticias dado. Un ejemplo sencillo para entenderlo es como si aplicamos estos algoritmos sobre un corpus de noticias de la categoría Nacional, el algoritmo identifica el Caso Gürtel conectando un conjunto de noticias por su contenido.

```
caso euros juez valencia investigación madrid dinero tribunal fiscal grupo
ayuntamiento josé fiscalía nacional empresa delito parte audiencia pp
```

1.8. Estructura del documento

El proyecto se estructurará conforme a la guía de estilo publicada en el sitio web de la Escuela Politécnica Superior de Jaén para Trabajos teóricos o experimentales con el único cambio de incorporar los objetivos y antecedentes en la introducción.

En el segundo capítulo se realizará un estudio de las tecnologías usadas en el proyecto y se justificará porque se ha decidido usar cada una de ellas finalmente.

En el tercer capítulo se describirá como se ha ejecutado cada una de las fases siguiendo el proceso que sufre la información desde que es recogida desde la prensa online hasta que es visualizada.

En el cuarto capítulo se estudiarán los algoritmos aplicados así como su parametrización y los efectos de la parametrización sobre los principales aspectos de los algoritmos.

En el quinto capítulo se usará el estudio del cuarto capítulo previo para obtener los mejores resultados posibles intentando enfrentarse a los diversos problemas que surgen en estos algoritmos y a las diferentes secciones en las cuales agrupamos la información.

En el sexto capítulo se obtendrán conclusiones del trabajo y estudio realizado y se propondrán soluciones y mejoras a futuro.

1.9. Integrantes del proyecto

Los integrantes del proyecto serán:

- Arturo Montejo Ráez – Tutor del proyecto
- Ismael Garrido Muñoz – Autor del proyecto.

2. Material

Para que la **aplicación** se ejecute será necesario tener una **máquina** conectada a **internet**. En la máquina estará instalado un **sistema operativo** que ejecutará la aplicación **PHP** a través de un **servidor web**. La aplicación web se encargará de ejecutar las tareas que deben repetirse periódicamente.

Tendremos que elegir un sistema operativo adecuado y un framework PHP sobre el cual construir la aplicación. El usuario podrá acceder a la interfaz de la aplicación a través de un navegador web, pues la interfaz es la de una aplicación web. Será necesario encontrar una interfaz adecuada (una plantilla o framework css) y librerías javascript que mejorarán la interacción.

2.1. Elección del sistema operativo

El sistema operativo elegido será la última versión estable de Ubuntu en su versión para servidor. En el momento del desarrollo la versión en concreto es Ubuntu server 14.04.4 LTS (Trusty Tahr).

Es un sistema operativo gratuito y Open Source compatible con todas las tecnologías que se describirán a continuación así que será el elegido.

2.2. Elección del servidor web

Los principales servidores webs disponibles son Apache y Nginx. En este desarrollo se usará Nginx aunque podría usarse Apache sin ningún problema. Ambos servidores tienen ventajas el uno sobre el otro pero ninguna es significativa para este proyecto (https://www.digitalocean.com/community/tutorials/apache-vs-nginx-practical-considerations).

La única variante a tener en cuenta será la forma de reescribir las rutas. Se proporcionará la configuración adecuada para cada servidor en la guía de instalación.

2.3. Elección del framework PHP

Se desarrollara la aplicación PHP sobre un framework para aprovechar las ventajas que estos proporcionan:

- Un framework te proporciona una estructura ya organizada sobre la cual montar tu aplicación, no es necesario preocuparse del orden en el sistema de ficheros ni la lógica de la aplicación ya que los frameworks la dividen.
- No es necesario realizar implementaciones de una gran multitud de componentes, pues estos ya están implementados y listos para usar con una interfaz simplificada.
- Protección contra vulnerabilidades y fallos de seguridad conocidos.

Para que un framework sea adecuado para nuestro desarrollo tendrá que cumplir las siguientes características:

- Un desarrollo activo, que de soporte para solucionar errores en el framework o en su seguridad.
- Una comunidad activa (Foros de soporte, artículos en blogs, etc.)
- Un documentación lo más completa posible.
- Debe seguir el patrón MVC

2.3.1. CodeIgniter

Ventajas:

- Sencillo de usar, buena curva de dificultad.
- Documentación buena y muy completa.
- Compatible con una gran cantidad de sistemas de gestión de base de datos de forma nativa.
- Poco consumo de memoria (256MB recomendados)
- Gran cantidad de herramientas

Desventajas:

- Carece de herramientas clave para facilitar el desarrollo (Autenticación por ejemplo)
- Es necesario construir un sub-framework extendiendo CodeIgniter o instalando una gran cantidad de plugins para poder empezar el desarrollo de un proyecto real. No está listo para desarrollar una aplicación nada más empezar.
- Carece de ORM real, las tablas de la base de datos no se mapean con clases realmente.
- Precisa una importante configuración inicial para tener todo funcionando, principalmente para que funcionen las rutas.

No incentiva la separación de Controlador y Vista ya que permite ejecutar PHP en la vista de forma directa sin ofrecer una capa de abstracción adecuada. Es necesario requerir a un tercero como Twig.

2.3.2. CakePHP

Ventajas:

- Seguridad integrada
- Gran manejo de sesiones

Desventajas:

- El proyecto que integra CakePHP y MongoDB lleva 2 años sin actualizarse.

2.3.3. Symfony

Ventajas:

- Es el framework más completo en cuanto a componentes y herramientas.
- Muy buena documentación
- Releases LTS.
- Muy Robusto

2.3.4. Zend

Ventajas:

- Alto rendimiento
- Cursos y certificaciones oficiales

Desventajas:

- No recomendado para proyectos grandes
- DE propio (Zend Studio) no gratuito

2.3.5. Laravel

Este es el framework elegido por tener una buena combinación entre funcionalidad y sencillez.

Laravel monta su framework apoyándose en una gran cantidad de los paquetes desarrollados para Symfony y Doctrine, gracias a esto consigue tener una gran cantidad de herramientas y funcionalidades pero no las expone de forma directa para su uso sino que crea capas de abstracción para simplificar el uso de estos componentes en el framework.

Ventajas:

- Uso de paquetes de terceros bien mantenidos que se pueden actualizar de una forma muy sencilla a través de Composer. Adicionalmente también se puede incluir nueva funcionalidad usando Composer.
- Herramienta de comandos “Artisan” que simplifica el desarrollo en el framework generando plantillas con las distintas clases que puede hacer falta crear ejecutando un comando sencillo.
- Buena gestión de la base de datos incorporando el concepto de “Schema” que permite crear tablas/colecciones desde el propio código, el concepto de “Migraciones” que permite actualizar la estructura de la base de datos

desde código sin necesidad de recurrir a consultas SQL, y los “Seeder” que alimentan la BBDD permitiendo realizar cualquier tipo de pruebas sobre las mismas y posteriormente borrar todo el contenido y reinsertarlo limpio ejecutando un comando.

- ORM “Eloquent” que abstrae las tablas como clases y las operaciones que implican a varias tablas como relaciones entre las clases.
- Potente sistema de plantillas llamado Blade que permite mostrar la información en las vistas sin usar nada de PHP pasando por un lenguaje intermedio que se compilará en PHP. Entre otros el lenguaje intermedio soporta bucles básicos (for, while…), bucles avanzados (foreach, forelse), herencia entre plantillas, paso de parámetros entre las mismas, etc.
- Autorización y Autenticación incorporadas.

Desventajas:

- Mayor consumo de memoria (1GB recomendado)

2.4. Elección del sistema de gestión de base de datos

Se usará una base de datos NoSQL (Not Only SQL) para recoger la información de los medios y almacenar los informes generados, la elegida es MongoDB.

En un estudio se compara MongoDB con un SGBD basado en SQL como MySQL vemos que el rendimiento es mejor para operaciones sencillas en las que no intervengan múltiples tablas. Cuando intervienen múltiples tablas o se hace uso de funciones de agregación el rendimiento es mucho peor. (Regueiro, 2012)

La principal operación será la de inserción, si observamos los resultados de un estudio podemos observar la diferencia:

	MySQL	MongoDB
70.000 users	12s	3s
70.000 IPs	12s	3s
1.300.000 domains	4m 36s	58s
70.000 unique users with indexes	28s	23s
70.000 unique IPs with indexes	31s	23s
1.300.000 unique domains with indexes	14m 13s	8m 27s
1.000.000 log entries	28m 14s	12m 7s
5.000.000 log entries	2h 10m 54s	1h 03m 53s
10.000.000 log entries	3h 27m 10s	1h 59m 11s
30.000.000 log entries	10h 18m 46s	5h 55m 25s

Tabla 2.1

La velocidad de inserción es entre 2 y 4 veces mayor aproximadamente.

Sin embargo si observamos los tiempos para un conjunto de pruebas usando funciones de agregación, vemos que los tiempos son considerablemente peores.

	MySQL	MongoDB
10 dominios más visitados con totales de visitas	2m 37s	13m 13s
10 dominios más visitados en la segunda quincena de Junio	17m 43s	52m 39s
10 usuarios con más accesos	3m 53s	24m 02s
Media de tráfico en Junio	2m 42s	12m 05s

Tabla 2.2

Otro aspecto a tener en cuenta es que MongoDB escala horizontalmente mucho mejor que MySQL.

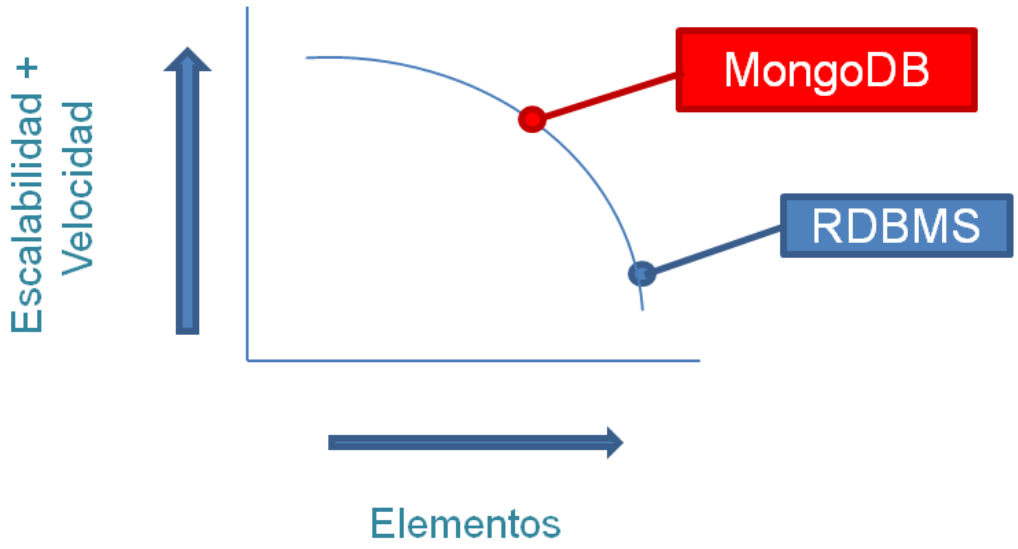

Ilustración 2.1

MongoDB se encuentra en el punto óptimo de escalado, manteniendo la velocidad aunque se aumente el número de elementos.

El uso de una base de datos NoSQL nos permitirá tener unos buenos tiempos a la hora de insertar la información recogida y a la hora de procesarla.

2.5. Elección de la plantilla

La plantilla usada será AdminLTE 2, esta plantilla usa Bootstrap, aportando las siguientes ventajas:

- La aplicación se comportará y dibujará igual en todos los navegadores
- La aplicación se adaptará a dispositivos móviles por sí misma.
- No se cometen errores de usabilidad al usar componentes ya testeados.
- Posibilidad de usar una gran cantidad de componentes disponibles
- Creado usando buenas prácticas, lo cual evita problemas al intentar extenderlo.

Uno de los problemas de bootstrap es que hay que personalizar sus estilos para que no parezca una aplicación genérica, pues es un framework muy usado. Al usar una plantilla, estos estilos ya cambian lo suficiente como para hacer esta aplicación distinguible.

Otro problema podría ser tener que aprender a usar los componentes, el grid o los plugins javascript, en este caso eso no será un problema al tener experiencia previa.

2.6. Elección librería javascript

Para gestionar la interacción en la parte cliente será necesario programar en javascript. Para eso se usará la biblioteca jQuery en su versión 1.9.x. La principal ventaja que ofrece es compatibilidad con los principales navegadores, así no solo la interfaz será consistente sino que la interacción también lo será.

Existen alternativas como XUI.JS, Sizzle, Qwery, ZeptoJs, StringJs, que compensan uno de los mayores problemas de jQuery, su tamaño, frente a los 84kb de jQuery, algunas de estas librerías solo usan 40Kb (XUI) o incluso 5Kb(Spring), sin embargo estas librerías no cumplen con las necesidades de los plugins de Bootstrap o bien llevan años sin actualizarse.

2.7. Entorno de desarrollo

Para desarrollar la aplicación se hará uso de una máquina virtual que mantendrá Virtual Box y será gestionada con Vagrant. Vagrant descarga la máquina (Ubuntu/homestead) y configurará todo para ejecutar el servidor con Virtual Box.

Las principales ventajas son:

- Entorno multiplataforma. Se puede arrancar la máquina (Ubuntu) desde cualquier sistema operativo (Windows, Linux, Mac).
- El software necesario se instarla en la máquina virtualizada y no en el sistema anfitrión. En el sistema operativo anfitrión solo será necesario instalar un editor de código, Atom en este caso, dejando el resto de configuración a la máquina virtual.

2.8. Algoritmos

Para la ejecución y prueba de los distintos algoritmos de modelado de temas usaremos el paquete MALLET (Machine Learning for LanguagE Toolkit) programado en Java que nos permitirá parametrizar los algoritmos de una forma sencilla para sus análisis y que ofrecerá información útil sobre cada ejecución para poder analizar todos los aspectos de los mismos.

La alternativa a MALLET es STMT (Standfor Topic Modeling Toolbox), sin embargo parece que este conjunto de herramientas no se encuentra en un desarrollo activo. Su código se puede descargar y mirando la fecha de edición de los ficheros encontramos que los más recientes son de 2011.

Usaremos MALLET ya que se sigue actualizando cómo podemos ver en Github y exíste más información sobre su uso publicada en internet.

3. Especificación de requisitos y planificación

En este capítulo se mostrará la especificación de requisitos basándonos en las historias de usuario. Estas historias de usuario describirán los requisitos de la aplicación y a partir de su análisis se obtendrá el conjunto de tareas que deben desarrollarse. El proyecto se considerará concluido si se cumple con lo especificado en las historias de usuario.

3.1. Historias de usuario

A continuación se especifican las historias de usuario:

ID	Historia de usuario
HU – 1	Como el usuario debo poder configurar las fuentes de datos para que el sistema recoja los datos automáticamente
HU – 2	Como el usuario debo poder configurar las secciones en las cuales se agruparán los datos
HU – 3	Como el usuario debo poder entrar al panel usando usuario y contraseña
HU – 4	Como usuario debo poder visualizar la información recogida por el sistema
HU – 5	Como el autor del TFG debo documentar las tecnologías usadas en la Wiki del proyecto.
HU – 6	Como el usuario debo poder configurar filtros para el extractor de noticias
HU – 7	Como el usuario debo poder supervisar la tarea del extractor
HU – 8	Como un usuario debo poder ejecutar los algoritmos para unos parámetros dados para así poder estudiar el comportamiento de parámetros y algoritmos
HU – 9	Como un usuario debo poder ver un informen a modo resumen de los resultados de la ejecución de los algoritmos para poder realizar el estudio con la información contextualizada.
HU – 10	Como un usuario debo poder ver la lista de informes generados a partir de las ejecuciones de los algoritmos
HU – 11	Como un usuario no registrado debo poder ver la evolución de un tema en una línea temporal para una sección dada
HU – 12	Como un usuario no registrado debo poder ver los informes de los cuales procede dicho tema
HU – 13	Como autor del TFG debo completar una memoria

Tabla 3.1

A continuación se detallarán las tareas que se obtienen del análisis de la historia de usuario.

3.1.1. HU – 1

Historia de usuario:

Como el usuario debo poder configurar las fuentes de datos para que el sistema recoja los datos automáticamente.

Descripción:

Se creará el modelo de datos que almacenará los enlaces a las noticias y la interfaz web que permita realizar las operaciones CRUD sobre esos datos. Además se creará una tarea automática que recoja los enlaces consultando las fuentes RSS especificadas.

Tareas obtenidas:

A continuación se detallan las tareas obtenidas.

ID	Historia de usuario
T – 1	Implementar las operaciones CREATE y READ en Feed
T – 2	Crear la tarea rss:feed que recoja la información de las fuentes dadas
T – 3	Programar la tarea para que se ejecute cada 5 minutos.
T – 4	Implementar la operación UPDATE en Feed
T – 5	Implementar la operación DELETE en Feed
T – 6	Implementar un seeder con información de medios periodísticos para cargarlos por defecto

Tabla 3.2

3.1.2. HU – 2

Historia de usuario:

Como el usuario debo poder configurar las secciones en las cuales se agruparán los datos

Descripción:

Se creará el modelo de datos que almacenará las secciones en las cuales se agruparán las noticias. El usuario tendrá una interfaz CRUD para actuar sobre ellas.

Tareas obtenidas:

A continuación se detallan las tareas obtenidas.

ID	Historia de usuario
T – 7	Implementar las operaciones CREATE y READ en Section.
T – 8	Implementar la operación UPDATE en Section.
T – 9	Implementar la operación DELETE en Section
T – 10	Actualizar la tarea rss:feed para que categorice los enlaces recogidos en secciones

Tabla 3.3

3.1.3. HU – 3

Historia de usuario:

Como el usuario debo poder entrar al panel usando usuario y contraseña

Descripción:

Se creará el modelo de datos de usuario, se implementará un login y se protegerá el panel de configuración tras dicho login.

Tareas obtenidas:

A continuación se detallan las tareas obtenidas.

ID	Historia de usuario
T – 11	Implementar formulario de login
T – 12	Añadir un usuario por defecto al sistema
T – 13	Proteger el panel de configuración tras el login

Tabla 3.4

3.1.4. HU – 4

Historia de usuario:

Como usuario debo poder visualizar la información recogida por el sistema

Descripción:

Se creará una vista que liste la información recogida dada una sección.

Tareas obtenidas:

A continuación se detallan las tareas obtenidas.

ID	Historia de usuario
T – 14	Implementar la operación READ en RssData.

Tabla 3.5

3.1.5. HU – 5

Historia de usuario:

Como el autor del TFG debo documentar diversos aspectos y elecciones en una Wiki.

Descripción:

Se documentará todas las elecciones que no están documentadas ya al no ser parte del proceso en la wiki del proyecto.

Tareas obtenidas:

A continuación se detallan las tareas obtenidas.

ID	Historia de usuario
T – 15	Documentar el framework PHP elegido
T – 16	Documentar e SGBD elegido
T – 17	Documentar el framework javascript elegido
T – 18	Documentar la instalación hasta el momento
T – 19	Documentar el resto de tecnologías y recursos usados
T – 20	Documentar elección MALLET

Tabla 3.6

3.1.6. HU – 6

Historia de usuario:

Como el usuario debo poder configurar filtros para el extractor de noticias

Descripción:

Se creará el extractor que obtenga el cuerpo de las noticias dado un enlace a la noticia y un filtro asociado.

Tareas obtenidas:

A continuación se detallan las tareas obtenidas.

ID	Historia de usuario
T – 21	Implementar las operaciones CREATE y READ en Filter
T – 22	Mejorar el extractor para que obtenga el cuerpo de la noticia dado un filtro
T – 23	Implementar la operación UPDATE en Filter
T – 24	Implementar la operación DELETE en Filter
T – 25	Mejorar el extractor para que saque la información en el formato que MALLET necesita
T – 26	Mejorar el extractor para que limpie símbolos y palabras vacías de los documentos obtenidos antes de almacenarlos.

Tabla 3.7

3.1.7. HU – 7

Historia de usuario:

Como el usuario debo poder supervisar la tarea del extractor, dada una noticia ver el resultado de la extracción

Descripción:

Se almacenará la información antes y después de aplicar la limpieza en la fase de extracción para poder mostrarla al usuario en una vista.

Tareas obtenidas:

A continuación se detallan las tareas obtenidas.

ID	Historia de usuario
T – 27	Como el usuario debo poder supervisar la tarea del extractor, dada una noticia ver el resultado de la extracción
T – 28	Implementar la operación SHOW en RssData de forma que muestre el resultado antes y después de limpiar los datos

Tabla 3.8

3.1.8. HU – 8

Historia de usuario:

Como un usuario debo poder ejecutar los algoritmos para unos parámetros dados para así poder estudiar el comportamiento de parámetros y algoritmos

Descripción:

Crear un formulario desde el cual se pueda ejecutar el algoritmo con unos parámetros. El resultado de la ejecución aparecerá en la misma página.

Tareas obtenidas:

A continuación se detallan las tareas obtenidas.

ID	Historia de usuario
T – 29	Crear una interfaz para generar el modelo y ejecutar los algoritmos dados unos parámetros. Debe mostrar el resultado por pantalla
T – 30	Implementar CREATE de Report dada la ejecución de un algoritmo. Debe resumir los 3 ficheros de resultados

Tabla 3.9

3.1.9. HU – 9

Historia de usuario:

Como un usuario debo poder ver un informen a modo resumen de los resultados de la ejecución de los algoritmos para poder realizar el estudio con la información contextualizada.

Descripción:

Se implementarán vistas que muestren todos los aspectos del informe. Las palabras, los documentos, los temas y los valores que todos toman. Además se podrá visualizar la parametrización con la que se ha dado lugar a dicho informe como “información avanzada”.

Tareas obtenidas:

A continuación se detallan las tareas obtenidas.

ID	Historia de usuario
T – 31	Implementar la operación SHOW de Report.

Tabla 3.10

3.1.10. HU – 10

Historia de usuario:

Como un usuario debo poder ver la lista de informes generados

Descripción:

Se implementará el listado de informes, indicando información útil para distinguirlos.

Tareas obtenidas:

A continuación se detallan las tareas obtenidas.

ID	Historia de usuario
T – 32	Implementar la operación READ de Report.

Tabla 3.11

3.1.11. HU – 11

Historia de usuario:

Como un usuario no registrado debo poder ver la evolución de un tema en una línea temporal para una sección dada.

Descripción:

Se implementará la visualización de los informes y se agregará la tarea periódica que los generará.

Tareas obtenidas:

A continuación se detallan las tareas obtenidas.

ID	Historia de usuario
T – 33	Implementar el comparador y enlazador de informes
T – 34	Implementar concepto de afinidad entre dos informes
T – 35	Implementar concepto de afinidad entre dos informes
T – 36	Crear tarea que genere automáticamente informes para cada sección cada semana

Tabla 3.12

3.1.12. HU – 12

Historia de usuario:

Como un usuario no registrado debo poder ver los informes de los cuales procede dicho tema.

Descripción:

Se sacará de la parte protegida por login la visualización de las líneas temporales que muestran la evolución de un tema.

Tareas obtenidas:

A continuación se detallan las tareas obtenidas.

ID	Historia de usuario
T – 37	Añadir a cada tema de la línea temporal una referencia al informe.
T – 38	Implementar concepto de afinidad entre dos informes

Tabla 3.13

3.1.13. HU – 13

Historia de usuario:

Como autor del TFG debo completar una memoria

Descripción:

Realización del documento final. Durante el desarrollo se documentará todo aquello que se realice en la wiki, en este conjunto de tareas se adaptará la documentación de la wiki al documento y se documentará todo aquello que falte.

Tareas obtenidas:

A continuación se detallan las tareas obtenidas.

ID	Historia de usuario
T – 39	Capitulo "Introducción" de la memoria
T – 40	Capitulo "Introducción" de la memoria.
T – 41	Capitulo "Material" de la memoria. Pasar de la Wiki al documento
T – 42	Capitulo "Métodos" de la memoria. Pasar de la Wiki al documento
T – 43	Capitulo "Resultados" de la memoria. Pasar de la Wiki al documento
T – 44	Capitulo "Discusión" de la memoria
T – 45	Capitulo "Conclusiones" de la memoria
T – 46	Capitulo "Guía de instalación" de la memoria. Pasar de la Wiki al documento
T – 47	Capitulo "Requisitos Hardware" de la memoria. Pasar de la Wiki al documento
T – 48	Capitulo "Bibliografía"
T – 49	Revisar formato del documento

Tabla 3.14

3.2. Desarrollo de SCRUM

Finalmente no se ha ejecutado el proyecto siguiendo la planificación SCRUM, se han realizado las fases del proyecto secuencialmente. Esto es debido a los siguientes motivos:

- Falta de tiempo por la carga de trabajo del resto de asignaturas de la carrera
- Identificación tardía de algunos requisitos.
- Malas estimaciones del tiempo requerido para completar las distintas fases del proyecto debido a la necesidad de aprender nuevas tecnologías y comprender el modelado de temas.

Se desarrollará la planificación temporal que se ha seguido simulando SCRUM para cumplir los requisitos del proyecto.

3.3. Planificación temporal

El proyecto se inició el 26 de Septiembre de 2015 y se finalizó el 4 de Junio de 2016, sin embargo esta no era la planificación base del mismo.

En la planificación base el proyecto se divide en 4 partes bien diferenciadas, y cada parte tenía que completarse en 1 mes:

- Parte 1 – Extracción de noticias
- Parte 2 – Obtención de información (Aplicar algoritmos a las noticias)
- Parte 3 – Visualización de la información
- Parte 4 – Redacción de la memoria

La primera parte se ejecutaría en Octubre de 2015 antes de que la carga de trabajo de las asignaturas cursadas requiriera mucho tiempo. Para después retomar el resto del proyecto tras los exámenes de enero. Se estimó mal la duración de esta parte y el aprendizaje necesario para empezar a manejar el framework de desarrollo. En la planificación final esta fase se divide en dos, quedando acabada la primera parte (obtención de enlaces) y sin empezar la segunda.

La segunda parte se ejecutaría en Febrero de 2016 tras los exámenes. En Marzo la tercera parte y en Abril finalizaría el proyecto, dejando suficiente tiempo para preparar los exámenes de Mayo.

En la planificación final se dividió el proyecto de una forma más realista:

- Parte 1 – Extracción RSS
- Parte 2 – Documentación
- Parte 3 – Extracción contenido
- Parte 4 – Obtención información
- Parte 5 – Visualización
- Parte 6 – Redacción de la memoria

Se corresponderá con el siguiente diagrama de Gantt:

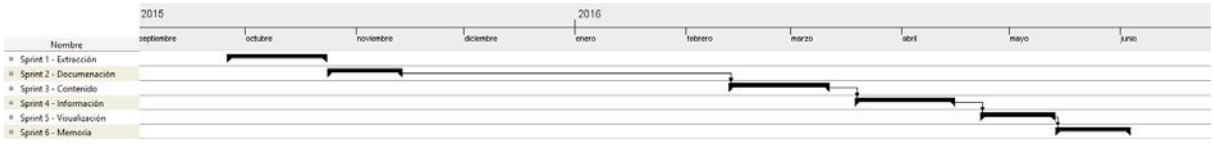

Ilustración 3.1

Primer cuatrimestre:

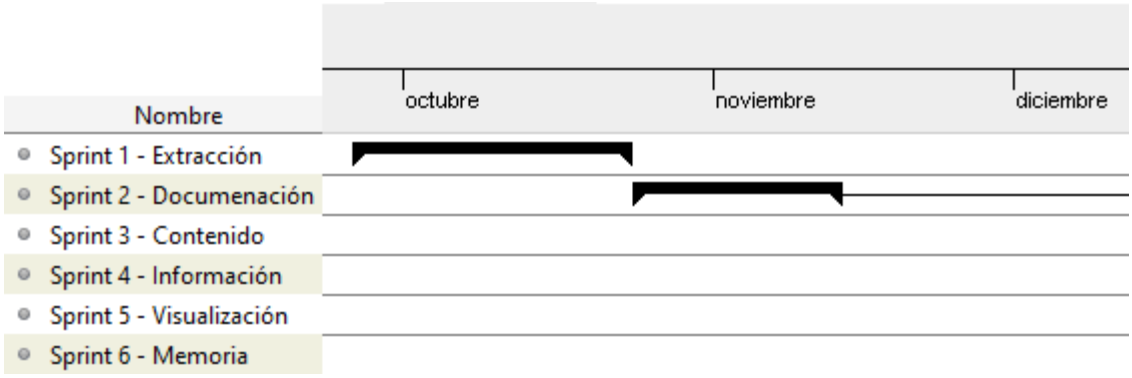

Ilustración 3.2

Segundo cuatrimestre:

Ilustración 3.3

3.4. Definición de Sprints

Una vez que tenemos definidas las tareas, tendremos que definir los Sprints. Al final de cada uno de los Sprints tendremos un entregable con una funcionalidad concreta completa.

A continuación se definirá que tareas entran en cada Sprint y en qué consistirá el entregable.

Para realizar la división tendremos en cuenta que el proyecto se está desarrollando para un Trabajo de fin de grado, el cual requiere una memoria, así que la memoria será una parte necesaria y fundamental para la conclusión del proyecto y por lo tanto se tendrá en cuenta en los Sprints.

3.4.1. Sprint 1

Este Sprint se iniciará el 26/09/15 y tendrá una duración de 4 semanas. En él se cubrirán las tareas que se corresponden a las Historias de usuario 1, 2, 3, 4.

Al final del sprint tendremos un entregable compuesto por una aplicación web en la cual se podrá configurar el extractor de datos el cual estará intentando almacenar enlaces a noticias automáticamente.

3.4.2. Sprint 2

Este Sprint se iniciará el 3/10/15 y tendrá una duración de 3 semanas. En él se cubrirán las tareas que se corresponden a la historia de usuario 5.

Al final del sprint tendremos toda la documentación del apartado material detalla en la wiki, la cual posteriormente se usará la memoria.

3.4.3. Sprint 3

Este Sprint se iniciará el 13/02/16, tras el periodo de exámenes y tendrá una duración de 4 semanas. En él se cubrirán las tareas que se corresponden a las historias de usuario 6 y 7.

Al final de este Sprint tendremos el extractor recogiendo noticias y limpiándolas de símbolos y palabras vacías por solo.

3.4.4. Sprint 4

Este Sprint se iniciará el 19/03/16 con una duración de 4 semanas. En él se realizarán las tareas que se corresponden a las historias de usuario 8, 9, 10.

Al final de este Sprint podremos ejecutar los algoritmos desde la plataforma web y estudiar sus resultados en unos informes detallados.

3.4.5. Sprint 5

Este Sprint se iniciará el 23/04/16 con una duración de 3 semanas. En él se realizarán las tareas que se corresponden a las historias de usuario 11 y 12.

Tras este sprint se podrá visualizar la evolución de un tema a lo largo del tiempo y tendremos a la plataforma generando estos informes semanalmente por si sola.

3.4.6. Sprint 6

Este Sprint se iniciará el 14/05/16 con una duración de 3 semanas. En él se realizarán las tareas que se corresponden a la historia de usuario 13.

Tras este sprint se habrá finalizado la memoria del TFG.

4. Métodos

Para poder analizar la prensa será necesario recoger las noticias, este será el proceso de obtención de datos. Una vez que tenemos los datos tendremos que sacar información a partir de los mismos, de esto se encargará el proceso de extracción de información que aplicará el modelado de temas y resumirá la información obtenida en informes. Finalmente tendremos que mostrar la información de forma coherente y útil, en el proceso de visualización.

4.1. Obtención de los enlaces

La información que se recogerá será el titular de las noticias, su sección, fecha de publicación y el cuerpo de la noticia. El sistema tendrá que recoger por si solo toda esta información según se vaya generando.

De esto se encargará el programador de tareas del sistema operativo (Cron en Unix, Tareas programadas en Windows), cada minuto pedirá al framework Laravel que ejecute las tareas programadas en el mismo. Es necesario programar esta tarea inicialmente editando el programador de tareas, pero solo habrá que editarlo una vez y crear las diversas tareas en el “Scheduler” de Laravel, De esta forma no habrá que modificar el programador de tareas del sistema operativo cada vez que haya que añadir o modificar una tarea.

Para esto añadiremos al cron de Ubuntu, con crontab –e una tarea que ejecutará php:

```
* * * * * php /vagrant/Code/artisan schedule:run >> /dev/null 2>&1
```

El siguiente paso será crear una tarea en el framework, para obtener la información lo primero que se necesita son los enlaces de cuales se extraerá el cuerpo de la noticia.

Las principales alternativas para obtener los enlaces a las noticias son la redifusión RSS o un Crawler de enlaces.

Crawler de enlaces

- Sería posible obtener todos los enlaces a las noticias disponibles dentro de un medio digital visitando su sitio web y listando los enlaces internos disponibles en el mismo mediante el análisis del documento HTML. Sin embargo esto presenta múltiples problemas:
- No siempre el orden en el cual se genera el documento HTML corresponde con la fecha de publicación de los enlaces. Los documentos HTML se construyen para crear una visualización concreta destacando un contenido concreto y el orden en el que aparece la información en los mismos no determina el orden real de publicación de dicha información. Esto dificulta el seguimiento temporal.
- No todo lo enlazado es relevante. Entre otras cosas es normal que en los medios digitales se anuncien servicios adicionales (Suscripciones de pago, redes sociales), información relevante para el usuario pero que no tiene interés para el proyecto como información meteorológica, banners publicitarios, noticias que enlazan a medios externos y noticias patrocinadas por empresas que se mezclan entre las noticias propias del medio e información sobre eventos.

Todos estos problemas complican la obtención de enlaces y la diferenciación del contenido de calidad con el contenido adicional. Sería necesario aplicar supervisión manual para filtrar correctamente los enlaces válidos.

4.1.1. Enlaces de redifusión web mediante RSS

Los enlaces RSS ofrecen un listado de enlaces a las últimas noticias publicadas ofreciendo un título, una fecha de publicación y a veces una breve descripción o extracto del cuerpo de la noticia.

Estos enlaces resuelven el problema de encontrar las noticias ordenadas temporalmente ya que siempre ofrecen las más recientes, y resuelve el problema de diferenciar las noticias que son válidas de las que no, ya que solo incluye las noticias reales.

Estos enlaces se ofrecen normalmente por categorías y a veces uno para todas las noticias del sitio web, lo cual será útil posteriormente, Así que esta modalidad será la elegida.

4.1.2. Tarea para la obtención de los enlaces

Se creará una tarea que consultará cada 5 minutos las distintas fuentes. Las fuentes a consultar tendrán que ser añadidas desde una sencilla interfaz que permitirá añadir nuevas fuentes y gestionar las ya existentes.

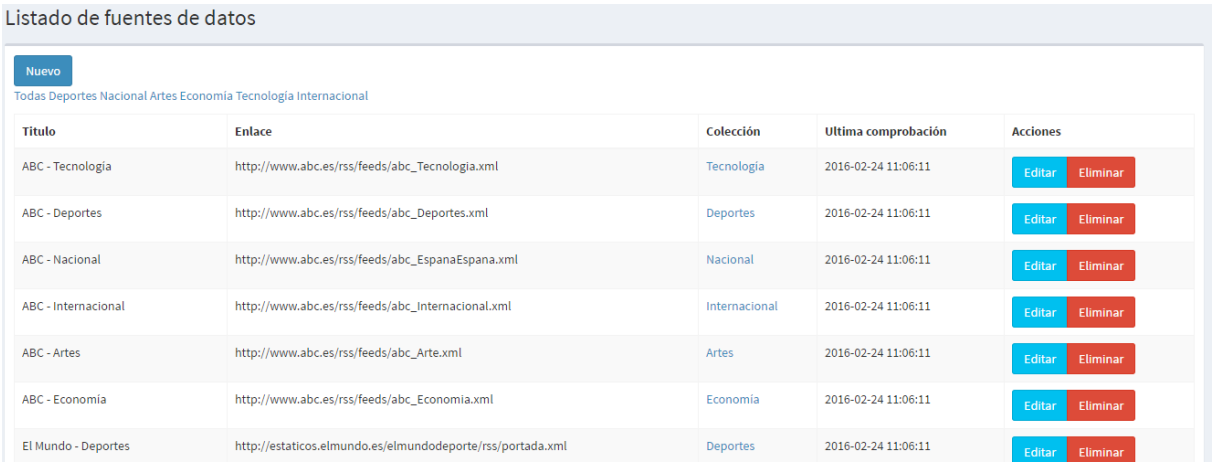

Ilustración 4.1

Ilustración 4.2

La información vital para que el sistema pueda consultar las fuentes es el enlace RSS que indica donde debe buscar la información, la colección que indica en que categoría debe colocar la información obtenida y el filtro que se usará posteriormente.

4.2. Obtención del documento

En esta parte del proceso ya tenemos las noticias identificadas mediante enlaces directos a las mismas. El siguiente paso será descargar el documento HTML de cada noticia y obtener el cuerpo de la noticia.

Para obtener el documento tenemos que hacer peticiones HTTP a los enlaces que hemos recogido anteriormente, y discriminar dentro del documento HTML el contenido que estamos buscando, en este caso el cuerpo de la noticia.

Los documentos HTML se organizan mediante un árbol DOM, como no todo el documento nos sirve, hay que añadir una forma de seleccionar el trozo del documento a recoger, en nuestro caso se usarán selectores CSS aunque también será posible usar una consulta XPATH.

Para hacer el proceso sencillo la aplicación permitirá crear estos filtros como veíamos previamente. Un filtro se compondrá de un conjunto de reglas que indican que nodo debe ser seleccionado como cuerpo de la noticia y que nodos deben ser descartados. De nuevo, se podrán crear y editar desde una sencilla interfaz CRUD en caso de ser necesario. Internet es un entorno cambiante y debe ser posible corregir el trozo de documento a extraer en caso de que algún medio cambie su sitio web.

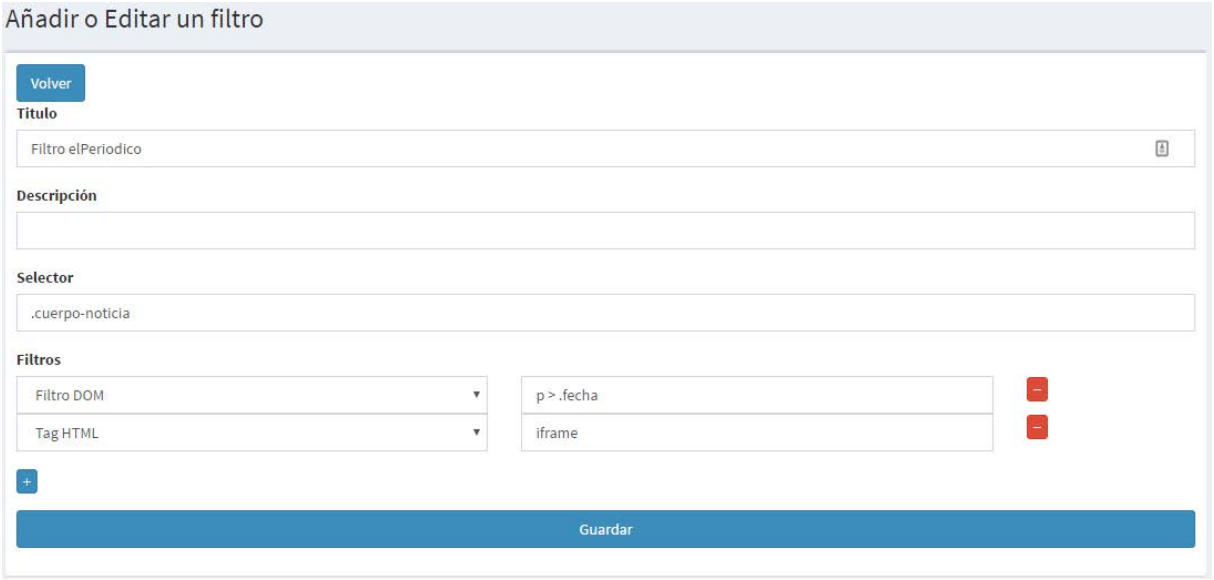

Ilustración 4.3

Usando los selectores citados es posible obtener la localización y contenido de un nodo concreto. A partir de este nodo concreto que es el que contiene la información debemos comenzar un proceso de limpieza ya que muchos medios añaden información externa a la noticia en el cuerpo de la misma como enlaces a otras noticias, así que tendremos que agregar reglas adicionales en nuestro filtro para eliminar la información sobrante.

Cuando seleccionamos el nodo que contiene el cuerpo de la noticia, tenemos un nodo con un conjunto de nodos hijos, incluiremos reglas de exclusión que se encargarán de eliminar los nodos que no queramos y finalmente se extraerá el texto del nodo inicial.

A continuación se explica de una forma más gráfica el proceso de extracción del cuerpo de la noticia mediante un ejemplo. Primero nos dirigimos al enlace:

http://www.elmundo.es/tecnologia/2016/03/21/56efc67be2704eb0678b45bf.html

Se visualizará en el navegador así:

Ilustración 4.4

Usando las herramientas de desarrolladores del navegador tendremos que buscar la caja mínima que contenga todo el texto de la noticia. Seleccionaremos la caja mediante el selector `[itemprop="articlebody"]`, resaltado en azul en la siguiente imagen:

Ilustración 4.5

El problema es que dentro de esta caja hay información no relevante, como la imagen y el pie de imagen, los datos del autor, la publicidad que ha quedado dentro de la caja y los enlaces relacionados de la izquierda. Para ello debemos añadir los selectores de estos elementos en el filtro.

Aquí se muestra en el cuadrado azul la zona seleccionada para extraer y en rojo las zonas seleccionadas para eliminar:

Ilustración 4.6

Tras eliminar los nodos quedaría algo similar a la siguiente figura:

Ilustración 4.7

Tal como se muestra en la imagen, la publicidad o texto que queda fuera de la caja no es necesario eliminarla pues simplemente se ignorará a la hora de obtener el texto. Finalmente solo tendremos que almacenar todo el texto que este dentro de la caja azul. Esta es una representación gráfica de cómo se soluciona el problema, realmente tenemos un árbol DOM en el cual buscamos el nodo que representa a la caja azul:

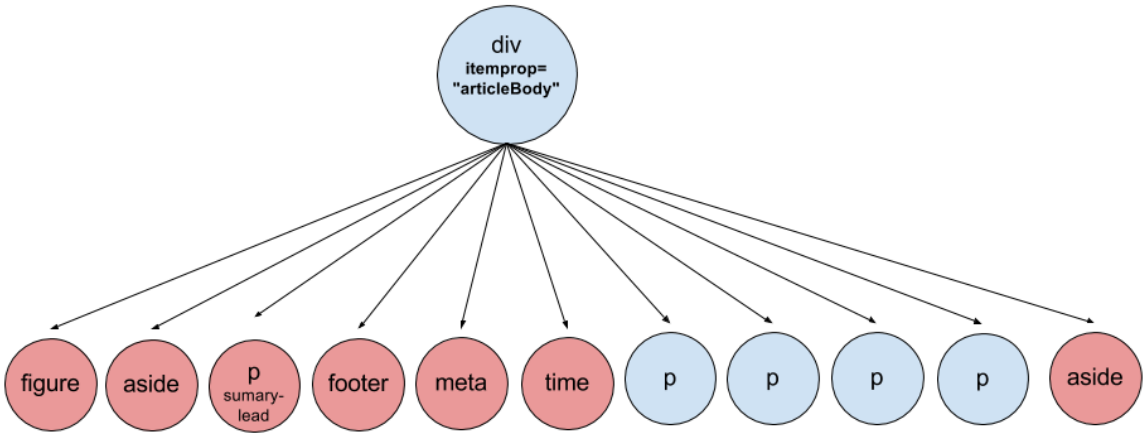

Ilustración 4.8

Los nodos marcados en rojo serán los nodos que eliminaremos, mientras que los azules los dejaremos. Tras la limpieza tendremos algo así:

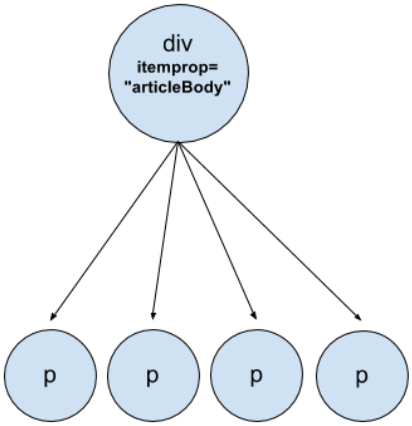

Ilustración 4.9

Este es el momento en el que tenemos que almacenar todo el texto contenido en este árbol. En este ejemplo el texto está dentro de los párrafos.

El ejemplo elegido muestra uno de los casos más adversos para llegar a la información ya que el sistema debe estar preparado para funcionar en cualquier caso, aunque normalmente es más sencillo.

4.3. Limpieza del texto

Una vez que tenemos el cuerpo de la noticia, será necesario aplicar pre-procesamiento antes de almacenarlo para que los algoritmos lo procesen.

Para realizar la limpieza de un texto normalmente se aplica:

- **Tokenización,** se eliminan los signos de puntuación. Normalmente se eliminan los acentos de las palabras para identificar la palabra bien escrita y la palabra mal escrita como la misma palabra pero en este caso no se realizará pues las noticias se recogerán de medios de calidad y se supondrá que las palabras siempre aparecerán bien escritas.
- Conversión de todos los caracteres a **minúsculas**.
- Las palabras se reducen a su **forma canónica**.
- Se eliminan las **palabras vacías** (StopWords).

4.3.1. Tokenización

Eliminaremos los signos de puntuación y caracteres extraños de los textos mediante una eliminación manual. Sería posible aplicar una expresión regular que eliminase cualquier letra que no esté contenida entre la a y la z pero esto crea numerosos problemas, así que se realizará un listado de tokens a eliminar manualmente, pudiendo seleccionar que eliminar y que no, esto resuelve los problemas de la expresión regular citada:

- Elimina las palabras acentuadas al no reconocerlas como integrantes del rango a-z, que en nuestro caso se mantendrán ya que suponemos que los medios son especializados y no cometen errores de este tipo.
- Enlaces web pierden significado, convirtiendo "google.com" en "googlecom". La solución será reemplazar primero eliminar “.com” y después eliminar el punto, quedando “google”. Si se mantiene el orden a la hora de eliminar los tokens, los enlaces nunca perderán significado. Esto se aplica con las principales extensiones de dominio.
- Se perderían caracteres de otras lenguas como el Chino, kanji Japonés, Coreano, o Cirílico. En noticias internacionales puede tener sentido escribir

un término concreto (o un nombre de persona, un producto) en su caracterización original. Eligiendo manualmente los caracteres a eliminar, estos caracteres e mantendrían.

Un ejemplo donde se usa la tokenización tal como se ha explicado previamente es para aprovechar información de un enlace ya que a veces los medios incluyen enlaces en el propio texto (En este ejemplo el texto del enlace es el mismo que la dirección URI del elemento enlazado):

https://drive.google.com/?hl=es

Si limpiamos en orden, se eliminara el "com" y se eliminarían el resto de tokens “/” , “?” y “=” en este caso). Quedando "https", "drive", "google", "hl", "es". Esto nos permite identificar a "google" sin confundirlo con "googlecom" y relacionar a "google" con "drive" ya que en la noticia probablemente se cite al servicio de almacenamiento Google Drive.

Para crear la lista de tokens a eliminar y estudiar el orden adecuado para la eliminación, es posible comprobar el resultado de la tokenización para una noticia concreta. Se mostrará la noticia tokenizada, el estado anterior de la noticia y un listado con los tokens que no se han eliminado (tokens permitidos como letras acentuadas)

Ilustración 4.10

Para eliminar los tokens, se reemplazan por un espacio así que finalmente se usará una expresión regular que buscará múltiples espacios seguidos y los reducirá a un solo espacio, que es el formato requerido para generar un modelo a partir de la información en pasos posteriores.

4.3.2. Pasar a minúsculas

No se realizará este paso en este momento ya que cuando se pase el texto a el algoritmo, este lo realizará como parte del pre-procesamiento y estaríamos haciendo lo mismo dos veces. Se supondrá realizado este paso.

4.3.3. Canonización

No se realizará para evitar la pérdida de significado de las palabras concretas en los documentos. Por ejemplo “Fernando Alonso quedó segundo por solo dos segundos”, aplicar la canonización provocaría una gran pérdida de información en “segundo” y “segundos”.

4.3.4. Eliminar las palabras vacías

Se incluirá una lista con las palabras que no aporten significado en el proceso de descarga y limpieza del documento y se eliminarán estas palabras tras realizar la tokenización.

Para obtener el listado de palabras vacías se han combinado dos listas públicas, una de ellas recomendada por los creadores de MALLET, sacada de ranks.nl y la otra para completar a la primera obtenida en elwebmaster.com. Adicionalmente se han añadido adverbios, preposiciones, determinantes y pronombres para completar las listas.

Eliminar las palabras vacías hará que el algoritmo no relacione estas palabras creando temas irrelevantes.

Tras ejecutar los algoritmos se detectan nuevas palabras vacías, así que también se eliminarán números ordinales para ya que normalmente actúan como palabras vacías. Si creamos un modelo antes y después de eliminar los números ordinales sobre la categoría deportes, y ejecutamos el algoritmo parametrizado igual, podremos observar diferencias:

Antes de eliminar los números obtendremos 2 de 5 temas que los contendrán, eliminamos el resto por no ser de interés.

```
3       10      cuarto canasta falta madrid personal real barcelona falla rebote
triple libre tiro fc lassa defensivo sergio asistencia encesta segundo

4       10      set nadal final gran rafael golpe punto derecha segundo primera
primer pista español victoria tres segunda partido juego resto
```

Después buscamos los temas equivalentes tras aplicar el modelo eliminando números:

```
3       10      canasta falta madrid real personal barcelona falla rebote triple
libre tiro fc lassa defensivo sergio asistencia encesta balón carlos

4       10      set nadal gran carrera final rafael golpe derecha punto año pista
español victoria campeón mejor resto título juego mundial
```

En el primero se eliminan dos palabras (cuarto y segundo) y permiten que otras avancen (cuanto más a la izquierda en el tema, más puntuarán para procesos posteriores) y que entren otras. En el segundo tema entran palabras significativas como campeón, carrera y mundial.

Como podemos observar, la limpieza de palabras vacías es una de las partes más importante del proceso, si se ejecuta pobremente los resultados serán de una calidad muy inferior.

La eliminación de palabras vacías no es una parte estática del proyecto sino que habrá que actualizar la lista según observemos nuevas palabras vacías para que los temas resultantes no las contengan y puedan incluir un mayor significado.

4.3.5. Almacenar el resultado

Se almacenará el resultado de dos formas, por un lado se almacenará el título y enlace en una colección de MongoDB. Habrá una colección de Mongo para cada sección de noticias (Nacional, Internacional, Tecnología, Deportes, etc.) y se podrán crear secciones adicionales desde una interfaz CRUD sencilla.

Esta división inicial de las noticias por categorías es muy importante ya que ayudará al algoritmo a obtener mejores resultados ya que se aplicará sobre cada una de las secciones de forma independiente, esto evita que noticias entre diferentes secciones se mezclen. Es equivalente a una fase de clasificación inicial en minería de datos.

Finalmente se almacenará el cuerpo de la noticia tras la limpieza en un fichero en disco, los ficheros se organizarán en carpetas en base a su sección y el nombre de los ficheros será el identificador de Mongo (ObjectID) del titular y enlace que se han almacenado previamente, así se podrá recuperar el titulo e información de un documento dado.

4.4. Creación del modelo

En este punto tendremos una carpeta para cada sección con un conjunto de documentos, estos documentos estarán preparados para ejecutar el algoritmo.

El algoritmo lo ejecutará el software MALLET y para eso hay que darle a conocer los documentos que debe procesar para que este pueda crear un modelo sobre el cual aplicará el algoritmo realmente.

Para eso ejecutaremos un comando, este comando se ejecutará semanalmente para cada sección mediante una tarea que agregaremos al framework.

```
mallet/bin/mallet import-dir
--input [ RUTA A LOS DOCUMENTOS ]
--output [ CARPETA SALIDA]/LDA[ SECCION ].model
--token-regex "\S+"
--remove-stopwords true
–-encoding utf-8
--keep-sequence
```

La primera línea es la ruta hacia el ejecutable y el parámetro import-dir que indica a MALLET que debe crear un modelo a partir del directorio dado, como entrada le daremos la ruta hacia una de las secciones y como salida la carpeta que queramos y nombraremos adecuadamente al fichero de salida.

Con token-regex “\S+” le indicamos que las palabras vienen separadas por espacios, aquí podemos verlo de una forma más gráfica:

Ilustración 4.11

Con encoding indicamos que las palabras pueden contener caracteres UTF-8, esto evitará que las elimine y permitirá que presente un resultado en UTF-8 (si una palabra acentuada o con ñ aparece en un tema, se mantendrá legible).

Con keep-secuence indicamos que el orden de las palabras en los documentos es importante y debe incorporarse al modelo. Una cuenta de palabras simple no es suficiente para aplicar los algoritmos a continuación.

4.5. Ejecución de los algoritmos

El modelado de temas es una forma concreta de hacer Minería de Textos, mediante un conjunto de algoritmos que son capaces de detectar y extraer relaciones semánticas latentes en un corpus de documentos se pueden identificar patrones, palabras que co-ocurren. Se asume que las palabras que suelen aparecer en los mismos contextos hacen referencia a los mismos temas.

Un símil sería coger un artículo periodístico, identificar los términos más importantes del artículo y marcar con un color diferente estos términos en base a su temática. Al finalizar, cada uno de los conjuntos representaría un tema.

Un tema entonces sería un conjunto de palabras que tienden a aparecer en los mismos contextos a lo largo del corpus de documentos dado y si la probabilidad de co-ocurrencia de dos palabras es muy marcada, se asume que hacen referencia a un tema común.

Los algoritmos estudiados para extraer los temas del corpus de documentos son:

- LDA - Latent Dirichlet Allocation
- Topical-N-Grams
- 4 Level PAM - Pachinko Allocation Model

4.5.1. LDA

Su pseudo-código:

```
PARA CADA iteración it
  PARA CADA documento
    PARA CADA palabra token

      actualizarTablaTopicPalabra(topic, token, -1 );
      actualizarTablaTopicDocumento(topic, documento, -1 );

      PARA CADA topic t
        peso1 = getPeso(documento,topic) + alpha;
        peso2 = getPeso(token,topic) + beta;
        peso[t] = += (peso1 * peso2);

      topicElegido = Distribución_multinomial(peso);

      actualizarTablaTopicPalabra(topicElegido, token, +1 );
      actualizarTablaTopicDocumento(topicElegido, documento, +1 );
```

El algoritmo recorre cada token de cada documento un número de iteraciones determinado, manteniendo dos tablas en memoria. Una tabla almacenará la afinidad tema-palabra y otra la afinidad tema-documento.

Cada ronda se puntuará la afinidad de la palabra con todos los posibles temas, usando los valores de ambas tablas, aunque en ese momento el valor de alguna tabla sea cero, se le dará un cierto valor para no descartar la posibilidad de que una palabra entre en un tema.

Una vez que tenemos la afinidad entre la palabra y los distintos temas hay que decidir cuál elegir, se decidirá siguiendo una distribución probabilística multinomial con una componente aleatoria. Cuanto más afinidad más probabilidades de que la

palabra se quede en ese tema, pero la componente aleatoria hace que cualquier tema pueda salir.

Las tablas se actualizan justo antes del cálculo, reduciendo la afinidad tema-documento y la afinidad tema-palabra y después del cálculo incrementando la afinidad pero con el tema elegido. Se podría pensar que la palabra se des categoriza durante un momento, modificando las tablas de afinidad y posteriormente se vuelve a categorizar y las tablas se actualizan.

Un ejemplo:

Para cada documento se asigna cada una de las palabras a un tema.

Palabra	Topic
Palabra 1	1
Palabra 2	2
Palabra 3	3
Palabra 4	2
Palabra 5	3

Tabla 4.1

Sumaremos la cantidad de ocurrencias de las palabras en el corpus y se anota en una tabla general.

Mantendremos una tabla global de dimensiones N (Numero de temas) x M (Número de palabras), en esta tabla se sumara el número de ocurrencias de una palabra para un Topic concreto a lo largo del corpus de documentos.

Palabra	Topic 1	Topic 2	Topic 3
Palabra 1	5	5	2
Palabra 2	23	3	4
Palabra 3	3	7	22
Palabra 4	2	4	11
Palabra 5	3	8	2
Palabra 6	3	6	2
Palabra 7	8	8	1

Tabla 4.2

A continuación se analizará cada palabra de nuevo en su documento, para ello se eliminara el tema asignado a dicha palabra en dicho documento

Palabra	Topic
Palabra 1	?
Palabra 2	2
Palabra 3	3
Palabra 4	2
Palabra 5	3

Tabla 4.3

Esto provocará que tengamos que actualizar la tabla global, restando 1 a la ocurrencia de dicha palabra para dicho tema.

Palabra	Topic 1	Topic 2	Topic 3
Palabra 1	4	5	2
Palabra 2	23	3	4
Palabra 3	3	7	22
Palabra 4	2	4	11
Palabra 5	3	8	2
Palabra 6	3	6	2
Palabra 7	8	8	1

Tabla 4.4

Ahora volvemos a decidir a que Topic va la palabra en base a la información actualizada. Se elegirá de forma aleatoria ponderada basándose en una distribución multinomial.

La probabilidad de que la palabra se calculará como el producto del peso de la palabra p para el tema (el valor de la tabla global) por el peso del documento para el tema.

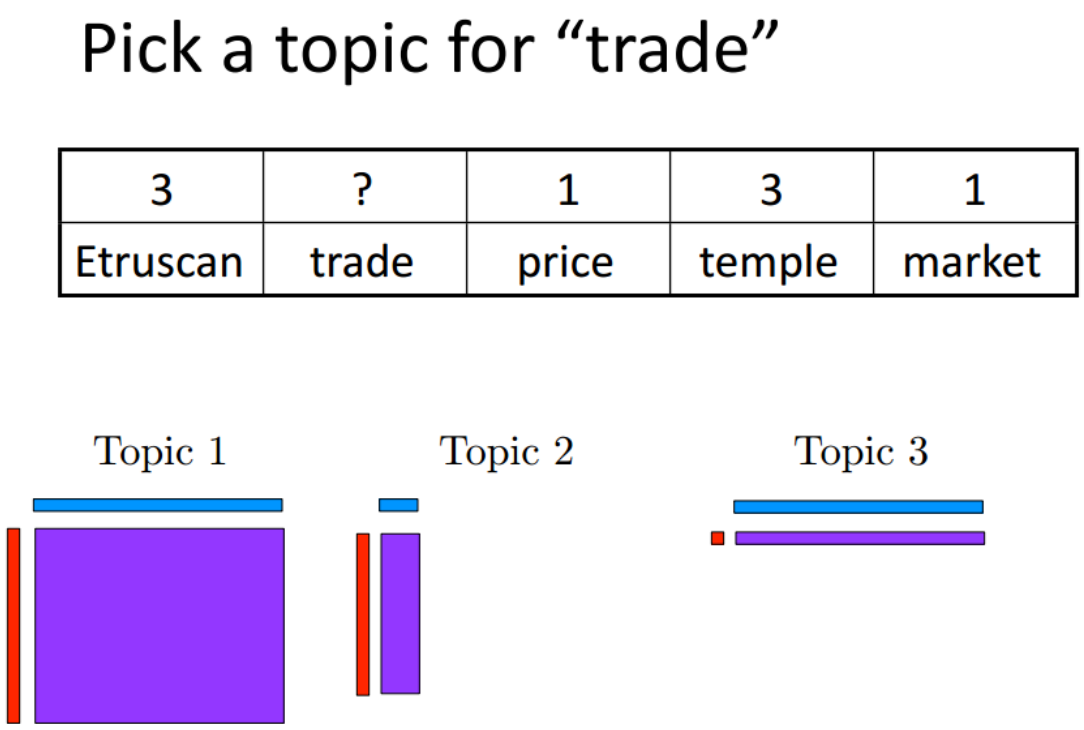

Ilustración 4.12 (Mimno, Topic Modeling Workshop: Mimno, 2012)

Con los pesos relativos obtenidos y usando una función de probabilidad multinomial se elegirá el topic al que pertenecerá la palabra. El 1 tendrá más probabilidad que el 2 y este tendrá más probabilidad que el 3 en el ejemplo dado, pero todos están dentro de las posibilidades.

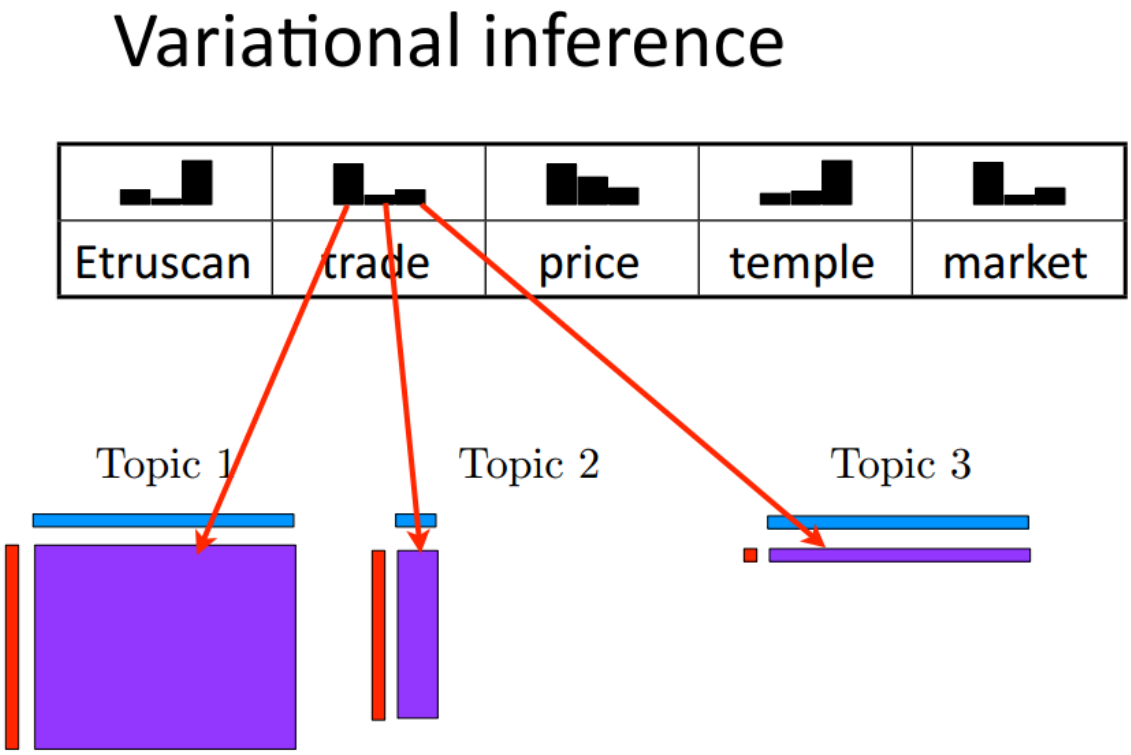

Ilustración 4.13 (Mimno, Topic Modeling Workshop: Mimno, 2012)

Tras elegir se actualizan las tablas y se continuará con el resto de las palabras de todos los documentos a lo largo de las iteraciones definidas.

Finalmente obtendremos una lista para cada tema con todas las posibles palabras puntuadas, el algoritmo simplemente que queda con las de mayor puntuación. La cantidad de palabras que toma es parametrizable.

4.5.2. Topical N-Grams

Un N-grama es una subsecuencia de n elementos de una secuencia dada. Un modelo de n-grama es un modelo probabilístico que permite predecir el próximo elemento de una secuencia de elementos. Se suele asumir que cada elemento depende de l-os últimos n elementos de la secuencia.

Este algoritmo es similar a LDA pero con descubrimiento de frases (n-gramas) integrado. A la hora de generar el modelo hay que indicar el tamaño máximo de los n-gramas, por ejemplo para bigramas añadimos --keep-sequence-bigrams en lugar de --keep-sequence. Para N-gramas mayores usaremos --gram-sizes.

Este método es bueno para encontrar frases que se repiten a lo largo del corpus de documentos y aplicar LDA a estas subfrases, pero el tamaño de las subfrases es fijo. Si creamos un tema con 20 términos y usando bi-gramas, obtendremos 20 bigramas formando el tema, lo cual equivale a un conjunto de 40 palabras unidas de dos en dos. Algunos de los bigramas tendrá sentido que estén unidos y otros de los bigramas no lo tendrán, el hecho de forzar a que todos los términos del tema estén formados por bigramas no ayuda a la formación de un tema de calidad.

Si aplicamos el algoritmo a la sección Deportes, uno de los términos que siempre aparece será “real_madrid”, obtenido como la unión de “Real” y “Madrid”, sin embargo aparece junto a otros 19 términos frutos de una unión también, y de los cuales no todos forman un concepto diferenciado como el de este ejemplo.

Por si solo este algoritmo no es útil pero puede ayudarnos en la identificación de términos que siempre van juntos y representan un concepto completo (un nombre propio como Ismael Garrido representa un concepto completo y podríamos representarlo en LDA como Ismael_garrido y esto permitiría aprovechar más el espacio de términos (Ismael_garrido ocuparía un solo termino) y aprovechar la ejecución del algoritmo que ya no se preocuparía de la relación entre Ismael y Garrido sino que podría ponderar otras relaciones menos evidentes.

Pasa de forma similar con “Madrid” y “Real Madrid” cuando aplicamos LDA a la sección de deportes, es posible que el término que representa a la ciudad se mezcle con el término que representa al equipo. En este caso si nos serviría aplicar un pre-procesamiento a LDA donde identifiquemos N-Gramas comunes y los reemplacemos en el corpus por la conexión de ambas palabras.

La dificultad que encontramos es distinguir cuales de estas conexiones son de calidad y cuáles no, sería necesario aplicar test estadísticos y supervisión manual para obtener buenos resultados.

4.5.3. Pachinko Allocation Model (4-Level PAM)

Pachinko Allocation Model de 4 niveles consiste en una raíz (nivel inicial), cuyos nodos hijos son un conjunto de súper temas (nivel 2), cuyos nodos hijos son un conjunto de temas (nivel 3), cuyos hijos son las palabras finales (nivel 4, nodos hoja).

Se ejecuta una vez LDA sobre el nivel 2 e individualmente se ejecuta LDA sobre el nivel 3. Esto permite la creación de “súper temas” que influirán en los temas de nivel 3.

4.6. Tratamiento de resultados

Una vez ejecutado un algoritmo se obtienen un conjunto de ficheros que se almacenarán en disco, estos ficheros tendrán que analizarse y resumirse para poder trabajar con la información que nos ofrecen posteriormente.

Los ficheros se leen y se organiza la información entorno a los temas para un acceso veloz. En los ficheros se muestran todas las combinaciones posibles de tema-palabra y tema-documento, así que habrá líneas que nos indican que la palabra “españa” no ha aportado nada al tema 3, así que se establecerá un mínimo de aportación al tema para recoger esa información, esto reduce considerablemente el tamaño de la información que se guarda finalmente.

Una vez que tenemos la información que deseamos guardar, se ajusta a un modelo de datos (a un Informe) y se almacena en la base de datos.

4.7. Visualización de informes

Una vez que tenemos un informe, podemos visualizarlo de una forma sencilla

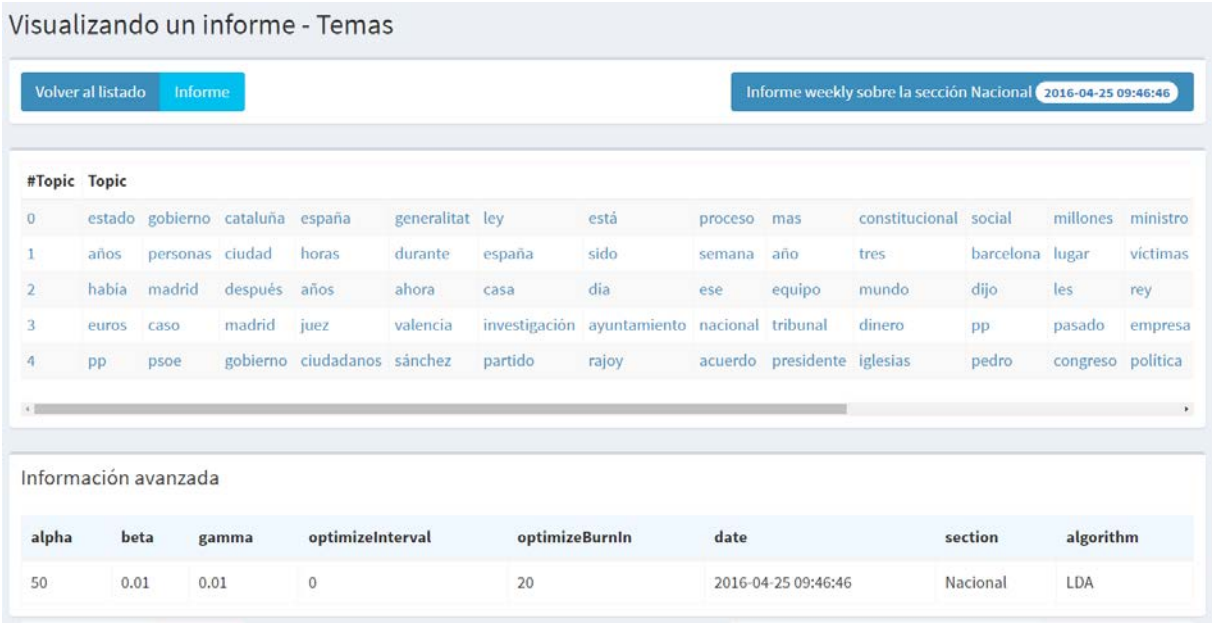

Ilustración 4.14

En la visualización del informe se puede visualizar en detalle un tema, viendo las palabras que quedaron fuera del tema y la puntuación de las mismas y permitiendo filtrar.

Adicionalmente se puede ver la aportación de una palabra a los distintos temas y navegar a través de palabras y temas. Desde aquí podemos estudiar los temas de una forma más sencilla para descubrir cuáles son los mejores parámetros,

4.8. Creación de informes

En el sistema existirán 2 tipos de informes los semanales que se generaran automáticamente cada semana, llamados “weekly” y los que se generan desde un sencillo panel manualmente etiquetados como “manual”. Además cada informe estará ligado a una sección.

Desde el backend podremos listar los informes disponibles:

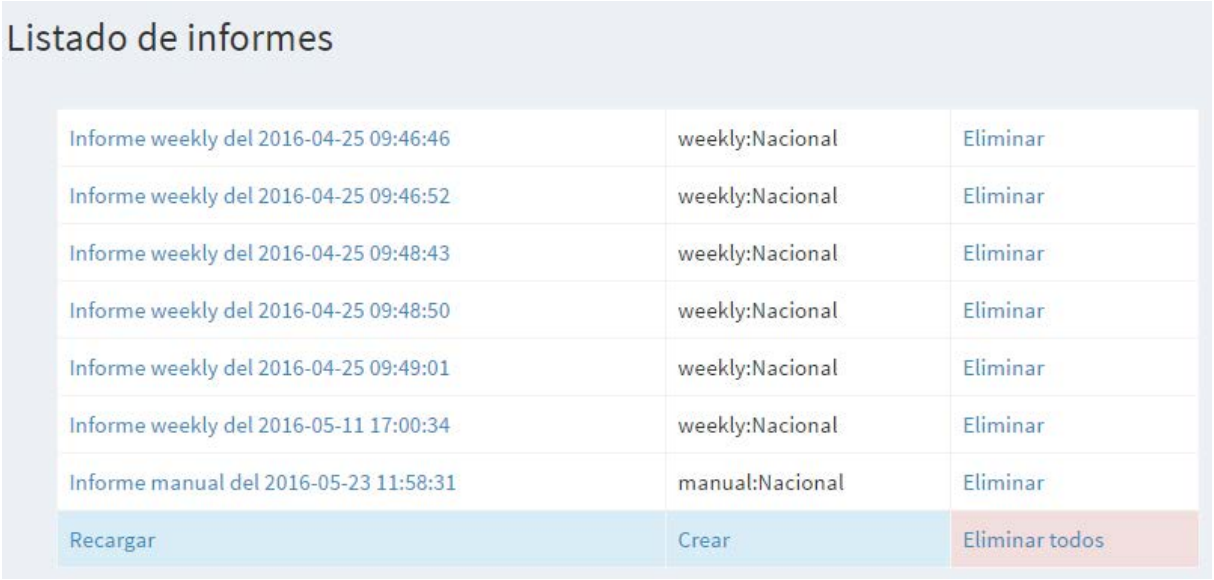

Ilustración 4.15

4.8.1. Informes manuales

Estos informes permitirán estudiar los parámetros para cada una de las secciones hasta encontrar los más adecuados. Permitirán introducir los parámetros más importantes y presentarán el resultado de la ejecución al finalizar. También se generará un informe para dicha ejecución.

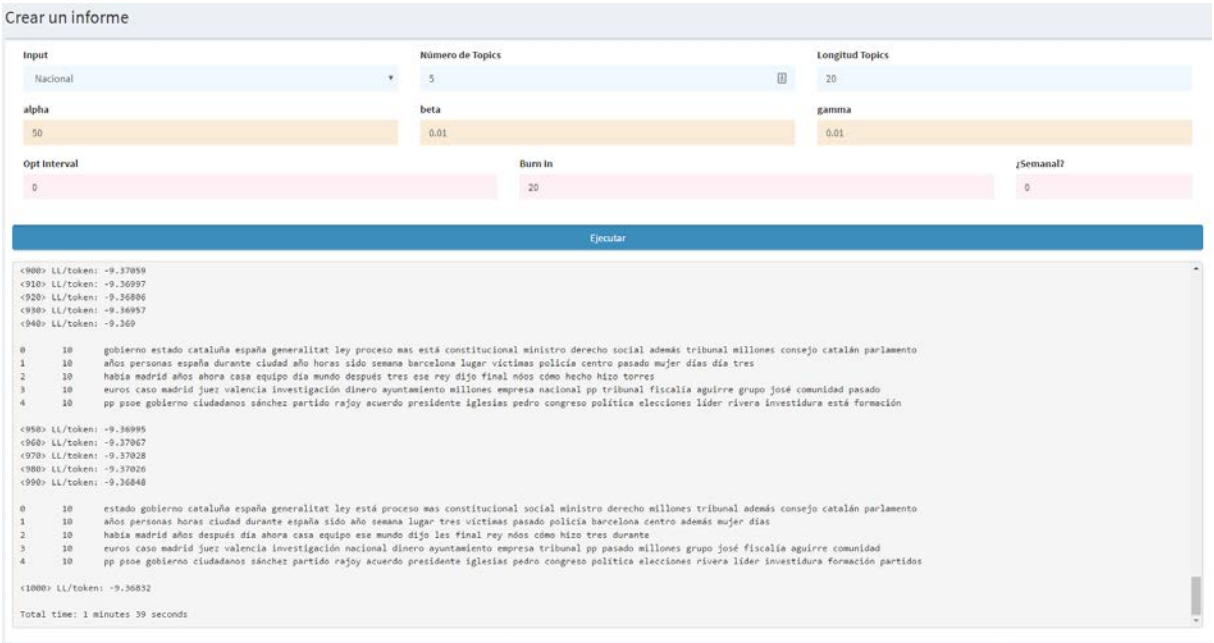

Ilustración 4.16

4.8.2. Informes semanales

Para poder estudiar la evolución de los temas a lo largo del tiempo, tendremos que generar automáticamente un informe semanal para cada una de las secciones, así solo tendremos que comparar todos los informes marcados como semanales “weekly” para cada una de las secciones por separado.

Para que esto sea posible se creará una tarea en el framework que se ejecutará los domingos a las 00:00 automáticamente. Esta tarea leerá los documentos que encuentre en unas determinadas carpetas (las de cada sección) y creará un modelo para cada sección. Posteriormente aplicará el algoritmo con los mejores valores para cada una de las secciones, generando así un informe por cada sección.

4.9. Visualización de la evolución de un tema

Para visualizar cómo evoluciona un tema a lo largo del tiempo, se comparan informes semanales ordenándolos temporalmente. Como se genera uno por semana, solo hay que comparar el de la semana 1 con el de la semana 2, el de la semana 2 con el de la semana 3 y así hasta el último informe.

Representaríamos gráficamente un informe como:

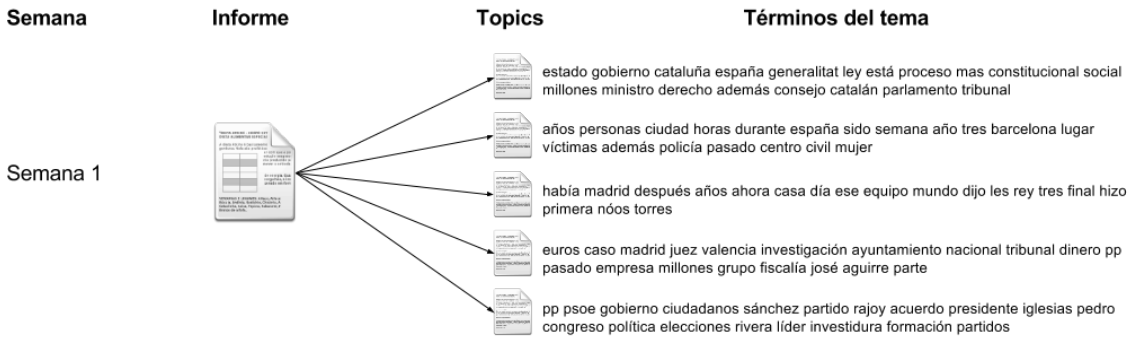

Ilustración 4.17

Para ver la evolución de los temas a lo largo de 2 semanas consecutivas, tendremos que enlazar los temas de la primera semana con los de la segunda. Los temas no aparecen en un orden concreto así que se usará un mecanismo concreto para buscar las parejas de temas más similares.

En la siguiente imagen podemos ver un ejemplo de cómo se re enlazan los temas a lo largo de 3 semanas. Para cada tema se ha usado un color distinto, siguiendo la línea de ese color podríamos ver la evolución del tema gráficamente. Los cambios de la semana 1 a 2 se muestran con un enlace continuo mientras que los de la semana 2 a 3 se muestran con un enlace discontinuo.

Cada enlace se ponderará con una puntuación a la cual llamaremos afinidad. A mayor afinidad, más cercano es un tema respecto al que le sigue en la semana posterior.

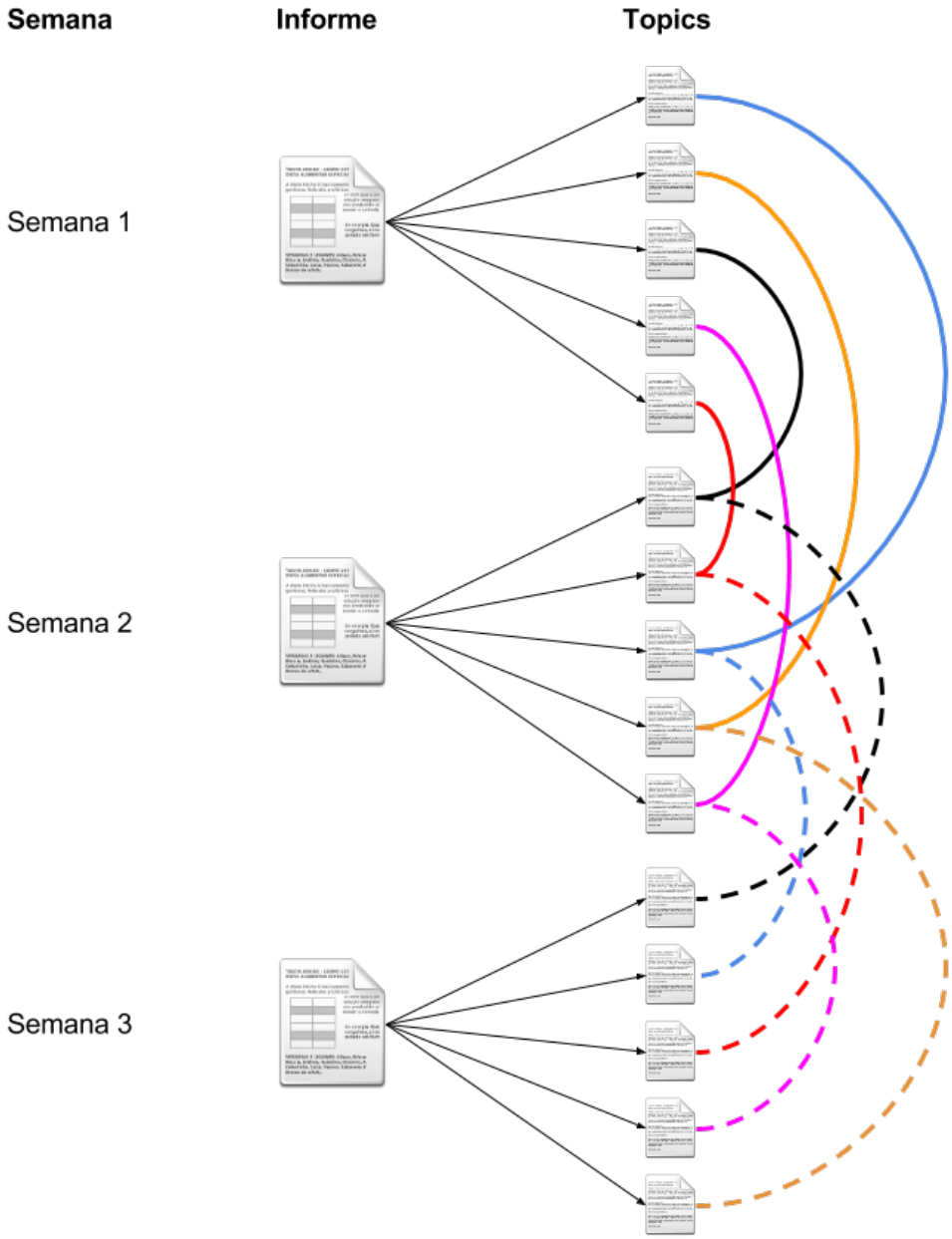

Ilustración 4.18

Para medir la afinidad entre 2 temas, se colocarán las palabras de ambos temas en un conjunto y se aplicará la operación intersección, manteniendo solo las palabras comunes entre ambos conjuntos.

No hay que olvidar que el orden de las palabras en un tema es equivalente a la importancia de dichas palabras en el tema ya que el algoritmo construye temas sin un tamaño límite y cuando finaliza corta con los N primeros que le pedimos. (Normalmente pedimos temas de 20 términos, así que se queda con las 20 palabras con mayor puntuación dentro del tema). Usaremos esta propiedad para puntuar la afinidad entre ambos conjuntos. Para simplificar la explicación tomaremos un tema existente y simularemos que solo tiene 5 palabras.

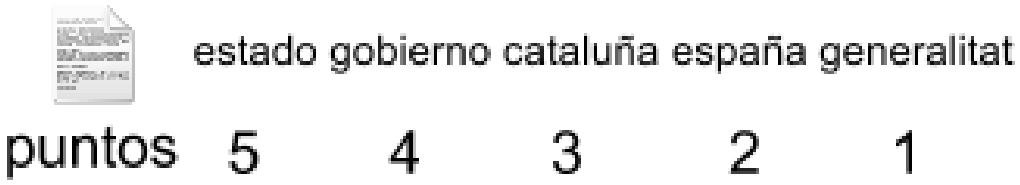

Ilustración 4.19

Asignaremos puntos a las palabras siguiendo algún tipo de lógica, en este caso se puntuarán las palabras en base a su posición. Si tenemos 5 palabras, la primera puntúa 5 puntos, la segunda 4 puntos y así hasta que la quinta puntué 1 punto.

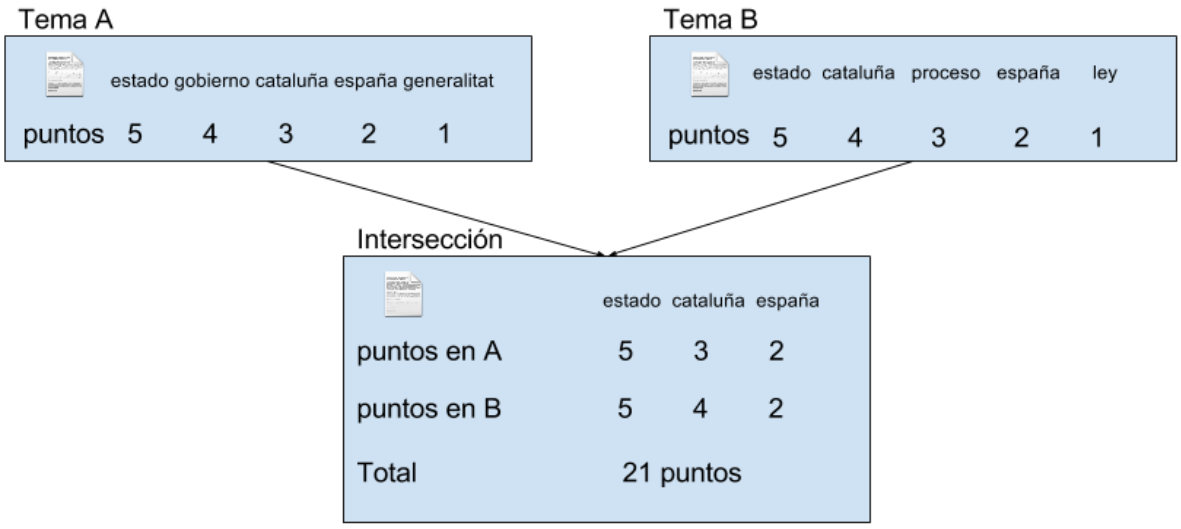

Ilustración 4.20

En el ejemplo dado se suman 21 puntos sobre un total de (5 + 4 + 3 + 2 +1) * 2 = 30 puntos. Así que su afinidad será del (21/30) = 70%.

Si la afinidad calculada baja del 50% diremos que el tema probablemente cambiado y si está por debajo del 25% diremos que ha cambiado seguro ya que es lo suficientemente distinto.

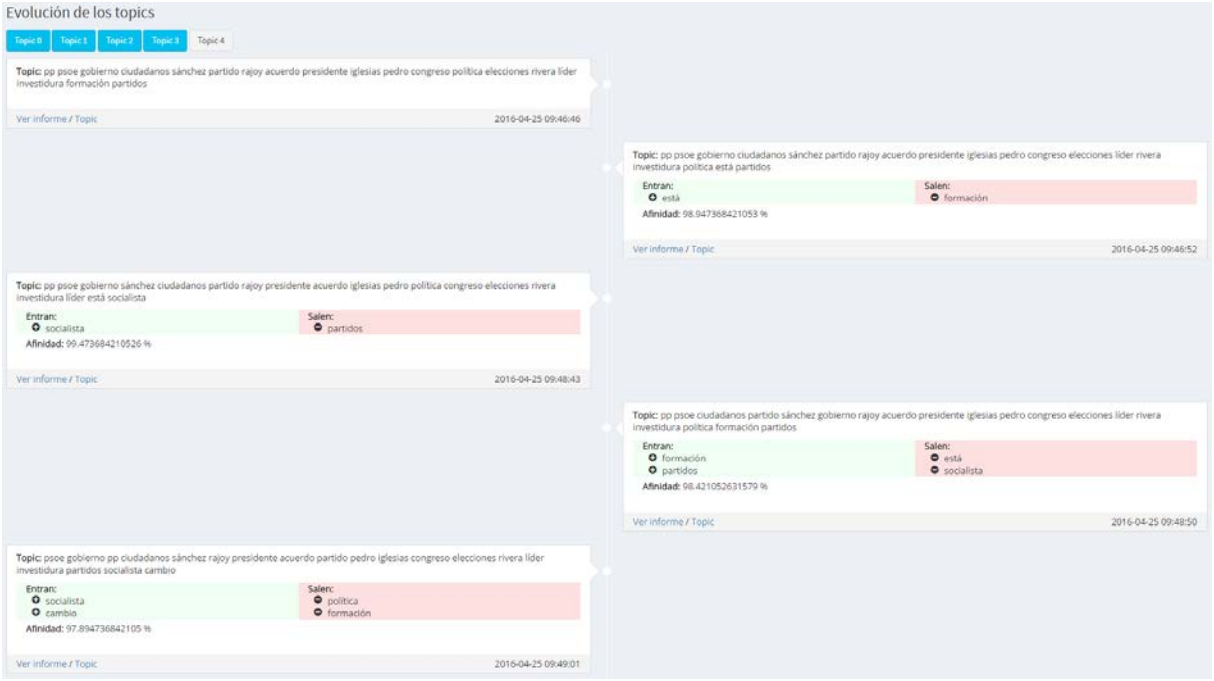

Ilustración 4.21

Los resultados se presentarán en una línea temporal vertical indicando semana a semana que términos han entrado y salido de un tema, cual es la afinidad entre ambas semanas y si el tema sigue siendo el mismo o podemos afirmar que ha desaparecido.

Un ejemplo en el que la afinidad se encuentra entre el 25% y el 50%

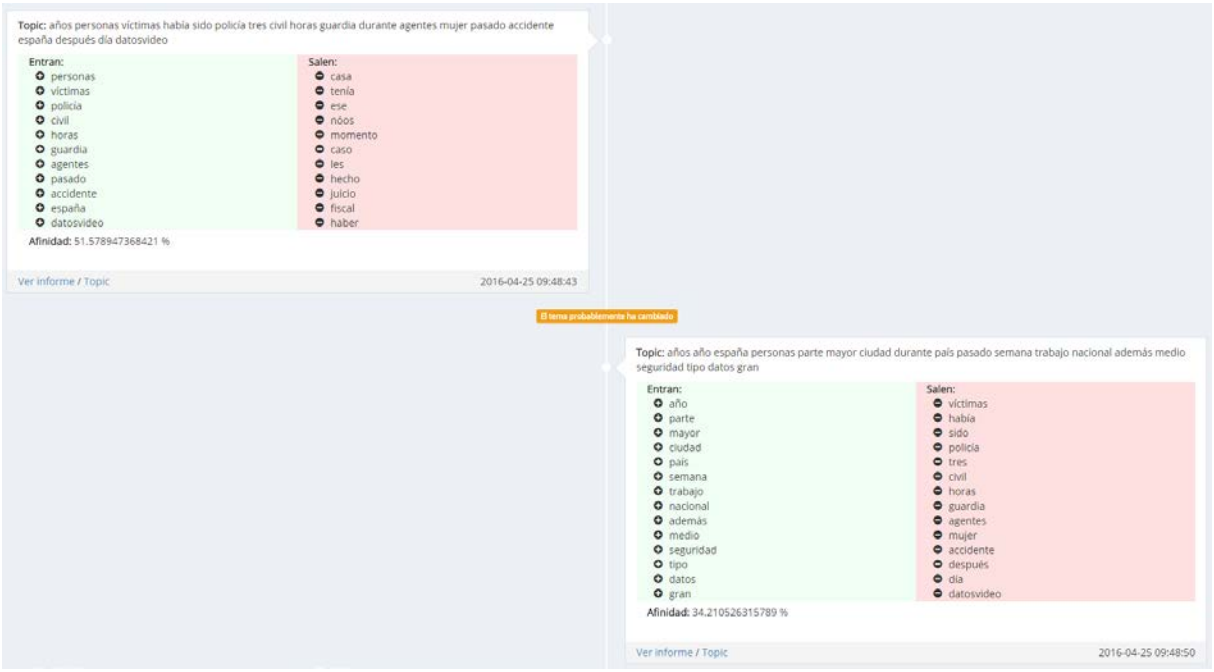

Ilustración 4.22

Un ejemplo con la afinidad por debajo del 25%

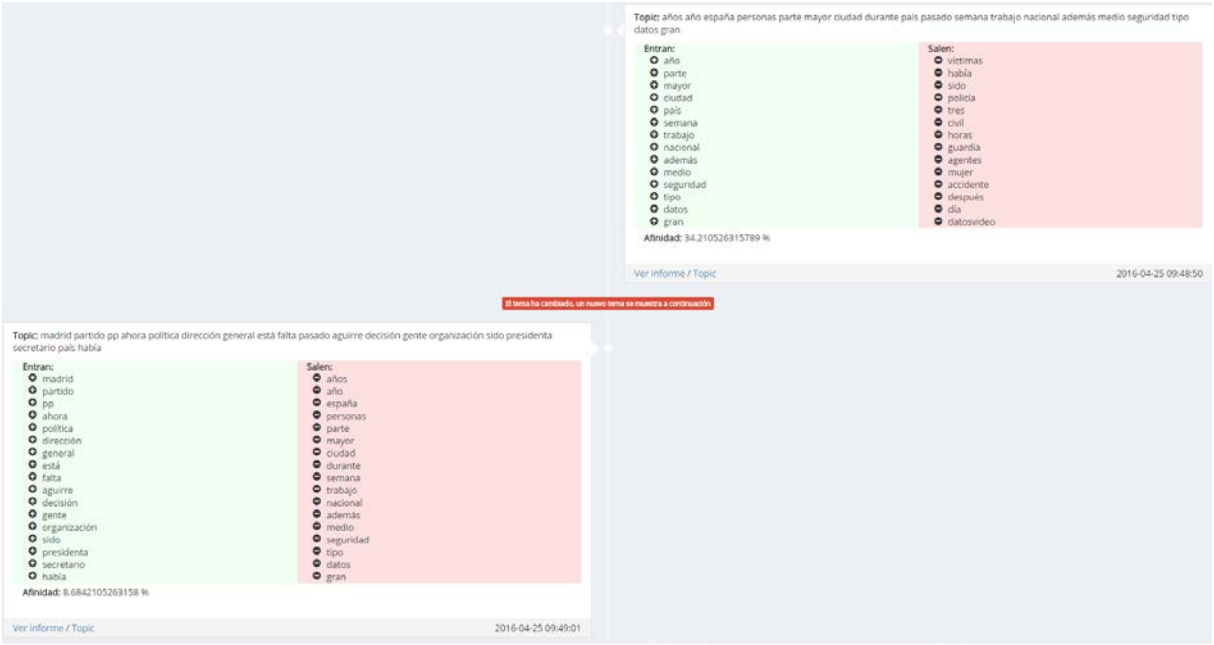

Ilustración 4.23

5. Resultados

De la fase de ejecución de los algoritmos obtendremos unos informes con un tema en cada informe, para evaluar los resultados tendremos que tener un sistema de evaluación apropiado y uno o varios corpus de documentos sobre los cuales ejecutar las pruebas.

Para evaluar si los resultados son temas de calidad o no lo son, usaremos los criterios propuestos por David Mimno, el creador del software Mallet en una conferencia en 2012 para identificar a los temas malos.

Un tema de poca calidad será aquel que podamos categorizar como:

- 1. Aleatorio: El tema es un conjunto de palabras no relacionadas.
- 2. Intrusos: El tema contiene una o dos palabras no relacionadas
- 3. Aburrido: El tema es un conjunto de palabras demasiado generales. No describe nada concreto.
- 4. Quimera: El tema está formado por dos o más temas buenos no relacionados entre sí, normalmente conectados por un término genérico.

Si el tema cae en alguno de estos problemas diremos que es un tema malo, siendo el objetivo minimizar el número de temas malos obtenidos. Para reducir los temas de poca calidad se parametrizará al algoritmo de una forma concreta para cada sección y se realizará un estudio de cómo estos parámetros influyen en la calidad de los resultados obtenidos. El estudio de los parámetros se realizará sobre LDA y posteriormente se estudiarán los algoritmos alternativos, si alguno mejora a LDA, se estudiará su parametrización también.

5.1. Longitud de los temas

El algoritmo nos devolverá los temas de la longitud especificada como entrada, sin embargo este parámetro no afectara a la ejecución del algoritmo. El algoritmo mantiene para cada tema una tabla con todas palabras y su importancia para dicho tema representada con un número. Cuanto mayor es el número, más importante será la palabra para el tema.

Si especificamos que el tema debe tener 20 términos de longitud, el algoritmo simplemente tomará las primeras 20 palabras y las presentará por orden, ese será el tópico. Lo que realmente se computa son los pesos de esas palabras para el tópico para finalmente quedarse con un subconjunto formado por las mejores.

Si visualizamos un tema concreto en cualquier informe veremos las palabras que lo forman en la parte superior:

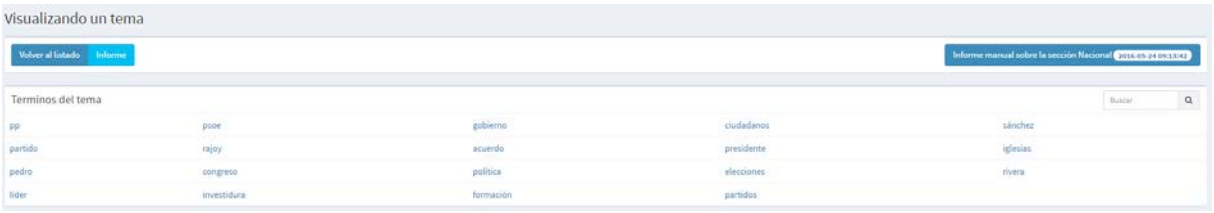

Ilustración 5.1

Y si nos fijamos justo debajo veremos el listado con el resto de palabras y su puntuación, en esta lista las 20 primeras palabras serán las que forman en tema y el resto podría haber entrado si hubiera conseguido una puntuación mayor (se posicionarían entre las 20 primeras) o si hubiéramos especificado una longitud de tema mayor.

Palabras destacadas

Buscar

Palabra	Peso
pp	3481
psoe	3176
gobierno	2748
ciudadanos	2592
sánchez	2486
partido	2271
rajoy	2133
acuerdo	1792
presidente	1759
iglesias	1400
pedro	1343
congreso	1197
política	1195
elecciones	1182
rivera	1160
líder	1154
investidura	1152
formación	901
partidos	851
pablo	837
está	825
socialista	818
general	770
portavoz	766
pacto	752
cambio	735
mariano	709
secretario	699
socialistas	679
izquierda	671
hoy	651
reunión	626
candidato	619
político	584

Ilustración 5.2

Se mostrarán todas las palabras en esta lista junto a su respectiva puntuación.

Con esto, podemos entender que la longitud de los parámetros no afecta directamente a la ejecución de los la algoritmos. Lo que se estudiará la calidad de los temas en base a los resultados buenos o malos que ofrezcan.

Generaremos un tópico aleatorio limitando la longitud del tópico hasta encontrar la longitud adecuada. El algoritmo sacará siempre un término menos que la longitud especificada, la si contamos el número de palabras siempre es una menos pero mantendremos esta numeración para evitar confusiones.

Para temas de longitud 3:

# Tema	Tema	Criterios incumplidos
0	año años	1 (Aleatorio)
1	pp psoe	3 (Aburrido)
2	caso euros	3 (Aburrido)
3	había les	1 (Aleatorio)
4	gobierno estado	3 (Aburrido)

Tabla 5.1

Los temas tan cortos no aportan suficiente información. Además podemos ver fallos a la hora de eliminar palabras vacías en el tema 3.

Para temas de longitud 5:

# Tema	Tema	Criterios incumplidos
0	gobierno estado cataluña españa generalitat	3 (Aburrido)
1	euros caso madrid juez valencia	3 (Aburrido)
2	años personas año tres españa	1 (Aleatorio)
3	pp psoe ciudadanos gobierno sánchez	1 (Aleatorio)
4	había día ahora ese hecho	3 (Aburrido)

Tabla 5.2

Los temas aún no son lo suficientemente concretos, si se está al día de las noticias es posible entender temas como el 0 y el 3 pero fuera de contexto no nos dicen nada.

Para temas de longitud 10:

# Tema	Tema	Criterios incumplidos
0	euros caso juez valencia investigación dinero empresa pp millones	
1	madrid ahora había hecho sido haber aguirre presidenta cómo	2 (Intrusos)
2	estado españa gobierno cataluña ley está generalitat mas proceso	
3	pp psoe gobierno ciudadanos sánchez rajoy partido presidente acuerdo	
4	años personas tres policía horas año ciudad mujer durante	1 (Aleatorios)

Tabla 5.3

Los temas empiezan a referirse a hechos más concretos. Solo hay uno que muestra términos no relacionados mientras que otro se acerca a crear un tema "bueno" pero añade palabras no relacionadas.

Para temas de longitud 15:

# Tema	Tema	Criterios incumplidos
0	gobierno estado cataluña ley generalitat españa mas tribunal ministro constitucional derecho social está presidente	
1	años había sido después durante casa día policía momento hecho tres nóos juicio fiscal	2 (Intrusos)
2	euros madrid caso juez valencia pp nacional ayuntamiento grupo millones investigación fuentes dinero pasado	
3	pp psoe ciudadanos sánchez gobierno partido rajoy acuerdo presidente iglesias pedro congreso elecciones investidura	
4	años españa año personas centro ciudad semana además mayor gran barcelona país durante mundo	1 (Aleatorios)

Tabla 5.4

Los resultados son similares a los anteriores pero los temas ganan significado al añadir más términos, hay una mayor calidad.

Para temas de longitud 20:

# Tema	Tema	Criterios incumplidos
0	españa años año millones trabajo mayor social parte además gran país está personas ahora seguridad semana servicios tipo ciudad	1 (Aleatorios)
1	gobierno estado cataluña presidente ley generalitat tribunal mas proceso está ministro constitucional españa derecho consejo parlamento catalán ejecutivo hecho	
2	euros caso madrid juez valencia ayuntamiento civil investigación nacional dinero grupo pp empresa guardia fiscalía fuentes pasado audiencia millones	
3	había años día sido durante horas después casa días nóos mujer les ese tres hecho personas tenía torres rey	2 (Intrusos)
4	pp psoe ciudadanos sánchez partido gobierno rajoy acuerdo iglesias pedro presidente congreso elecciones investidura rivera líder política formación socialista	

Tabla 5.5

Resultados similares a longitud 15. A partir de este valor empeoran los resultados.

Para temas de longitud 25:

# Tema	Tema	Criterios incumplidos
0	gobierno psoe ciudadanos sánchez pp rajoy acuerdo presidente iglesias pedro partido congreso elecciones investidura rivera líder partidos socialista pacto mariano formación socialistas cambio españa	
1	españa año millones ley años estado además parte social euros generalitat está gobierno trabajo sistema plan país servicios forma junta público ministerio sociales economía	4 (Quimeras)
2	caso euros juez valencia tribunal había investigación madrid dinero fiscal fiscalía nóos ayuntamiento empresa audiencia juzgado josé declaración delito barberá causa durante juicio grupo	
3	años día durante vida sido había les tres ciudad personas policía horas pasado gran después lugar mujer vez días agentes centro semana víctimas calle	1 (Aleatorios)
4	partido pp madrid política cataluña está ahora ayer estado frente general decisión dirección presidente mas proceso hecho aguirre sido constitucional después país presidenta parlamento	4 (Quimeras)

Tabla 5.6

Empiezan a aparecer temas que son mezcla de dos temas distintos realmente, se pierde la precisión de los temas.

Para temas de longitud 30:

# Tema	Tema	Criterios incumplidos
0	pp psoe gobierno ciudadanos sánchez partido rajoy acuerdo presidente iglesias pedro congreso elecciones investidura rivera líder política formación partidos socialista pablo cambio pacto portavoz está general mariano socialistas secretario	
1	había hecho ahora día después cómo durante casa dijo les ese gente nóos rey años mundo sido está momento hizo dicho fuera juicio josé torres haber tenía urdangarin ver	2 (Intrusos)
2	euros caso madrid juez valencia pp ayuntamiento investigación nacional dinero empresa partido aguirre fiscalía grupo audiencia empresas millones comunidad delito barberá juzgado pasado josé tribunal causa número delitos cargo	
3	años personas tres españa sido policía horas año mujer ciudad durante centro lugar cinco pasado civil semana agentes víctimas calle nacional guardia además interior días cuatro había mañana mujeres	1 (Aleatorios)
4	gobierno estado españa cataluña ley generalitat millones tribunal está mas proceso ministro constitucional derecho país social consejo además catalán reforma medidas año parte sistema constitución española forma debe parlamento	4 (Quimeras)

Tabla 5.7

Los resultados empeoran respecto a 25.

Los mejores resultados se han obtenido con 15 y con 20 palabras, probaremos a buscar en ese rango: 15 + 20 = 35/2 = 17.5 -> 18.

Para temas de longitud 18:

# Tema	Tema	Criterios incumplidos
0	años tres año personas policía españa sido ciudad horas centro semana lugar durante civil además gran mujer	1 (Aleatorios)
1	pp psoe gobierno ciudadanos sánchez partido rajoy acuerdo presidente iglesias pedro congreso elecciones investidura rivera líder política	
2	euros caso madrid juez valencia ayuntamiento pp investigación dinero millones empresa grupo aguirre nacional fiscalía tribunal empresas	
3	gobierno estado cataluña españa ley generalitat mas proceso ministro constitucional tribunal país derecho social está millones parlamento	
4	había durante hecho ahora años día después ese casa les está sido nóos dijo madrid haber gente	2 (Intrusos)

Tabla 5.8

Exactamente igual que 15 y 20, un Topic con intrusos y uno aleatorio. Nos quedaremos con el rango 15 a 20 como el rango adecuado.

En la tabla a continuación se resumen los resultados:

Longitud Tema	Temas malos	Notas
3	5	
6	5	
10	2	Temas con poco significado
15	2	Adecuado
18	2	Adecuado
20	2	Adecuado (Valor por defecto)
25	3	
30	3	

Tabla 5.9

En el siguiente capítulo se discutirá como obtener mejores resultados.

5.2. Parámetro alfa

El parámetro alfa modifica el peso documento-tema. Para el estudio de este parámetro nos basaremos en la información recogida en los informes, se generarán temas con la misma configuración y variando este parámetro. Adicionalmente se fijará la semilla aleatoria para poder repetir las pruebas manteniendo los resultados.

El informe nos ofrecerá una tabla para cada documento en la cual se muestra la afinidad de ese documento para los distintos temas. Si tenemos 5 temas, mostrará un valor de afinidad para cada uno en el rango 0 a 1. Todos los valores deberán sumar 1 (el 100 %). Si un documento tiene un valor del 0.70 de afinidad con un tema y un 0.10 con el resto de temas, ese documento aportará mucha más información al primer tema que al resto, ese es el significado del parámetro. Sin embargo el hecho de que un documento tenga un valor alto para un tema no implica que ninguna de sus palabras aparezca en el mismo ya que hay un gran número de documentos, simplemente significa que aportas significativamente más a ese tema que al resto.

Tomaremos el documento “No sin mi «smartphone»” obtenido de http://sevilla.abc.es/tecnologia/moviles/telefonia/abci-internacional-telefono-movil-no-sin-smartphone-201604011747_noticia.html .

Para el artículo dado, se ejecutará el algoritmo para los valores 1, 25, 50, 75, 100, 200 y se verá como varia el valor de la varianza.

Valor alfa	Varianza	Desviación Típica
1	0.018013599537037	0.1342147515627
25	0.016711145546373	0.1292715960541
50	0.0098262712371738	0.099127550343856
75	0.0063106008357333	0.079439290251949
100	0.011087616710609	0.10529775263798
200	0.0043985865577701	0.066321840729658

Tabla 5.10

Si representamos los valores de varianza gráficamente obtendremos:

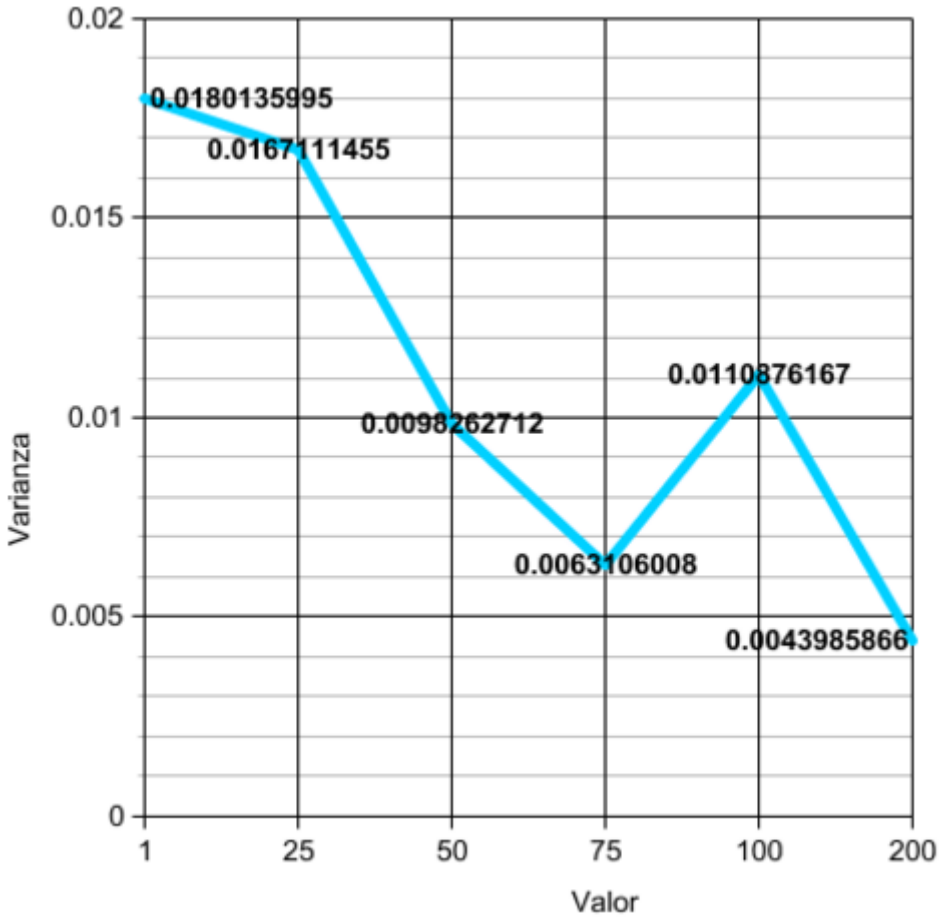

Ilustración 5.3

En general la varianza va reduciéndose según el valor de alfa aumenta, lo cual significa que según aumentamos este valor de este parámetro, el peso documento-tema se reparte de forma más uniforme (menor varianza) entre los distintos temas. En 100 aumenta cuando debería seguir bajando pero es algo que puede pasar dado el componente aleatorio.

Si realizamos el mismo estudio para otro documento obtenemos los mismos resultados, incluyendo el resultado que destaca para el valor 100. A continuación los valores:

Valor alfa	Varianza	Desviación Típica
1	0.011114456697655	0.10542512365492
25	0.0055277531527925	0.074348861139848
50	0.0030215379138483	0.054968517479083
75	0.00081539639302802	0.028555146524366
100	0.0037337567740808	0.061104474255825
200	0.00040673361619005	0.020167637843586

Tabla 5.11

Cuya representación es:

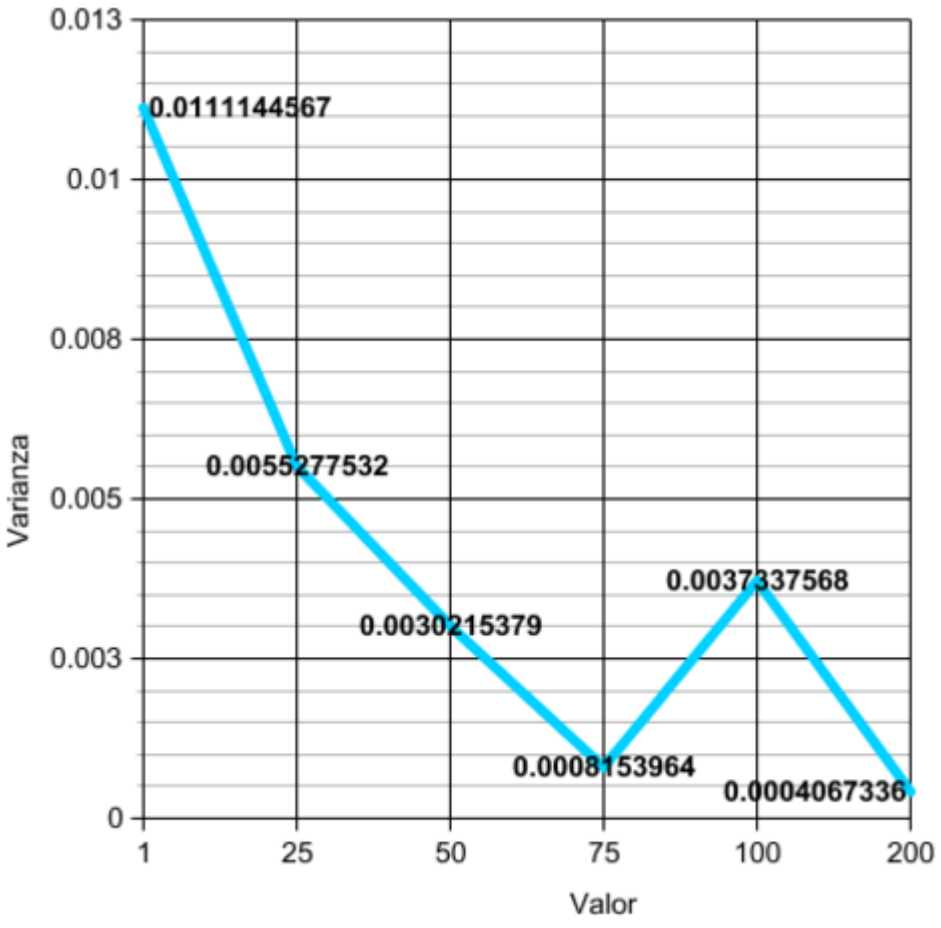

Ilustración 5.4

Los resultados y la interpretación son similares. Como mantenemos todos los parámetros iguales incluyendo la semilla, el valor atípico sigue apareciendo.

Una vez que sabemos cómo se comporta el parámetro, lo llevaremos hasta sus límites. Para esta parte del estudio usaremos las propias tablas que se generan en los informes. Generaremos un informe con 5 temas y distintos valores del parámetro alfa.

Para alfa = 0.0001:

Tema	Afinidad documento-tema
0	0.65270072381909
1	0.11715827789632
2	0.23012426258316
3	0
4	0

Tabla 5.12

Para un valor bajo de alfa, el documento aporta muchísimo a un tema, algo a otros dos y nada a los dos restantes.

Para alfa = 0.01:

Tema	Afinidad documento-tema
0	0.31380748376256
1	0.2468619050787
2	0.43933044379479
3	0
4	0

Tabla 5.13

Se reparte de forma más uniforme.

Para alfa = 1:

Tema	Afinidad documento-tema
0	0.4175
1	0.33833333333333
2	0.18416666666667
3	0
4	0.055

Tabla 5.14

Para alfa = 100:

Tema	Afinidad documento-tema
0	0.23893805309735
1	0.35988200589971
2	0.17994100294985
3	0.11799410029499
4	0.10324483775811

Tabla 5.15

Para alfa = 10000:

Tema	Afinidad documento-tema
0	0.19962887000684
1	0.20255884363707
2	0.19933587264381
3	0.19933587264381
4	0.19914054106846

Tabla 5.16

Para valores muy bajos de alfa (menores a 0.0001) lograremos hacer que los documentos aporten toda la información posible a un solo tema, concentrando su contenido.

Para valores de alfa muy bajos (mayores a 10000) haremos que la información del documento se divida a partes casi iguales a lo largo de todos los temas.

Valores en el rango 1 a 100 repartirán la información de un documento beneficiando principalmente a un tema pero sin dejar a los demás sin nada.

5.3. Parámetro beta

El valor del parámetro beta se encarga de priorizar el peso de las palabras individuales respecto a los temas. Normalmente todas las palabras obtienen un mismo valor beta, los valores de este parámetro suelen estar por debajo de 1.

Cuando más alto sea este valor, más se diversificará cada palabra entre los distintos documentos.

Para el estudio del parámetro se generará un conjunto de informes con la semilla aleatoria fija y con los mismos valores, salvo el parámetro beta que tomará los valores 0.0001, 0.001, 0.01, 0.1, 0.5, 0.75 y 1. En estos informes estudiaremos el parámetro en 3 palabras que serán sistema, web y facebook.

Palabra sistema:

Para beta = 0.0001 la palabra solo aporta a un tema:

Tema	Peso
0	800
1	0
2	0
3	0
4	0

Tabla 5.17

Para beta = 0.001 la palabra aporta a dos temas y distribuye el peso:

Tema	Peso
0	571
1	229
2	0
3	0
4	0

Tabla 5.18

Para beta = 0.01 la palabra aporta tres temas:

Tema	Peso
0	112
1	148
2	0
3	540
4	0

Tabla 5.19

Para beta = 0.1 la palabra aporta a cuatro temas:

Tema	Peso
0	449
1	264
2	0
3	86
4	1

Tabla 5.20

Para beta = 0.5 la palabra aporta a todos los temas:

Tema	Peso
0	496
1	144
2	155
3	1
4	4

Tabla 5.21

Para beta = 0.75 la palabra aporta a todos los temas:

Tema	Peso
0	532
1	76
2	82
3	91
4	19

Tabla 5.22

A medida que aumentamos el parámetro beta, la palabra aporta información a más temas distintos y reparte más la información que aporta a cada tema.

Palabra web:

Para beta = 0.0001 la palabra solo aporta a un tema:

Tema	Peso
0	0
1	0
2	0
3	386
4	0

Tabla 5.23

Para beta = 0.001 la palabra solo aporta a un tema:

Tema	Peso
0	0
1	0
2	0
3	386
4	0

Tabla 5.24

Para beta = 0.01 la palabra solo aporta a un tema:

Tema	Peso
0	0
1	0
2	0
3	386
4	0

Tabla 5.25

Para beta = 0.1 la palabra aporta a dos temas:

Tema	Peso
0	0
1	234
2	0
3	134
4	0

Tabla 5.26

Para beta = 0.5 la palabra aporta tres de los temas:

Tema	Peso
0	1
1	309
2	58
3	0
4	0

Tabla 5.27

Para beta = 0.75 la palabra solo aporta a todos los temas:

Tema	Peso
0	0
1	217
2	149
3	2
4	0

Tabla 5.28

El comportamiento es similar pero el término aporta información a menos temas, esto podría deberse a que en el contexto dado la palabra sistema se comporta de una forma más genérica que web, siendo web un término concreto que no encaja con todos los temas.

Palabra facebook:

Para beta = 0.0001 la palabra solo aporta a un tema:

Tema	Peso
0	0
1	806
2	0
3	0
4	0

Tabla 5.29

Para beta = 0.001 la palabra solo aporta a un tema pero el tema al que aporta tiene una numeración distinta al anterior:

Tema	Peso
0	0
1	0
2	0
3	806
4	0

Tabla 5.30

Esto puede deberse a que los cambios en el parámetro hayan provocado que el mismo tema se situé en una posición distinta, o bien que la propia ejecución del algoritmo ha decidido que es más adecuado para el tema 3 en lugar del 1 debido a que para valores tan bajos de alfa no se reparte el peso entre varios temas y se debe elegir uno u otro. Si ambos tenían probabilidades similares, todo depende de la elección al azar basándose en dichas probabilidades.

Para beta = 0.01 la palabra solo aporta a un tema:

Tema	Peso
0	806
1	0
2	0
3	0
4	0

Tabla 5.31

De nuevo el peso cambia de tema.

Para beta = 0.1 la palabra aporta a dos temas:

Tema	Peso
0	0
1	17
2	0
3	789
4	0

Tabla 5.32

Para beta = 0.5 la palabra aporta tres de los temas:

Tema	Peso
0	0
1	1
2	635
3	167
4	3

Tabla 5.33

Para beta = 0.75 la palabra solo aporta a todos los temas:

Tema	Peso
0	0
1	62
2	735
3	6
4	3

Tabla 5.34

En este ejemplo tenemos un caso ligeramente anómalo, en el mayor valor de beta para la palabra facebook, salen 3 temas (menos que con los valores 0.75 y 0.5).

Como exíste una componente aleatoria es posible que el peso de los temas se distribuya así de forma normal. En lugar de tener 2 temas con menos de 10 repeticiones, tiene solo uno, siendo tan bajos estos temas no es relevante si hay uno o dos. Por lo demás se aprecia el mismo comportamiento.

5.4. Tiempo de ejecución de los algoritmos

Se estudiarán 3 ejecuciones de cada algoritmo, para un número de temas reducido (5), por defecto (20) y grande (100).

Se usará el valor 'real' de la instrucción de Ubuntu 'time'.

Algoritmo	Longitud 5	Longitud 20	Longitud 100
LDA	1m 23.826s	1m 40.075s	2m 7.970s
N-Gramas	3m 39.002s	5m 31.275s	14m 21.747s
4 Level PAM	2m 5.425s	8m 10.840s	29m 42.796s

Tabla 5.35

El algoritmo 4 Level PAM mostrado para una longitud de 100 está generado con 10 súper-temas, aunque es posible ejecutarlo con 50 súper-temas, en cuyo caso el tiempo de ejecución pasaría de 29m 42.796s a 118m 44.803s.

5.5. Evaluación N-Gramas

Este algoritmo nos ofrecerá un resultado como un tema formado por uni-gramas, similar al resto de algoritmos y un resultado formado por bigramas asociado al anterior. En lugar de obtener 5 temas cuando lo parametrizamos para tal objetivo, obtendremos 10, 5 bigramas y 5 uni-gramas.

A continuación ejecutaremos el algoritmo para una longitud de tema 20, generando previamente un modelo usando la opción keep-secuence-bigrams. Para mejorar la visualización separaremos uni-gramas y bigramas en dos tablas aunque realmente pertenecen a la misma ejecución.

# Tema	Tema	Criterios incumplidos
0 Unigramas	mas espana anos ano barcelona tambien ciudad semana personas centro victimas horas interior lugar gran dias castilla accidente alcalde cinco	1 (Aleatorios)
1 Unigramas	madrid mas ahora habia mundo direccion paso gente vida organizacion anos hizo comunidad dimision decision dia frente ver eta esperanza	1 (Aleatorios)
2 Unigramas	caso euros juez segun habia valencia investigacion guardia grupo fiscalia tres policia ayuntamiento empresa tambien anos dinero pasado pp audiencia	
3 Unigramas	pp gobierno psoe ciudadanos partido presidente acuerdo mas rajoy congreso sanchez investidura lider espana pedro formacion elecciones tambien cambio partidos	2 (Intrusos)
4 - Unigramas	estado cataluna mas generalitat tribunal ley millones gobierno derecho proceso justicia ministro parte forma sistema plan tambien consejo urdangarin declaracion	4 (Quimera)

Tabla 5.36

# Tema	Tema	Criterios incumplidos
0 Bigramas	semana_santa mayor_parte freginals_tarragona gobierno_municipal estados_unidos ciudad_real agencia_estatal pasado_ano tres_anos primera_hora gran_canaria ano_pasado direccion_general metros_cuadrados ultimos_anos victimas_mortales guerra_civil santa_cruz buena_parte policia_municipal	1 (Aleatorios)
1 Bigramas	noticia_completa gobierno_regional izquierda_abertzale primera_linea pablo_echenique ese_momento habia_estado seis_anos real_madrid ano_despues pasado_domingo dio_cuenta pasado_mayo artes_escenicas mas_justo mas_adecuado primera_persona mas_jovenes salvamento_maritimo tribunal_europeo	1 (Aleatorios)
2 Bigramas	guardia_civil audiencia_nacional tribunal_supremo instruccion_numero policia_nacional rita_barbera audiencia_provincial ministerio_publico fiscalia_anticorrupcion cinco_anos grupo_municipal tribunal_superior habia_sido ese_momento seguridad_social director_general caso_imelsa pp_valenciano caso_noos francisco_granados	
3 Bigramas	pedro_sanchez pablo_iglesias mariano_rajoy secretario_general albert_rivera partido_popular elecciones_generales tribunal_constitucional primera_vez artur_mas junts_pel secretaria_general lider_socialista pais_vasco mas_alla izquierda_unida formar_gobierno mayoria_absoluta esperanza_aguirre union_europea	4 (Quimera)
4 Bigramas	instituto_noos gobierno_central inaki_urdangarin infanta_cristina tribunal_superior caso_noos diego_torres dona_cristina gobierno_catalan consejo_general tribunal_constitucional manos_limpias ese_sentido cataluna_tsjc administraciones_publicas agencia_tributaria ministerio_fiscal miguel_tejeiro declaracion_independentista sector_publico	4 (Quimera)

Tabla 5.37

Se obtienen demasiados tenas con términos aleatorios que no se corresponden con nada y bastantes Quimeras, esto puede ser debido a que en los bigramas obtenidos hay 20 términos que están formados por 40 palabras (pueden ser menos palabras distintas como en ministerio_fiscal y ministerio_publico comparten una palabra), así que reduciremos el número de palabras que forma cada Topic a la mitad y nos centraremos en los bigramas. Con esto el número de palabras distintas estará en el rango estudiado como adecuado, de 15 a 20.

El resultado con la mitad de longitud es:

# Tema	Tema	Criterios incumplidos
0 Bigramas	tribunal_constitucional gobierno_central derechos_humanos junts_pel tribunal_superior artur_mas gobierno_catalan consejo_general parlamento_catalan ley_organica	
1 Bigramas	semana_santa ano_pasado mayor_parte medio_ambiente estados_unidos gobierno_municipal pasado_ano agencia_estatal ciudad_real cambio_climatico	1 (Aleatorio)
2 Bigramas	noticia_completa diez_anos gran_canaria violencia_machista nueva_york mas_importantes mas_grave iglesia_catolica santa_cruz policia_municipal	1 (Aleatorio)
3 Bigramas	pedro_sanchez pablo_iglesias secretario_general mariano_rajoy albert_rivera partido_popular elecciones_generales primera_vez mas_alla esperanza_aguirre	2 (Intrusos)
4 Bigramas	guardia_civil audiencia_nacional instituto_noos tribunal_supremo caso_noos instruccion_numero policia_nacional ministerio_publico inaki_urdangarin audiencia_provincial	

Tabla 5.38

Los resultados son de menor calidad que los que obtenemos tras ejecutar LDA base. Tenemos demasiados temas aleatorios que no se acercan a ningún tema real (1,2), uno que no aporta gran información y que tiene algún termino que no debería (en el 3).

Adicionalmente aparecen términos como "ano_pasado" y "pasado_ano" en el mismo tema.

5.6. Evaluación 4-Level PAM

Evaluaremos el algoritmo en condiciones similares a las de LDA. Realizaremos una prueba inicial que buscará 20 temas de longitud 20 usando 10 súper-temas.

#	Tema	Criterios
0	está política presidente hecho congreso dicho haya mariano ahora sea ningún mejor nadie asegurado hemos ese decir gente nueva parece	3 (Aburrido)
1	semana calle día hoy san santa horas gran año plaza primera durante vida maría aragón iglesia tarde jesús jornada obra	
2	parte días nacional general debe público haber falta ello respecto cargo sociales meses últimos propio sentido marcha estado número pasado	2 (Intrusos)
3	nóos rey torres urdangarin fiscal infanta lópez casa tejeiro cristina instituto manos aizoon marido caso declaración preguntas juicio abogado iñaki	
4	además día lugar centro mañana hora durante imagen mujeres miércoles punto historia acto será da presencia nombre equipo resto aún	1 (Aleatorio)
5	barcelona víctimas accidente horas personas interior fuentes hospital autobús tráfico fernández informado estudiantes ministro tarragona familias seguridad lunes jóvenes servicio	
6	tres pasado años parte durante primera meses josé alcalde mes seguridad cinco octubre seis julio mayo cuatro asegura enero pública	2 (Intrusos)
7	euros millones empresa empresas año servicios dinero administración públicos cuentas contratos total público además comunidad fundación informe contrato hacienda fondos	2 (Intrusos)
8	cataluña tribunal generalitat mas catalán resolución cup parlament justicia catalana independentista artur independencia catalanes proceso declaración psc derecho cdc recurso	
9	ciudad ayuntamiento civil datosvideo valencia zona fallas guardia pontevedra capital var zonas vehículos actualidad tipo turismo fiestas grados nieve	2 (Intrusos)
10	españa país española año proyecto madrid trabajo español españoles está mundo economía derechos europea mayor internacional personas derecho nuevos vez	3 (Aburrido)
11	gobierno estado ley constitucional junta social propuesta ejecutivo medidas parlamento caso consejo constitución plan control fuentes será sistema comisión pleno	
12	pp madrid partido corrupción aguirre rajoy popular presidenta comunidad portavoz gonzález esperanza regional dimisión decisión presidente dirección ayer política quiroga	
13	había años sido después les tenía haber hombre casa vez habían cuatro tuvo fuera cuenta vida puesto ahora iba dice	1 (Aleatorio)
14	psoe ciudadanos sánchez gobierno pp rajoy acuerdo pedro rivera investidura presidente socialista pacto líder elecciones socialistas partidos candidato reunión portavoz	
15	juez caso valencia investigación barberá fiscalía civil juzgado operación causa grupo nacional fuentes instrucción guardia blanqueo dinero trama está delito	
16	iglesias partido pablo secretario formación congreso iu líder general organización errejón compromís elecciones generales dirigentes semana izquierda dirección cambio político	
17	ese nuevo momento país primer frente políticos cualquier les político posible proceso vasco esto final después ahora sido programa posibilidad	2 (Intrusos)
18	mayor galicia castilla mancha metros toledo diputación gallego zaragoza cabeza semana norte besteiro días profesor tierra coruña comunidad defensa provincia	4 (Quimera)
19	policía años mujer prisión audiencia hechos agentes violencia cárcel madre tres delito caso sentencia padre víctima padres hija libertad acusado	-

Tabla 5.39

El 50% de los temas obtenidos es de calidad. No mejora de forma notable a LDA con su peor configuración y tarda varias veces más.

5.7. Evaluación LDA

La evaluación de la longitud del tema en el apartado 6.1 se realiza sobre LDA, así que dichos resultados nos valen para realizar la evaluación.

Todos los algoritmos se han evaluado con temas de longitud 20, y LDA solo mostraba 2 temas malos en dicha situación.

6. Discusión

En este capítulo se discutirá como mejorar los resultados que soluciones se han aplicado para corregir los casos que generan temas de baja calidad.

6.1. Solución a los temas categorizados como 1 – Aleatorios

Los temas aleatorios son aquellos que están formadas por palabras no relacionadas entre ellas. La solución que mejor funciona es actualizar la lista de palabras vacías para que estas no se incorporen al modelo ya que los temas “Aleatorios” suelen ser un conjunto de palabras vacías.

Para ello se actualizará la lista base añadiendo todos los pronombres, adverbios, preposiciones, determinantes y conjunciones.

Si aun así no mejora la calidad de los resultados, podemos modificar el parámetro alfa y parametrizarlo con un valor muy bajo (0.0001 o menor) según lo estudiado en el capítulo anterior. Esto provocará que los documentos aporten información de forma exclusiva a un tema en concreto y no dividan su información entre varios temas. Las noticias suelen estar centradas en un hecho concreto así que focalizar su información en un solo tema será favorable en la mayoría de los casos.

Modificar el parámetro beta también puede ser favorable si los temas aleatorios contienen palabras que también tienen los temas buenos. Según lo visto en el capítulo anterior, si este parámetro toma valores bajos, la palabra aparecerá en menos temas y si lo reducimos mucho, solo aparecerá en un tema.

Si le damos un valor bajo a beta y la lista de palabas vacías es de calidad, favorecerá a que las palabras que forman estos temas aleatorios solo aparezcan en los temas buenos ya que solo pueden aparecer en un tema para valores muy bajos de beta (0.0001 a 0.001).

Por el contrario si la lista de palabras vacías es insuficiente, modificar el parámetro beta podría empeorar los resultados. Como los términos formados por palabras vacías suelen aparecer juntos estos tendrán un gran peso, mayor al de cualquier tema y si un término solo puede aportar información a un tema, el elegido será el tema aleatorio. Esto puede provocar que los temas buenos pierdan palabras que les correspondían inicialmente.

6.2. Solución a los temas categorizados como 2 – Intrusos

Los temas con intrusos son aquellos que añaden una o dos palabras no relacionadas. Esto suele deberse a que un término demasiado común las relaciona con el tema. La eliminación de palabras vacías debería eliminar dicho término común evitando la inclusión de palabras no relacionadas.

Otra posibilidad es que el término sea demasiado largo, tal como se vio en el apartado 6.4 los temas deben tener una longitud en el intervalo 15 a 20. Añadir más términos implica dejar entrar a términos con menores puntuaciones, y deja entrar a “términos intrusos”. Si los términos detectados como intrusos se encuentran al final del tema, sin duda la solución es reducir el tamaño del tema.

6.3. Solución a los temas categorizados como 3 – Aburridos

Los términos categorizados como aburridos son aquellos que a pesar de mostrar palabras relacionadas, no aportan información alguna. Suelen aparecer en temas cortos, debido a que necesitamos más términos en el tema para comprenderlo y que sea útil. La solución simplemente es aumentar la longitud del tema.

Otra opción viable puede consistir en diversificar incrementando el parámetro de alfa para permitir que más documentos aporten información.

6.4. Solución a los temas categorizados como 4 – Quimeras

Los términos categorizados quimeras mezclan dos o más términos buenos y normalmente están conectados por una palabra en común. La solución es eliminar esa palabra en común añadiéndola a la lista de palabras vacías si procede.

Otra posibilidad es que aparezcan temas quimera cuando se diversifica demasiado el significado de los términos, en otras palabras cuando el valor de alfa es demasiado alto, en cuyo caso habría que reducirlo a valores más normales.

6.5. Mejorar resultados en Nacional e Internacional

Las noticias de estas categorías se obtienen de las secciones Política, España o Nacional de los medios. Estas noticias destacan porque los temas de los que hablas son muy persistentes en el tiempo y suelen aparecer noticias relacionadas a lo largo de las siguientes semanas e incluso meses como puede ser el caso de las elecciones. Además normalmente todos los medios cubren las noticias principales de estas secciones, debido a estos dos factores, los resultados que se obtienen de la aplicación de los algoritmos son muy buenos ya que el algoritmo no tiene dificultades para encontrar temas buenos y para diferenciarlos claramente evitando temas “Quimera” o “Aleatorios”.

Estas categorías se procesarán con valores centrados de alfa y beta ya que no hay necesidad de potenciar nada. Los valores serán alfa = 50, beta = 0.01 y longitud 20.

Resultados:

```
0       caso euros juez valencia investigación madrid dinero tribunal fiscal grupo
ayuntamiento josé fiscalía nacional empresa delito parte audiencia pp

1       millones españa euros trabajo comunidad social mayor está personas tipo
sistema datos parte proyecto plan público junta total servicios

2       psoe pp partido ciudadanos gobierno sánchez rajoy acuerdo iglesias
elecciones congreso pedro líder formación rivera presidente partidos pablo
electoral

3       madrid horas ciudad sido lugar policía semana vida personas les civil
calle centro agentes mujer guardia san hombre toledo

4       gobierno españa estado presidente cataluña país está generalitat política
derecho ministro hecho proceso ley funciones sido constitucional pasado parlamento
```

6.6. Mejorar resultados en Deportes

Lo primero que observamos en esta sección es que los temas resultantes normalmente tienen intrusos que suelen agruparse al final del tema, la parte final del tema se corresponde con la de menor importancia. Es muy común ver temas que relacionan a dos equipos deportivos que se han enfrentado en un partido destacado pero unos términos después de aparecer el segundo equipo aparece un tercero. Esto es debido a que o bien el primer equipo o el segundo se enfrentaron a este tercero y están relacionados pero no lo suficiente pues el tercer equipo aparece al final del tema. Esto hace que el tema pierda significado. Si eliminamos las últimas 5 palabras del tema, reduciendo el parámetro de longitud a 15, el tercer equipo desaparece y obtenemos un tema lo suficientemente concreto. Esto no solo ocurre con equipos sino que ocurre similarmente con los nombres de los jugadores en deportes como el tenis.

Esto lo podemos observar en el ejemplo a continuación. Se resaltarán en negrita los equipos/nombre de jugadores:

```
0       puntos partido equipo valencia basket var final minutos baskonia kutxa
rebotes mejor temporada victoria campo espanyol granada nba asistencias

1       equipo madrid partido gol liga final barça barcelona jugadores fútbol
partidos entrenador goles técnico jugador luis atlético real amarilla

2       set nadal rafael golpe derecha final punto gran pista torneo resto rafa
red djokovic juego español saque partido número

3       canasta falta personal real falla madrid rebote triple barcelona tiro
libre lassa fc defensivo sergio encesta asistencia balón muerto

4       carrera está pasado sido mundial equipo mundo españa mejor gran club
temporada hecho presidente millones selección será vida deporte
```

Ahora eliminamos las últimas 5 palabras (las menos significativas), que sería equivalente a parametrizar el algoritmo con una longitud de tema de 5 palabras menor.

```
0       puntos partido equipo valencia basket var final minutos baskonia kutxa
rebotes mejor temporada victoria

1       equipo madrid partido gol liga final barça barcelona jugadores fútbol
partidos entrenador goles técnico

2       set nadal rafael golpe derecha final punto gran pista torneo resto rafa
red djokovic

3       canasta falta personal real falla madrid rebote triple barcelona tiro
libre lassa fc defensivo

4       carrera está pasado sido mundial equipo mundo españa mejor gran club
temporada hecho presidente
```

Esto provoca que los temas se reduzcan a enfrentamientos concretos que pueden aportar una información más útil.

En cuanto a los valores alfa y beta, alfa se mantendrá en su valor por defecto de tal forma que una palabra distribuirá su significado de forma normal por los distintos temas sin centrarse en uno exclusivamente y sin aportar significado a todos los temas pues puede que no tengan nada que ver. El valor de alfa será el 50.

El parámetro beta es el que distribuye el significado de los documentos en uno o varios temas. Tendrá un valor bajo pero no excesivamente para permitirle un poco diversificar. El valor será el 0.25.

Resultados con estos parámetros:

```
0       club españa selección fútbol presidente millones juegos sido jugador caso
pasado euros río mundial

1       set puntos nadal final partido rafael golpe derecha gran punto pista juego
mejor equipo

2       equipo gol partido liga barça madrid goles balón atlético luis marcador
centro área final

3       canasta falta madrid real personal falla barcelona rebote triple libre
tiro fc defensivo lassa

4       equipo carrera mejor está mundo ganar vida hemos gran pasado mundial sido
hecho dijo
```

6.7. Mejorar resultados en Tecnología y Arte

En estas secciones los resultados son peores ya que las noticias normalmente son puntuales y no trascienden en el tiempo ni se vuelve a hablar del mismo tema concreto de nuevo, esto dificulta al algoritmo encontrar los temas con claridad.

Para mejorar los resultados usarán vales bajos en los parámetros alfa y beta. Esto provocará que una palabra este asociada a un solo tema y que un documento solo aporte información a un tema, con esto centramos la información en crear temas concretos.

Además no todos los medios digitales cubren las mismas noticias en estas secciones, así que se precisará contar con una mayor cantidad de fuentes para suplir esta falta de información.

Los mejores temas se obtienen con valores muy bajos de alfa y beta (0.0001 en ambos):

```
0       millones juego euros virtual dólares realidad juegos compañía mundo nuevo videojuegos ventas gran nueva jugadores título playstation xbox historia

1       usuarios aplicación red usuario google millones twitter móvil internet está aplicaciones través personas servicio social redes forma mundo compañía

2       tecnología está mundo internet futuro cómo digital ejemplo   proyecto empresas universidad datos españa forma cosas empresa google gran

3       apple pantalla sistema mercado cámara dispositivo iphone nuevo euros móvil está samsung precio batería cuenta permite móviles dispositivos gb

4       datos seguridad apple información google caso compañía usuarios empresa internet protección iphone privacidad millones pasado acceso web sistema sido
```

Estos temas nos hablan a grandes rasgos de la realidad virtual y como irrumpe en plataformas de videojuegos como playstation y Xbox (primer tema), sobre las redes sociales en el segundo tema, sobre avances en dispositivos móviles en el cuarto tema y sobre la preocupación existente por seguridad y privacidad en el último tema.

En cuanto a los resultados para la sección Arte:

```
0       0.00002 muestra prado bosco juan san real ejemplo cuenta antonio sociales
cuadro director josé garcía teatro dalí nacional redes taller

1       0.00002 artistas mundo arte grandes ver fotografía trabajo momento
imágenes está imagen forma mejor galería época explica manera cómo propio

2       0.00002 ciudad millones madrid museo euros pasado parte centro cultura
españa está proyecto edificio nuevo patrimonio casa gran mercado sido

3       0.00002 obras arte exposición obra artista museo gran siglo nueva pintura
pintor barcelona colección piezas picasso historia muestra york cuadros

4       0.00002 vida foto gente dice cosas sido está parece libro autor feria
embargo arco hombre veces ampliar cómic país cualquier
```

Como se aprecia sigue apareciendo un tema aleatorio tal como pasaba en Tecnología

7. Conclusiones

Es muy interesante como un algoritmo puede encontrar relaciones entre documentos sin entender el significado de los mismos y describir de qué se está hablando con unas pocas palabras bien escogidas.

A continuación propondré mejoras para las distintas fases del proyecto.

La fase de extracción de datos en la cual se recoge el cuerpo de las noticias se ha ejecutado de la forma más flexible posible dado el tiempo disponible para el proyecto, sin embargo debería ser posible explorar formas de obtener las noticias de una forma más sencilla.

Una de estas formas sería crear un extractor de noticias genérico que descargue las páginas web y almacene los nodos y su contenido por separado. Este extractor debería poder encontrar cual es la información que cambia completamente en cada uno de los documentos descargados. Las páginas web normalmente tienen una gran cantidad de contenido estático (header, footer) y contenido semi-dinámico como noticas relacionadas o noticias destacadas, pero el cuerpo de la noticia debería ser un texto único para cada uno de los documentos descargados. Este extractor eliminaría la necesidad de buscar e indicar un selector DOM en la aplicación para resaltar el cuerpo de la noticia, sin embargo no está libre de dificultades.

Otra aproximación válida para el extractor podría ser realizar una sencilla extensión de navegador (también conocidas como plugins) que permita seleccionar la zona de la web a extraer de forma visual, y del mismo modo que elementos dentro de esta zona debe eliminar. Algunos bloqueadores de publicidad ya incorporan opciones similares, en los al pasar el ratón por los elementos de la web se resaltan permitiendo elegir un elemento concreto y al hacer clic te muestran el selector de ese elemento y te preguntan si deseas bloquearlo. A continuación un ejemplo con AdblockPlus

Ilustración 7.1

Para la fase de visualización sería interesante implementar una búsqueda que encuentre los documentos más relevantes para cada uno de los temas ya que a pesar de que un tema tiene significado por sí solo, sería interesante ver de qué noticias ha salido dicho tema principalmente para poder consultarlas.

Esta propuesta podría mejorarse aún más si al realizar la búsqueda de los términos clave a lo largo del corpus de documentos, se almacenasen las frases o párrafos en las que co-ocurren dos o más términos y se mostrase dicha un tema como un conjunto de estos extractos de las noticias originales. En otras palabras contextualizar un tema a partir de las frases o párrafos en los cuales se han dado las relaciones entre los términos.

8. Anexo 1 - Instalación

Necesitamos una máquina con una imagen de Ubuntu server 14.04.4 LTS instalada. Se puede encontrar la descarga y más información en el sitio web de Ubuntu: http://www.ubuntu.com/.

Para un entorno de desarrollo es posible usar la máquina virtual homestead que incluirá gran parte del software instalado. Más información en https://laravel.com/docs/5.2/homestead.

8.1. Instalar un servidor web

Tanto apache como Nginx son completamente compatibles con la aplicación final, solo cambia ligeramente la configuración inicial.

Si se elige apache la documentación estará disponible en: http://httpd.apache.org/. El módulo mod_rewrite debe estar habilitado y la directiva AllowOverride debe tener el valor "All" para la carpeta de la aplicación web.

Si se elige Nginx, encontramos más información en http://nginx.org/. Hay que añadir la directiva en la configuración del sitio por defecto:

```
location / {
 try_files $uri $uri/ /index.php?$query_string;
}
```

Una guía paso a paso está disponible en https://www.digitalocean.com/community/tutorials/how-to-install-laravel-with-an-nginx-web-server-on-ubuntu-14-04.

8.2. Instalar Composer

Composer se encargará de servir los paquetes PHP necesarios para el proyecto de forma automática y de mantenerlos actualizados. Más información en: https://getcomposer.org/

8.3. Instalar MongoDB y el driver de MongoDB para PHP

Seguiremos la guía de php.net:

- En la línea de comandos ejecutamos sudo pecl install mongo
- Seleccionar 'no' pulsando enter
- Agregar permisos a php.ini: sudo chmod -R ugo+rw /etc/php5/cli/php.ini
- Agregar "extension=mongo.so" a php.ini: sudo echo "extension=mongo.so" >> /etc/php5/cli/php.ini
- Comprobar que ha agregado bien: cat /etc/php5/cli/php.ini | grep mongo
- sudo apt-get install mongodb
- Reiniciar php/servidor web para que se carguen los cambios: /etc/init.d/php5-fpm reload, en este caso.

Referencia adicional Instalación MongoDB:

http://php.net/manual/en/mongodb.installation.pecl.php

Notas:

- php está en dos carpetas fpm (producción) y cli (desarrollo), con <?php phpinfo() ?> en un fichero index.php se puede ver cuál de las dos está usando. La que aparece es en la que use hay que añadir la extensión mongodb.
- La extensión se puede agregar a mano, seleccionando donde ponerla, editándolo mediante vim.

- Es posible que php.ini este en otro lugar, o se encuentre adicionalmente en la carpeta /etc/php5/fpm/php.ini, en cuyo caso hay que darle permisos y agregar la extensión también. Documentación 28/09/2015
- Dependiendo de la versión de mongo, es posible que guarde la información en "/data/db", en ese caso hay que crear carpetas en esa localización, o modificar la localización en la configuración de mongo. Comprobar la carpeta actual.

8.4. Descargar el repositorio

Descargaremos el código del repositorio y lo colocaremos en el servidor web.

La carpeta public/ contiene la raíz del proyecto web y el servidor web tendrá que estar configurado para apuntar a ella. El index que se cargará inicialmente será el de esta carpeta. Añadir permisos de escritura a las carpetas storage y bootstrap/cache. Referencia adicional: http://laravel.com/docs/5.1

8.5. Configurar el entorno

Las contraseñas y el resto de información privada no se subirá al repositorio. En su lugar al descargar el repositorio, este contendrá un fichero llamado ".env.example", es un fichero que se debe configurar en el entorno en el que estemos y una vez configurado no debe salir de dicho entorno ni subirse a ningún repositorio, así las contraseñas están seguras.

Renombraremos el fichero .env.example a .env y empezaremos la configuración.

Primero definimos el entorno actual, si marcamos local, se mostrarán errores y la barra de debug. En producción los errores mostrará un error genérico y no habrá barra de producción. Elegir entre: APP_ENE=local o APP_ENV=production

Después configuraremos APP_KEY que se usará para cifrar todo tipo de información. Es recomendable generar un string aleatorio y colocarlo. También es posible rellenar este campo automáticamente ejecutando en consola: php artisan key:generate. Esto generará una nueva key y la colocará en el .env.

La información a continuación depende de tu instalación de la base de datos. Rellena los valores conocidos y comenta las lineas desconocidas con #, para que tomen valores por defecto. (Los valores por defecto se encuentran en la configuración de laravel y sí se subirán a un repositorio, los que añadas en el .env pisarán a los que hay por defecto y serán secretos)

Edita .env.example con los datos de tu entorno (desarrollo/producción / conexión a la db/app key), en nuestro caso quedará.

En nuestro caso quedaría algo así:

```
APP_ENV=local
APP_KEY=UnStringSecretoAqui

DB_HOST=localhost
DB_USERNAME=homestead
DB_PASSWORD=secret
DB_DATABASE=homestead
```

El resto de campos no son importantes en la aplicación, se pueden dejar en sus valores por defecto.

8.6. Instalar las dependencias

En la raíz del proyecto ejecutar desde consola: composer install para que Composer descargue e instale los paquetes que usará el proyecto. Requiere conexión a internet.

8.7. Cargar la información de prueba

En la raíz del proyecto ejecutar "artisan migrate" y "artisan db:seed". El primer comando creará las colecciones necesarias en Mongo.

El segundo poblará la BBDD con un conjunto de fuentes de datos, categorías y filtros para obtener información de las fuentes de datos. Este comando no es necesario, ya que se pueden añadir las fuentes manualmente, sin embargo es recomendable para ver el sistema en funcionamiento.

Las tablas se crearán en la BBDD con el nombre dado en el fichero .env en la directiva DB_DATABASE.

8.8. Preparar el cron

Para que las tareas definidas en el framework se ejecuten cuando deben, es necesario que alguien llame al framework continuamente para que este planifique y ejecute las tareas. Solo editaremos el cron una vez y podremos añadir todas las tareas que necesitemos en el framework:

Añade la entrada * * * * * php /path/to/artisan schedule:run >> /dev/null 2>&1 al cron de Ubuntu

Recuerda cambiar /path/to/artisan por la ruta hacia artisan (ejecutable que debe encontrarse en la carpeta del proyecto)

No es necesario reiniciar el cron tras editarlo.

8.9. Instalar java

Ejecutar: sudo apt-get update

¿Está instalado ya java?: java -version

Instalar java: sudo apt-get install default-jre

8.10. Instalar MALLET

Lo descargamos desde http://mallet.cs.umass.edu/ y descomprimimos la carpeta contenedora de MALLET en la carpeta del proyecto. Esta carpeta tendrá un nombre similar a "mallet-2.0.7", lo renombramos a "MALLET"

9. Anexo 2 – Requisitos Hardware

El servidor que ejecute la aplicación deberá tener el hardware mínimo que se detalla a continuación.

9.1. CPU

No es necesario medir la capacidad de la CPU, si se escoge mejor o peor simplemente variará el tiempo que tardan en ejecutar los algoritmos y que tarden algo más no es problema.

El proceso que más consume CPU es java al ejecutar los algoritmos de modelado de temas. Si ejecutamos el comando top de Ubuntu mientras ejecutamos 2 procesos de modelado de temas (generamos dos informes), obtenemos:

PID	USER	%CPU	%MEM	TIME+	COMMAND
2016	vagrant	99.9	5.1	0:28.82	java
2052	vagrant	99.6	4.9	0:28.41	java
1058	vagrant	15.6	2.2	0:03.94	php5-fpm
2072	vagrant	4.3	1.5	0:00.13	php5-fpm

Para cada proceso de modelado de temas que se quiera ejecutar el paralelo será necesario un núcleo de la CPU, si simplemente se desea ejecutar de uno en uno o dejar al sistema generar los informes semanalmente nos valdrá con 2 núcleos en la CPU, uno para realizar la ejecución y el otro para mantener el resto de servicios (web y base de datos principalmente).

Con una CPU de 4 núcleos podremos paralelizar 3 ejecuciones de algoritmos sin comprometer la estabilidad de la plataforma web.

9.2. Memoria

De nuevo, lo que más memoria consume es el proceso de modelado de temas. Ejecutando top podremos obtener el uso de memoria porcentualmente, lo hacemos mientras ejecutamos un proceso de modelado de temas que buscará 100 temas de tamaño 50 en 2300 documentos. El mayor valor de memoria observado fue:

```
PID USER %CPU %MEM TIME+ COMMAND
2289 vagrant 100.8 6.5 1:19.91 java
```

Si calculamos el valor del porcentaje obtenido sobre el total de memoria usado en el momento de la prueba (2Gb) obtenemos que el proceso ha usado 131.56 Mb.

Para procesar la salida de MALLET (leer los ficheros, recoger, categorizar la información y finalmente guardarla), ha necesitado 78.75 MB de memoria. Esta medición se ha obtenido usando PHP Debug bar.

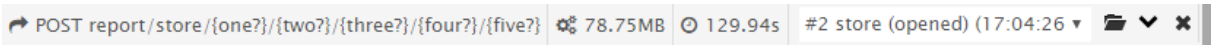

Ilustración 9.1

Ambos suman un total de 210.31 Mb pero no deben tratarse de forma conjunta pues el post-procesamiento empieza cuando el algoritmo ha acabado y los 131Mb iniciales ya se han liberado.

Si elegimos un número menor de temas, el proceso llega a usar el 6% de la memoria pero solo usa 35.75Mb para el post-procesamiento ya que los ficheros generados son considerablemente menores.

Esto es 121.44Mb del algoritmo y 35.75Mb del post-procesamiento.

Tomando ambas pruebas (una para una carga de trabajo grande y la otra para una carga habitual), obtenemos que el uso de memoria por cada proceso de modelado de temas, como mucho será menor a 150Mb.

Con 1Gb de RAM será suficiente para el funcionamiento automático del mismo, pero 2Gb permitirán realizar procesamiento paralelo sin afectar al resto del sistema.

9.3. Resumen

Requisitos mínimos:

- Conexión a internet
- 1Gb de RAM
- CPU de 2 núcleos

Requisitos recomendados:

- Conexión a internet
- 2Gb de RAM
- CPU de 4 núcleos

Bibliografía

Blei, D. M. (2011). *Probabilistic Topic Models.* Obtenido de https://www.cs.princeton.edu/~blei/papers/Blei2012.pdf

Blei, D. M. (2012). *Topic modeling.* Obtenido de https://www.cs.princeton.edu/~blei/topicmodeling.html

BLEVINS, C. (1 de 4 de 2010). *TOPIC MODELING MARTHA BALLARD'S DIARY.* Obtenido de http://www.cameronblevins.org/posts/topic-modeling-martha-ballards-diary/

Block, D. J. (2006). *Probabilistic Topic Decomposition of an Eighteenth-Century American Newspaper.* Obtenido de http://www.ics.uci.edu/~newman/pubs/JASIST_Newman.pdf

BRETT, M. R. (2012). *Topic Modeling: A Basic Introduction.* Obtenido de http://journalofdigitalhumanities.org/2-1/topic-modeling-a-basic-introduction-by-megan-r-brett/

Griffiths, M. S. (s.f.). *Probabilistic Topic Models.* Obtenido de http://cocosci.berkeley.edu/tom/papers/SteyversGriffiths.pdf

Jordan, D. B. (2003). *Modeling Annotated Data. SIGIR.* Obtenido de http://mimno.infosci.cornell.edu/topics.html

Jordan, D. M. (s.f.). *Latent Dirichlet allocation.* Obtenido de http://www.cs.princeton.edu/~blei/papers/BleiNgJordan2003.pdf

Lafferty, D. M. (2006). *Correlated Topic Models.* Obtenido de https://www.cs.princeton.edu/~blei/papers/BleiLafferty2006.pdf

McCallum, A. K. (2002). *Mallet*. Retrieved from Abour Mallet: http://mallet.cs.umass.edu/about.php

McCallum, D. M. (2008). *Topic models conditioned on arbitrary features with Dirichlet-multinomial regression.* Obtenido de http://mimno.infosci.cornell.edu/papers/dmr-uai.pdf

McCallum, W. L. (s.f.). *Pachinko Allocation: DAG-Structured Mixture Models of Topic Correlations.* Obtenido de https://people.cs.umass.edu/~mccallum/papers/pam-icml06.pdf

McCallum, X. W. (24 de 12 de 2005). *A Note on Topical N-grams.* Obtenido de https://people.cs.umass.edu/~mccallum/papers/tng-tr05.pdf

Mimno, D. (3 de 9 de 2012). *Topic Modeling Workshop: Mimno.* Obtenido de The details: training and validating big models on big data: https://vimeo.com/53080123

mimno, D. (17 de 2 de 2015). *Using phrases in Mallet topic models.* Obtenido de http://www.mimno.org/articles/phrases/

Mimno, D. (s.f.). *Topic Modeling Bibliography.* Obtenido de http://mimno.infosci.cornell.edu/topics.html

Regueiro, J. M. (21 de 09 de 2012). *Comparativa MongoDB vs MySQL.* Obtenido de http://www.ciges.net/tests-de-carga-mongodb-vs-mysql-insercion-de-datos

Rhody, L. (6 de 5 de 2015). *THE STORY OF STOPWORDS: TOPIC MODELING AN EKPHRASTIC TRADITION.* Obtenido de http://www.lisarhody.com/the-story-of-stopwords/

Shawn Graham, S. W. (s.f.). *Getting Started with Topic Modeling and MALLET.* Obtenido de http://programminghistorian.org/lessons/topic-modeling-and-mallet

Underwood, T. (7 de 4 de 2012). *Topic modeling made just simple enough.* Obtenido de https://tedunderwood.com/2012/04/07/topic-modeling-made-just-simple-enough/

Weingart, S. (25 de 07 de 2012). *Topic Modeling for Humanists: A Guided Tour.* Obtenido de http://www.scottbot.net/HIAL/index.html@p=19113.html

Wikipedia. (2016). Obtenido de Latent Dirichlet allocation: https://en.wikipedia.org/wiki/Latent_Dirichlet_allocation