UNIVERSIDAD DE OVIEDO

ASTURIAS
CAMPUS DE EXCELENCIA
INTERNACIONAL
AD FUTURUM

MÁSTER UNIVERSITARIO EN INGENIERÍA WEB

TRABAJO FIN DE MÁSTER

"OBTENCIÓN DE RESÚMENES AUTOMÁTICOS PARA TEMAS DE ACTUALIDAD EN MEDIOS SOCIALES"

Autor: Iván Teja Martínez

Director: Daniel Gayo Avello

Oviedo, Julio de 2014

Agradecimientos

Quería agradecer la ayuda recibida por parte de mi director Daniel Gayo que me ha ayudado inmensamente durante la realización del mismo y que pese a las diferentes circunstancias que han acompañado a este trabajo siempre me ha animado y apoyado para sacarlo adelante.

También quiero dar las gracias a todos los amigos que han colaborado conmigo en este trabajo, por las molestias que les hubiera podido causar, ya que han desempeñando un papel importante como evaluadores en la fase de evaluación de resultados.

No quiero olvidarme tampoco de mi familia y de Patricia, por todo lo que me han ayudado y por hacer que cada día merezca la pena.

Por último quería recordar y agradecer especialmente a mi abuelo que ya no está y al que echo de menos por todo lo que aportó en mi vida.

A todos ellos GRACIAS.

Resumen

Este proyecto se fundamenta en el desarrollo de un prototipo que implementa una serie de algoritmos con el fin de obtener resúmenes abstractivos a partir de un volumen considerable de tuits para un trending topic. Teniendo en cuenta que no existe a nivel de aplicaciones nada basado en ese enfoque, se hace un estudio de los resúmenes generados por el prototipo, siendo evaluados por humanos y analizando las posibilidades que aporta un prototipo de este estilo en la actualidad.

Cabe destacar que el desarrollo del prototipo que genera los resúmenes está implementado en Java y viene precedido de la construcción de otro prototipo en PHP, que se encarga de la descarga de tuits para los trending topic del momento mediante el uso del API de Twitter. Para la evaluación de la calidad de resúmenes se escogen una serie de trending topics para que sus tuits sean procesados por el prototipo principal generando así, una serie de resúmenes extractivos por cada algoritmo, que posteriormente son evaluados por una serie de personas para intentar analizar la calidad de estos resúmenes generados por cada algoritmo y poder ver si merece la pena costear la implementación de una aplicación que siga este enfoque o por lo contrario sería mejor que la obtención de resúmenes para trending topics se hiciese por resúmenes generados por humanos.

En cuanto a los posibles clientes que quisiesen que a partir de este prototipo se construyese una aplicación real, algunos candidatos podrían ser empresas que quisiesen hacer estudios de mercado sobre algún producto, otras relacionadas con el periodismo para obtener información de temas de actualidad, otras cuyo interés sea obtención de opinión general acerca de un tema, etc.

En definitiva, este proyecto trata de ver en qué medida la calidad de los resúmenes generados automáticamente por un prototipo es equiparable a la calidad de los resúmenes abstractivos producidos por personas y así poder considerarlos también resúmenes abstractivos. También se analizan las posibilidades que ofrece este prototipo y las mejoras que pueden llevarse a cabo.

Palabras Clave

Twitter, trending topic, resumen automático, resumen abstractivo, resumen extractivo, sumrank, textrank.

Abstract

This project is concerned with the development of a prototype which implements a series of algorithms. By doing this, abstractive summaries will be obtained from a considerable amount of tweets for a trending topic. Considering that no application based on this yet exists, research on the resultant summaries of the prototype was carried out. Humans evaluate the resultant summaries and then it is possible to analyse the possibilities that this kind of prototype could contribute to our current lives.

It is necessary to point out that the prototype development which generates the summaries is implemented in Java and it is preceded by another prototype construction in PHP. This latter prototype is in charge of downloading tweets for the current trending topics through the use of the API of Twitter. As far as the qualitative evaluation of the summaries is concerned, a series of trending topics are chosen, and corresponding tweets are processed by the main prototype. Thanks to this, we obtain a series of extractive summaries for each algorithm. A group of people have subsequently evaluated the process to try to assess the quality of these summaries generated by each algorithm and to see whether it merits financing an application implementation that follows this approach or to see whether it is better to obtain summaries for trending topics through summaries generated by humans.

Regarding possible clients who may want to have a real application constructed from this prototype, there will be some candidates such as businesses that would want to do a marketing study on a product. Others could be related to journalism, where, for example, the target is to obtain information about current affairs or the public opinion about a particular topic.

To sum up, this project will be based on analysing how far the quality of summaries generated automatically by a prototype is comparable to the quality of abstractive summaries produced by people, and in this way we can consider abstractive summaries too. The possibilities that this prototype offers and the improvements that could be carried out will also be taken into account.

.

Keywords

Twitter, trending topic, automatic summary, abstractive summary, extractive summary, sumrank, textrank

Índice General

Índice de Figuras

Capítulo 1 Introducción

1.1 Resumen de la Motivación, Objetivos y Alcance del Proyecto

Hoy en día, las redes sociales ocupan un puesto muy importante en la vida de millones de personas, utilizándose en gran medida para expresar opiniones, estados de ánimo, vivencias y demás información que puede ser muy interesante analizar.

Una de estas redes sociales, **Twitter**, es un fenómeno social que ha alcanzado más de 200 millones de usuarios, donde éstos pueden enviar sus **tuits** (mensajes de un máximo de 140 caracteres) categorizando estos mensajes por medio de las etiquetas llamadas **hashtags**.

Aproximadamente 500 millones de tuits son publicados cada día, generando una gran cantidad de información, que es una oportunidad única para la realización de aplicaciones que exploten esa información, ya sea para estudios de mercado, periodísticos, etc.…

Una de las características que proporciona Twitter son los **trending topics**, definidos como un listado de los temas más populares en tiempo real que más tuits generan. Pese a que en algunos de estos trending topics se puede intuir sobre qué tema se está tratando en sus tuits, en la mayoría no es así, por lo que se presenta un problema a la hora de realizar búsquedas sobre un tema determinado, ya que Twitter no indica la descripción de sus trending topic. Es por ello que este trabajo se va a centrar en avanzar hacia la obtención de resúmenes abstractivos de diversos temas de esta red social, a través de la implementación de unos prototipos que permitan hacer un estudio a la posibilidad de crear estas descripciones automáticamente.

Todo este proceso consistirá en la creación de dos prototipos, un prototipo para la descarga de información de Twitter y otro prototipo para la aplicación de los algoritmos sobre los tuits descargados y la posterior evaluación por parte de usuarios.

La intención es hacer que estos prototipos puedan ser extrapolables a su puesta en producción ya sea para que los usuarios de Twitter tengan la posibilidad de realizar búsquedas de información de un tema determinado, de empresas que decidan investigar la opinión general de los usuarios de Twitter, ya sea para hacer estudios de mercado, estadísticos…

A continuación se describirán detalladamente los aspectos más relevantes que se contemplarán en este trabajo.

Por otra parte es necesario hacer una pequeña introducción sobre todo el entramado contextual del trabajo y explicar así brevemente los puntos más característicos del mismo.

Como ya se ha mencionado anteriormente, la ausencia por parte de Twitter de proveer a sus usuarios de un mecanismo donde pueda consultar la descripción de un trending topic, ha

hecho que se plantee la posibilidad de obtener estas descripciones automáticamente y proceder a su evaluación.

Esta posibilidad pasa por la creación de dos prototipos que se encarguen de la descarga de datos y de la obtención de resúmenes con los algoritmos aplicados además de un proceso de evaluación de los resúmenes obtenidos.

En el prototipo encargado de la descarga de información se tuvo en cuenta que los trending topic van cambiando continuamente a lo largo del día, es por ello que se estipuló que cada 15 minutos se actualizase la lista de trending topics actuales a descargar.

La información es descargada en ficheros nombrados por el trending topic y la fecha al que corresponde. Todo este proceso tendrá como objetivo, permitir la selección manual de varios de estos temas que sirvan como entrada al prototipo de obtención de resúmenes.

Todo el conjunto de tuits de esos trending topics serán filtrados previamente y procesados generando como salida una cantidad de resúmenes extractivos para cada algoritmo, estos posibles resúmenes abstractivos serán evaluados por un grupo de usuarios (curators) que permitirán establecer diferentes ranking entre los algoritmos realizados e intentar alcanzar una interpretación razonable al comportamiento que tienen en temáticas diferentes.

1.2 Justificación del Proyecto

La gran cantidad de información que maneja Twitter añadida a la ausencia de una herramienta de búsqueda de información de trending topics hace que se haya planteado la cuestión de si sería posible mediante una aplicación de resúmenes automáticos ofrecer unas descripción aceptable para cada trending topic, para después plantear si a partir de esos resultados merecería la pena invertir en la implementación de un algoritmo que lleve a cabo esa tarea que probablemente tenga unos costes elevados, o por el contrario utilizar algún servicio de pago en donde los usuarios resuman manualmente la información.

Consistirá en la implementación de dos prototipos que permitan la descarga de información de Twitter, el procesamiento de esa formación para su posterior generación de resumen mediante el uso de una serie de algoritmos y la evaluación de los resúmenes creados.

Con este trabajo también se quiere que los prototipos implementados sirvan como hoja de ruta para una posible puesta en producción en una empresa, ya que ofrece múltiples posibilidades para compañías que por ejemplo deseen hacer estudios de marketing a través de información de las redes sociales.

1.3 Objetivos del Proyecto

- Avanzar en la posibilidad de obtener de forma automática resúmenes abstractivos
- Implementación de un prototipo para una aplicación de descarga de información de Twitter.
- Elección de algoritmos de resumen.
- Implementación de un prototipo que genere resúmenes para Trending Topics a partir de tuits.
- Obtención de resúmenes para una serie de temas elegidos manualmente
- Evaluación de los resúmenes obtenidos
- Generación de ranking para cada algoritmo en diversos aspectos
- Análisis de los ranking
- Estudio de viabilidad con respecto a resúmenes manuales

1.4 Estudio de la Situación Actual

En este trabajo se han evaluado una serie de alternativas existentes en el ámbito académico donde se han hecho diversos experimentos mediante la utilización de algoritmos de resumen, en el apartado de Conceptos teóricos del capítulo II se explican las técnicas utilizadas en el ámbito académico, por el contrario se ha comprobado que no hay aplicaciones en producción que sigan el enfoque de generar automáticamente resúmenes abstractivos, en cambio, sí se han analizado otras aplicaciones cuyo enfoque guarda una cierta relación desde el punto de vista de obtención de resúmenes y métodos de evaluación de contenido.

Se han estudiado en el ámbito de aplicaciones Whatthetrend, Buscador de Twitter, Storify y generador de galerías multimedia.

En el primer caso **Whatthetrend** se ha analizado debido su cometido de conseguir descripciones que resuman eventos de Twitter, pero en este caso los resúmenes no son generados con la ayuda de una aplicación de resumen automático sino que son generados por humanos; también se ha prestado especial atención a la herramienta de **búsqueda de Twitter** que pese a no seguir realmente el mismo enfoque de avanzar en cuanto a la obtención de resúmenes abstractivos, su sistema de evaluación podría ser aplicable en este trabajo.

Otra aplicación que se ha estudiado ha sido Storify, donde su proceso de extracción de información a partir de redes sociales para generar artículos e historias ha hecho que sea un punto de comparación interesante para este proyecto.

Por último también hay que destacar una aplicación de generación de galerías multimedia a partir de eventos procedentes de las redes sociales, que al igual que Storify, su generación de datos multimedia que tratan de explicar un tema es comparable con el objetivo de este trabajo pese a no tener el mismo enfoque en común.

A continuación se explicarán cada una de estas aplicaciones:

1.4.1 Herramientas parcialmente similares

1.4.1.1 Whatthetrend[1]

La primera aplicación que se ha analizado es Whatthetrend, una web que ofrece a sus usuarios la posibilidad de definir ellos mismos cada trending topic. Cuando se inició este proyecto se pretendía que fuese incluida en el proceso de evaluación de los resúmenes obtenidos, haciendo una comparación con las definiciones que completaban los usuarios de esta página, pero al poco tiempo de barajarse esa opción la página web había quedado en deshuso, y casi ningún usuario realiza definiciones actualmente.

[1] www.whatthetrend.com

Como se puede apreciar en la Figura 1.1, la web lista los trending topics actuales y permite que se realice la definición de los mismos seleccionando la opción Define.

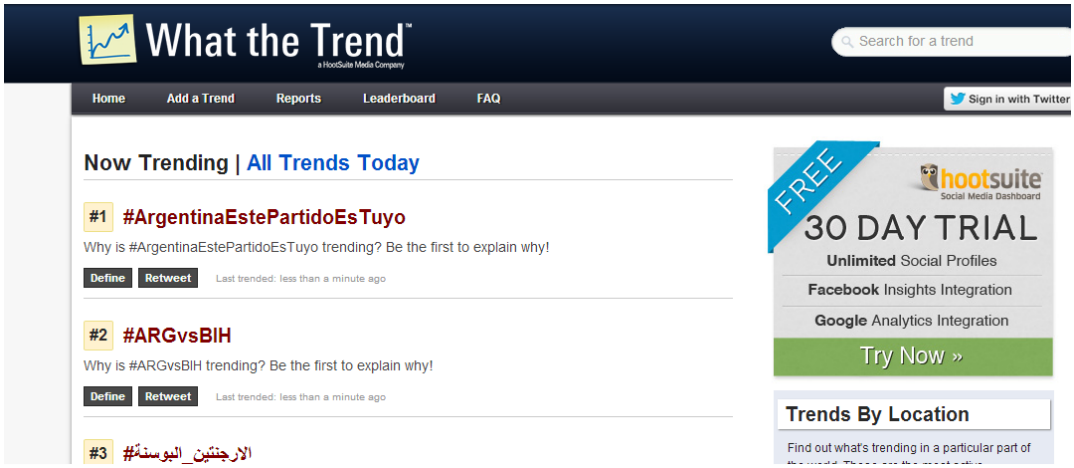


Figura 1.1 Página principal de Whatthetrend

1.4.1.2 Búsqueda de Twitter[2]

La segunda aplicación que se ha analizado es la propia herramienta de búsqueda de Twitter, que pese a que no aporte descripciones acerca de trending topics que facilite a sus usuarios buscar temas de un determinado contenido, su funcionamiento guarda relación en el sistema de aprendizaje de sus consultas con el sistema de aprendizaje que se podría aplicar en la generación de resúmenes tratada en este proyecto.

Partiendo de la base de que esta aplicación monitorea constantemente las consultas más populares en tiempo real, siguiendo una topología Storm que realiza un seguimiento de las estadísticas sobre dichas consultas, donde tan pronto como aparece una nueva consulta se envía a su evaluación humana a través de Amazon Turk, sus jueces devuelven información sobre la consulta que será añadida a la información utilizada por los modelos de aprendizaje automático del buscador.

Es por ello que un sistema de evaluación similar podría utilizarse para poder analizar la calidad de los resúmenes obtenidos y poder dar un paso más hacia la creación de un sistema de aprendizaje que obtenga resúmenes abstractivos para un trending topic.

[2] https://twitter.com/search-home

En la Figura 1.2 se puede observar la interfaz de la herramienta de búsqueda de Twitter.

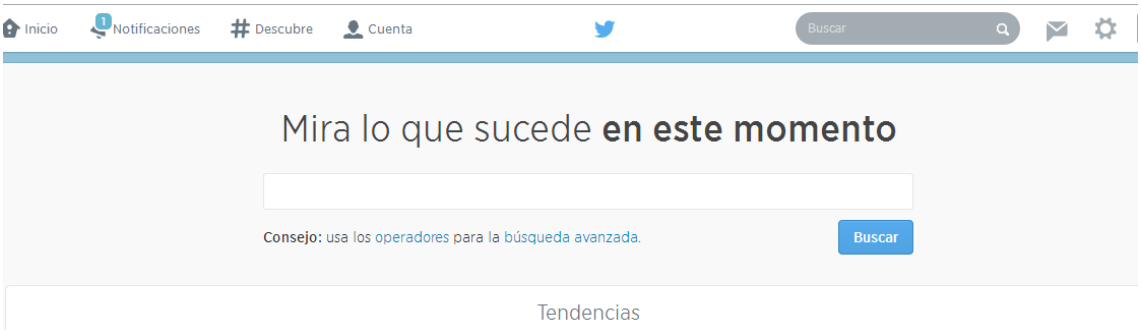


Figura 1.2 Página principal del buscador de Twitter

En la siguiente figura se observa cómo al escribir en el buscador un texto, Twitter ofrece una serie de tendencias

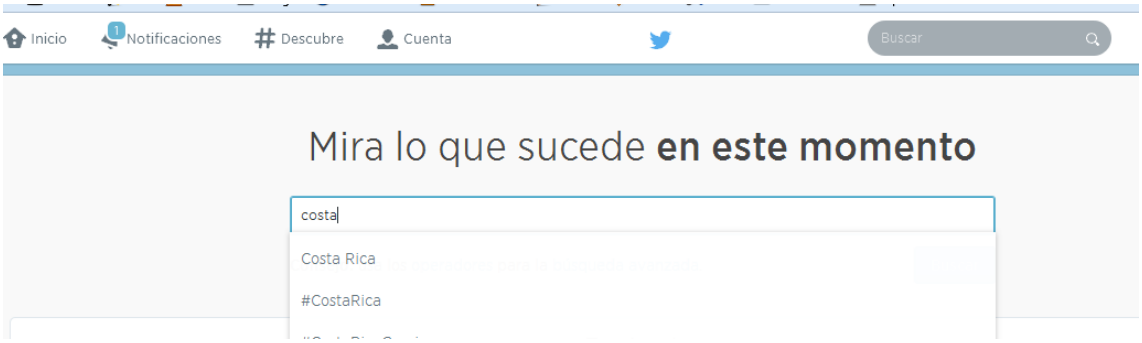


Figura 1.3 Vista de sugerencias en el buscador de Twitter

En la Figura 1-4 se observa el resultado de la búsqueda de #CostaRicaGrecia

Figura 1.4 Vista del resultado de una búsqueda en Twitter

1.4.1.3 Storify[3]

Es un sitio web que utilizando contenido de redes sociales permite a sus usuarios crear y almacenar historias, noticias y demás información sobre un tema, está orientada al mundo del periodismo.

A través de una herramienta de búsqueda, Storify genera todos los resultados pertenecientes a un tema solicitado por un usuario, que pueden ser añadidos a la historia creada, con la opción de añadirle título, descripción y texto. La relación de esta aplicación con este proyecto reside en la relación de contenidos procedentes de redes sociales con un tema en particular, donde los usuarios manualmente le asignan un título y descripción. En este caso se estaría ante una técnica de resumen manual.

En la Figura 1.5 se muestra la página principal de Storify donde sus usuarios pueden crear su historia.

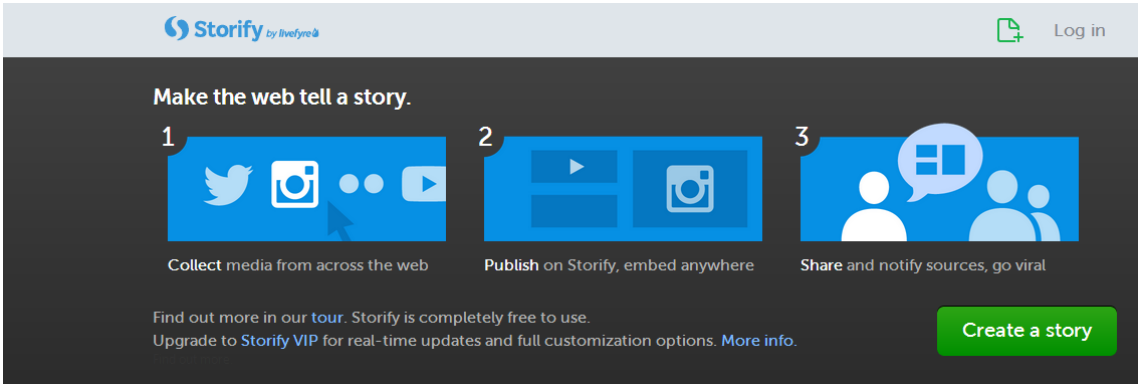


Figura 1.5 Página principal de Storify

En la Figura 1.6 se puede observar un ejemplo de una historia creada a través de Storify:

Figura 1.6 Ejemplo de historia en Storify

[3] https://storify.com/

1.4.1.4 Generador de Galerías Multimedia[4]

Es una aplicación para Twitter que genera galerías multimedia a partir de eventos basados en elementos multimedia compartidos en las redes sociales.

Para su generación sigue los siguientes pasos, que guardan una relación en la forma de aplicación de los algoritmos de resumen en este trabajo:

- Recuperación de una lista de eventos potencialmente relevantes a través de la API de búsqueda de diferentes redes sociales.
- Mediante el uso de herramientas de Procesamiento del Lenguaje Natural se reconocen y eliminan entidades que aparecen en microposts con el fin de determinar su relevancia.
- Se extraen datos binarios de los elementos multimedia de las redes sociales o los elementos multimedia de plataformas de almacenamiento.
- Utilizando técnicas de recuperación de contenido basado en video y en imágenes, se eliminan datos redundantes de los elementos multimedia y se almacenan los elementos multimedia similares.
- Se puntúan todos los elementos multimedia siguiendo un criterio de puntuaciones
- Se hace un ranking a partir de las puntuaciones obtenidas donde se generarían las galerías multimedia de los Top n elementos con mayor puntuación.

Aunque el objetivo principal de esta aplicación no sea el mismo que el objeto de este trabajo fin de máster, utiliza un sistema de ranking de puntuaciones que coincide con el usado en algunos de los algoritmos de extracción de resúmenes que se implementaron.

En la siguiente imagen se puede observar esta aplicación de generación de galerías multimedia que resumen un evento.

Figura 1.7 Aplicación de galerías multimedia

[4] https://twitter.com/mediagalleries

Capítulo 2 Aspectos Teóricos

A continuación se describirán aquellos conceptos, herramientas y tecnologías que serán utilizados en este proyecto cuyo significado es imprescindible para saber en rasgos generales la base sobre la que se sustentará este proyecto.

Inicialmente se explicaran aquellos conceptos tales como qué es Twitter, los algoritmos utilizados en la generación de resúmenes y

2.1 Twitter

Es una red social con un servicio de microblogging que permite a sus usuarios enviar y leer **tuits** (mensajes de un máximo de 140 caracteres), además estos pueden ser compartidos por otros usuarios retwiteándolos. Fundado en Marzo de 2006 por Jack Dorsey, Twitter se ha convertido en una de las más populares redes sociales del momento con aproximadamente 500 millones de usuarios, generando diariamente gran cantidad de información.

Twitter muestra en su página de inicio los 10 **trending topics** más relevantes en tiempo real, dichos trending topics son las palabras más repetidas en un momento concreto, tienen su origen en los **hashtag**, etiquetas precedidas por la almohadilla utilizadas para la organización de tuits sobre un tema en concreto, su gran repercusión ha hecho que sean denominados como temas de gran interés, sus temas suelen ser de gran variedad, en un principio la mayoría eran memes y coberturas de eventos, ahora las noticias de la actualidad tienen cada vez mayor repercusión.

Existen multitud de aplicaciones que trabajan con Twitter, como por ejemplo aquellas que permiten actualizar información por parte de sus usuarios, revisar los trending topics actuales,… todo esto es posible gracias a que Twitter proporciona una API para desarrolladores que ofrece multitud de funcionalidades. Este aporte por hace que sea posible su integración en infinidad de escenarios.

2.2 Conceptos Teóricos

A continuación se explicarán diferentes conceptos estudiados sobre diversas técnicas de resumen utilizadas en el campo académico, de las cuales algunas tienen su influencia a la hora de desarrollar el prototipo de generación de resúmenes de este trabajo. Muchas de estos métodos se utilizaron para diferentes experimentos como por ejemplo extracción de resúmenes en tiempo real de microblogs para eventos programados,

2.2.1 Resumen Extractivo

La generación de resúmenes extractivos Se basa en la construcción de resúmenes a partir de sentencias extraídas del documento original tras la aplicación de una serie de heurísticos tales como la utilización de palabras clave o que apareciesen en el título.

Alguna de estas técnicas de extracción de palabras clave son las siguientes:

- **Frecuencia de los términos:** Donde a partir de las estadísticas de la frecuencia de sus términos, se procede a una puntuación de sus oraciones, donde las que tengan mayor puntuación serán las frases candidatas a ser el resumen. La medida más común para el cálculo de las frecuencias de la palabra es TF-IDF que se explicará el apartado 2.2.4 de este capítulo.
- **Ubicación:** Basado en la intuición de que las sentencias importantes se encuentran en cierta posición en el texto o en el párrafo, como el principio o el final de un párrafo
- **Método de señales**: Detección de palabras que tendrían efecto positivo o negativo en el respectivo peso de una frase para indicar el significado o idea clave. Por ejemplo: “En resumen”, “en conclusión”, etc.
- **Palabras/Título:** Se asume que los títulos y las cabeceras de los documentos de un documento contienen las palabras claves que reflejan la idea general del texto.
- **Longitud de la oración**: Parte de que las frases cortas no son valorables como resumen ya que aportan demasiada información y las muy largas contenidos inapropiados.
- **Similitud**: Utilizando esta característica se determinaría la similitud entre una sentencia y el resto del documento.

2.2.2 Resumen Abstractivo

Los resúmenes abstractivos consisten en obtener una síntesis de un texto totalmente nuevo que recoja las ideas principales del mismo, donde por lo general requiere de técnicas generación de lenguaje avanzado y de compresión, trabajando de una forma similar a la del ser humano.

A continuación se enumeran algunos métodos utilizados en la obtención de resúmenes abstractivos:

Basados en un enfoque estructurado: Donde codifica la información más importante del documento (s) a través de esquemas cognitivos, como plantillas, reglas de extracción y otras estructuras, como el árbol, la ontología, y la estructura del cuerpo de la frase.

A continuación se enumeran diferentes métodos que usan este enfoque:

- **Método basado en árboles:** se utiliza un árbol de dependencias para representar el contenido de un documento donde mediante el uso de diferentes algoritmos como el algoritmo de intersección.
- **Método basado en plantillas:** se utiliza una plantilla para representar un documento completo. Patrones lingüísticos o reglas de extracción se adaptan a identificar

fragmentos de texto que se asignan en los campos de la plantilla. Estos fragmentos de texto son indicadores del contenido para el resumen.

- **Método basado en ontologías:** muy utilizado en la web donde la mayoría de los documentos pertenecientes a un dominio están relacionados entre sí al discutir del mismo tema o evento.
- **Método basado en cabecera y cuerpo de frases:** se basa en las operaciones de las frases (inserción y sustitución) que tienen la misma cabecera sintáctica con el fin de reescribir la oración principal.
- **Método basado en reglas:** los documentos que se resumen se representan en términos de categorías y una lista de aspectos. El módulo de selección de contenido selecciona al mejor candidato entre los generados por reglas de extracción de información para responder a uno o más aspectos de una categoría. Por último, los patrones de generación se utilizan para la generación de sentencias resumen.

Basados en un enfoque semántico: la representación semántica de documento (s) se utiliza para alimentar al sistema de generación de lenguaje natural. Está enfocado en la identificación de las frases nominales y frases verbales mediante el procesamiento de los datos lingüísticos.

A continuación se enumeran diferentes métodos que usan este enfoque:

- **Modelo semántico Multimodal**: se construye un modelo semántico que captura los conceptos y relaciones entre conceptos con el fin de representar el contenido de los elementos multimodales. Los conceptos importantes se clasifican en función de una cierta medida y, finalmente, los conceptos seleccionados se expresan en frases de forma resumida.
- **Método basado en artículos de Información:** el contenido del resumen se genera a partir de una representación abstracta de los documentos de origen. La representación abstracta es el artículo de la información, que es el elemento más pequeño de información coherente en un texto.
- **Método semántico basado en gráficos:** pretende resumir un documento mediante la creación de un grafo semántico para el documento original, aplicando la reducción del mismo para su posterior generación del resumen abstractivo.

2.2.3 Algoritmo Aleatorio (RANDOM)

Algoritmo que elige aleatoriamente k mensajes por tema como resumen. Este método ha sido elegido con el fin de proporcionar un peor caso de rendimiento y establecer un límite inferior de rendimiento.

2.2.4 Algoritmo Hybrid TF-IDF

Este ha sido uno de los algoritmos elegidos para su implementación en el prototipo de generación de resúmenes. Está basado en el enfoqué TF-IDF que explicaremos a continuación por su importancia para lograr entender este algoritmo **Hybrid TF-IDF**.

TF-IDF es una técnica de ponderación estadística muy utilizada en áreas de indexación automática, de búsqueda de coincidencias en documentos, así como en la elaboración de resúmenes automáticos. Sin embargo, a pesar de su empleo extendido a la hora de generar resúmenes automáticos su enfoque es bastante simplista. En general, el enfoque TF-IDF consiste en asignar a cada una de las frases que conforma un documento un peso que refleje la prominencia de cada una de ellas, siendo el peso de cada frase el resultado de la suma de los pesos de los términos individuales que la constituyen (palabras, frases o cualquier otro tipo de función léxica) (. El resultado final serán frases ordenadas por sus pesos, siendo aquellas con pesos superiores las que son elegidas como resumen del documento.

La fórmula que se emplea para determinar el peso de un término es la siguiente:

$$TF\_IDF = tf_{ij} * log_2 \frac{N}{df_j}$$

Donde tf_{ij} es la frecuencia del término Tj dentro del documento Di, N es el número total de documentos y df_j es el número de documentos dentro del conjunto que contiene el término T_j

El valor final de TF-IDF se compone de dos partes principales:

1. **TF** (Frecuencia de término): asigna más peso a las palabras que aparecen frecuentemente en un documento al considerar que las palabras que se repiten son las más importantes **[Luhn58]**.
2. **IDF** (Frecuencia Inversa de documento): debido a que palabras vacías, como puntos o comas, son comunes en un documento, pero no ayudan a discriminar el peso de una frase sobre otra, este parte del algoritmo compensa esto penalizando a estas palabras en proporción a su frecuencia inversa en el documento. Es por tanto, una corrección del efecto del componente TF.

Este método es bastante simplista y una de sus limitaciones es que la fórmula no es sensible a la longitud del documento. Respecto a esto, autores como Singhal et al., (1996) se dieron cuenta que los documentos más extensos tienen términos con frecuencias más altas, ya que se repiten con más frecuencia, en contraste a los documentos menos extensos. Esta limitación es sobre todo evidente cuando se genera un resumen de múltiples documentos debido a que los términos que forman parte de documentos largos tienen más peso.

Esta limitación se debe sobre todo a que la ecuación para determinar el peso de un término está hecha para documentos tradicionales y no para mensajes de microblogs. De este modo, una posible opción sería definir un documento simple que contenga todos los mensajes. En este caso, el componente TF es sencillo ya que se pueden contabilizar las frecuencias de los términos a lo largo de todos los mensajes. Sin embargo, debido a que se tiene solo un documento, se pierde el componente IDF. Otra alternativa opuesta a la anterior consistiría en definir cada post como un documento, lo cual haría la definición del componente IDF más clara, pero afectaría negativamente al componente TF ya que cada post contiene una pequeña

cantidad de palabras y por tanto, la frecuencia de los términos será una pequeña constante para un post determinado.

Para manejar esta situación, se redefinió TF-IDF en términos de un documento híbrido. Primero se define un documento como un post simple, pero cuando se contabilizan las frecuencias de los términos se asume que el documento es el conjunto entero de todos los mensajes. De este modo, no sólo se ha conseguido diferenciar las frecuencias de los términos, sino que no se pierde el componente IDF, siendo un término una palabra en una oración.

A continuación, se debe de elegir un método de normalización para compensar el sesgo del algoritmo hacia posts largos, lo cual consiste en dividir en peso de una frase por un factor de normalización. Por otro lado, el problema de las palabras vacías se resuelve dando a cada una de ellas un peso de cero a través de la comparación con una lista codificada previamente. A continuación, se presenta el resumen del algoritmo **[Sharifi10]**:

$$W(S) = \frac{\sum_{i=0}^{\#\ PalabrasEnSentencia} W(w_i)}{nf(S)}$$

$$W(w_i) = tf(w) * log_2(idf(w_i))$$

$$tf(w_i) = \frac{\#\ AparicionesDePalabraEnTodosLosPosts}{\#\ PalabrasEnTodosLosPosts}$$

$$idf(w_i) = \frac{\#\ SentenciasEnTodosLosPosts}{\#\ SentenciasQueContienenLaPalabra}$$

$$nf(S) = max[\ MínimoUmbral\ , \#\ PalabrasEnSentencia\ ]$$

Donde W es el peso asignado a una frase o una palabra, nf es un factor de normalización, wi es la palabra i, y S es una sentencia.

2.2.5 Algoritmo SumBasic

SumBasic se caracteriza por ser un sistema que produce resúmenes de múltiples documentos genéricos **[Vanderwendea07]**. La elaboración de su diseño fue motivada por la observación que palabras que aparecen frecuentemente en un documento aparecen con mayor probabilidad en los resúmenes elaborados por personas que las palabras que han sido menos citadas.

El algoritmo en el que se basa SumBasic puede dividirse en los siguientes pasos:

- **Paso 1**. Calcular la distribución de probabilidad sobre las palabras wi que aparecen en la entrada, P (wi) para todo i, p (wi) = n/N, donde n es el número de veces que la palabra apareció en la entrada, y N es el número total de contenido fichas de palabras en la entrada.
- **Paso 2**. Para cada frase Sj, asignar un peso que sea igual a la probabilidad media de las palabras en la frase. Sería el encargado de hacer que las palabras que aparecen con mayor frecuencia en la entrada, sean las que con mayor probabilidad sean citadas en los resúmenes realizados por personas. El cálculo de del peso se realiza de la siguiente forma:

$$Weight(S_j) = \sum_{Wi \in Sj} \frac{p(wi)}{|\{wi|wi \in Sj\}|}$$

- **Paso 3**. Seleccionar la frase con mejor resultado que contiene la palabra con más elevada probabilidad. Con este paso nos aseguramos que la palabra con mayor probabilidad de aparición es incluida en el resumen. De este modo, cada vez que se selecciona una frase, la palabra con más alta probabilidad hasta ese punto del documento también es seleccionada.
- **Paso 4.** Para cada palabra wi en la frase elegida en el paso 3, se debe actualizar su probabilidad mediante el siguiente algoritmo:

$$p_{new}(w_i) = p_{old}(w_i) * p_{old}(w_i)$$

Gracias a este paso se cumplen tres objetivos. Primero de todo, nos ofrece feedback acerca de lo que es importante incluir en el texto en base a la información previa ya incluida. De hecho, mientras que $p_{old}(w_i)$ puede considerarse como la probabilidad con la que la palabra w_i se incluirá en el resumen, $p_{new}(w_i)$ es una aproximación de la probabilidad con que la palabra w_i aparecerá en el resumen dos veces. A continuación, mediante la actualización de la probabilidad de aparición de las palabras también permite que palabras que inicialmente

tenían poca probabilidad al final tengan alto impacto en la elección de las frases siguientes. Por último, esta actualización o la probabilidad de la palabra permiten de forma natural hacer frente a posibles redundancias en el caso de multi-documentos.

- **Paso 5.** En el caso de que no se haya logrado la longitud deseada en el resumen, se debe de volver al paso 2 previamente descrito.

En resumen se puede decir que SumBasic es un algoritmo que trabaja con probabilidades, que computa el peso de una frase igual a la probabilidad media de las palabras en una oración y que permite resumir multi-documentos y resuelve el problema de la redundancia a través de la actualización de la probabilidad de las palabras en base a las frases previas.

2.2.6 Algoritmo TextRank

En este proyecto se ha aplicado este algoritmo en su especialidad de palabras clave, con el fin de obtener un grafo con las palabras clave del algoritmo y así poder hacer un ranking de resúmenes puntuando cada tuit con la suma del peso de sus palabras clave, La aplicación de este algoritmo, por tanto tiene como objetivo final obtener un conjunto de palabras o frases representativas de un determinado texto.

De este modo, secuencias de una o más unidades léxicas, extraídas del texto original, son objeto de clasificación y conforman un grafo de texto en el que cualquier relación que pueda ser definida entre dos unidades léxicas representa una conexión potencial entre ellas. A la hora de establecer esas relaciones entre unidades léxicas, este algoritmo se basa en una relación de coocurrencia, es decir, relaciona dos palabras de un texto si su aparición dentro del texto está dentro de una ventana de un máximo de N palabras, donde N se corresponde con cualquier punto entre 2 y 10 palabras. De este modo, estos enlaces de coocurrencia expresan relaciones entre elementos sintácticos y representan indicadores de cohesión para un texto dado.

Las relaciones entre unidades léxicas se representan en grafos de texto en los que se pueden establecer filtros para seleccionar un solo tipo de unidad sintáctica (nombres o verbos) de una cierta parte del texto. Por ejemplo, uno puede considerar solo nombres y verbos y en consecuencia establecer enlaces basados únicamente en las relaciones posibles entre nombres y verbos.

Los pasos para la obtención de palabras claves a través del algoritmo TextRank son los siguientes:

- **Paso 1:** se realiza una fase de pre-procesamiento necesaria para posteriormente aplicar los filtros sintácticos. En esta primera fase se realiza un etiquetado gramatical (speech tags) en el que se asigna a cada una de las palabras de un texto su categoría gramatical. Por otro lado, para evitar grafos de gran tamaño solo se tienen en cuenta palabras individuales como candidatas para ser incluidas en el grafo de texto.
- **Paso 2:** todas las palabras que cumplen los requisitos para pasar el filtro sintáctico son añadidas al grafo y una unión se establece entre aquellos elementos que coocurren dentro de una ventana de N palabras. Inicialmente, una vez que el grafo ha sido construido, la puntuación asociada con cada unión es un valor inicial de 1, ejecutando posteriormente el algoritmo hasta que el gráfico converge en un umbral de 0.0001.

  Específicamente, G = (V,E) es un grafo dirigido con un conjunto de vértices (V) y un grupo de uniones (E), donde E no es más que un subconjunto de V x V. De este modo, para un vértice dado (Vi), In (Vi) sería el conjunto de vértices que apuntan hacia el (predecesores), y Out (Vi) el conjunto de vértices que salen de Vi (sucesores). Por tanto el resultado del vértice Vi se define a través del algoritmo siguiente:

$$S(V_i) = (1-d) + d * \sum_{j \in In(V_i)} \frac{1}{|Out(V_j)|} S(V_j)$$

  Donde d es un factor de amortiguación que va del 0 al 1 y que juega un rol importante en integrar en el modelo la probabilidad de saltar de un vértice dado a otro aleatoriamente dentro del grafo. Generalmente, el valor de d suele ser 0.85 (Brin y Page, 1998) y es el que varios trabajos están empleando a la hora de implementar este algoritmo **[Mihalcea04]** Una vez ejecutado este algoritmo, cada vértice obtiene con una puntuación que representa su importancia dentro de la gráfica.

Paso 3: Una vez que el resultado final es obtenido para cada vértice, los vértices se ordenan en orden inverso a su puntuación y aquellos que ocupen los puestos superiores en la clasificación son post-procesados. Generalmente, T suele ser un conjunto de 5 a 20 palabras clave, en cambio TextRank tiene en cuenta el tamaño del texto a la hora de extraer las palabras clave.

Paso 4: Durante el post-procesamiento todas las unidades léxicas que han sido seleccionadas como palabras claves son marcadas en el texto y palabras adyacentes son consideradas como una sola palabra clave compuesta por varias palabras.

2.3 Herramientas

2.3.1 MySQL

Sistema de administración de Bases de Datos, en este caso utilizado debido a su gran portabilidad ya que es soportado en múltiples plataformas, cuyo desarrollo data de 1980 por programadores de IBM con el fin de contar con un código de programación que permitiese crear múltiples bases de datos para diferentes empresas y organizaciones.

MySQL implemente además múltiples motores de almacenamiento como InnoB, MyISAM, Merge, BDB,… posibilitando al usuario escoger la más adecuada tabla de la base de datos. También permite agrupar transacciones con el fin de incrementar el número de transacciones por segundo.

Se puede decir que con el paso del tiempo los desarrolladores cada vez utilizan más este sistema de administración de base de datos

2.3.2 PHP

PHP (acrónimo recursivo de PHP: Hypertext Preprocessor) es un lenguaje de programación originalmente diseñado para el desarrollo web de contenido dinámico con acceso a información almacenada en una base de datos.

Se distingue de lenguajes como JavaScript en que el código de la aplicación es ejecutado en el servidor, que genera un HTML y lo envía al cliente siendo el código fuente invisible para este y para el navegador.

Existen esencialmente tres campos donde se usan Scripts de PHP:

- Scripts del lado del Servidor: es el campo más usado en el que es necesario la combinación de tres elementos para su funcionamiento. El analizador de PHP, un servidor web y un navegador web
- Scripts desde la línea de comandos: sin la necesidad de un navegador web ni de ningún servidor se pueden ejecutar Scripts a través de línea de comandos. Este tipo de Script es ideal para scripts ejecutados regularmente.
- Escribir aplicaciones desde escritorio: aunque no es un lenguaje apropiado para este cometido también se puede utilizar para tal fin.

Es la tecnología utilizada para el desarrollo del prototipo recopilador de Tuits.

2.3.3 Java

Java es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems

Existen tres plataformas diferentes:

- **JSE (Java Standard Edition):** orientada al desarrollo de aplicaciones web o de escritorio.
- **JEE (Java Enterprise Edition):** orientada a entornos distribuidos empresariales o de Internet. Permite la creación de aplicaciones cliente/servidor, desarrollo de servicios web, persistencia de objetos…
- **Java ME (Java Micro Edition):** Versión de la plataforma Java para dispositivos con recursos y capacidades limitadas.

Ha sido utilizada la plataforma JEE para llevar a cabo el desarrollo del prototipo de generación de resúmenes

Capítulo 3 Planificación

Con el fin de que se pueda ver el cambio de planificación que se ha tenido que tomar por los contratiempos imprevistos, se incluye en este apartado el diagrama de la planificación inicial.

3.1 Planificación Inicial

A continuación se incluye la lista de tareas que se habían planificado inicialmente a muy alto nivel del proyecto:

	Nombre de tarea	Duración	Comienzo	Fin
1	**OBTENCIÓN DE RESÚMENES AUTOMÁTICOS PARA TEMAS DE AC**	**211 días?**	**mié 11/09/13**	**mié 02/07/14**
2	**Recopilación de datos (Fase I)**	**103 días?**	**mié 11/09/13**	**vie 31/01/14**
3	Estudio de tecnologías a utilizar en Fase I	14 días?	mié 11/09/13	lun 30/09/13
4	Especificación de requisitos	10 días	mar 01/10/13	lun 14/10/13
5	Análisis y diseño de la aplicación de descarga	20 días	mar 15/10/13	lun 11/11/13
6	Desarrollo de prototipo de descarga de Trending topics, tweets	48 días	mar 12/11/13	jue 16/01/14
7	Documentacion Fase I	11 días	vie 17/01/14	vie 31/01/14
8	**Analisis de datos y formulación de Hipótesis (Fase II)**	**40 días**	**lun 03/02/14**	**vie 28/03/14**
9	Elección de estrategia para formar patrones de descripción	15 días	lun 03/02/14	vie 21/02/14
10	Formulación de Hipótesis	15 días	lun 24/02/14	vie 14/03/14
11	Documentación Fase II	10 días	lun 17/03/14	vie 28/03/14
12	**Generación de resúmenes(Fase III)**	**68 días**	**lun 31/03/14**	**mié 02/07/14**
13	Especificación de requisitos	10 días	lun 31/03/14	vie 11/04/14
14	Analisis/Diseño	20 días	lun 14/04/14	vie 09/05/14
15	Desarrollo de Prototipo	30 días	lun 12/05/14	vie 20/06/14
16	Pruebas	5 días	lun 23/06/14	vie 27/06/14
17	Fin Documentacion	38 días	lun 12/05/14	mié 02/07/14

Figura 3.1 Tares planificación Inicial

En la siguiente Figura 3.2 se muestra el diagrama de Gantt con la planificación Inicial

Id	Nombre de tarea	Duración	Comienzo	Fin
1	**OBTENCIÓN DE RESÚMENES AUTOMÁTICOS PARA TEMAS DE ACTU**	**211 días?**	**mié 11/09/13**	**mié 02/07/14**
2	**Recopilación de datos (Fase I)**	**103 días?**	**mié 11/09/13**	**vie 31/01/14**
3	Estudio de tecnologías a utilizar en Fase I	14 días?	mié 11/09/13	lun 30/09/13
4	Especificación de requisitos	10 días	mar 01/10/13	lun 14/10/13
5	Análisis y diseño de la aplicación de descarga	20 días	mar 15/10/13	lun 11/11/13
6	Desarrollo de prototipo de descarga de Trending topics, tweets y des	48 días	mar 12/11/13	jue 16/01/14
7	Documentacion Fase I	11 días	vie 17/01/14	vie 31/01/14
8	**Analisis de datos y formulación de Hipótesis (Fase II)**	**40 días**	**lun 03/02/14**	**vie 28/03/14**
9	Elección de estrategia para formar patrones de descripción	15 días	lun 03/02/14	vie 21/02/14
10	Formulación de Hipótesis	15 días	lun 24/02/14	vie 14/03/14
11	Documentación Fase II	10 días	lun 17/03/14	vie 28/03/14
12	**Generación de resúmenes(Fase III)**	**68 días**	**lun 31/03/14**	**mié 02/07/14**
13	Especificación de requisitos	10 días	lun 31/03/14	vie 11/04/14
14	Analisis/Diseño	20 días	lun 14/04/14	vie 09/05/14
15	Desarrollo de Prototipo	30 días	lun 12/05/14	vie 20/06/14
16	Pruebas	5 días	lun 23/06/14	vie 27/06/14
17	Fin Documentacion	38 días	lun 12/05/14	mié 02/07/14

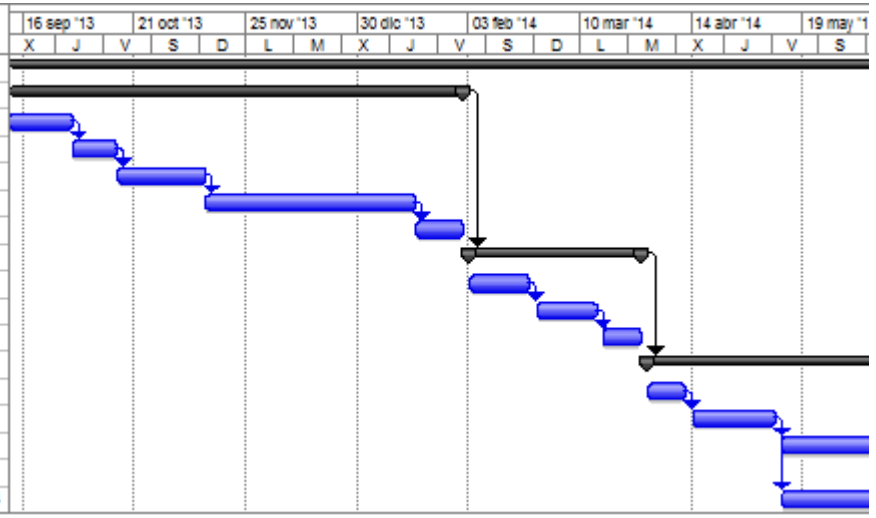


Figura 3.2 Diagrama de Gantt Inicial

3.2 Planificación Real

En la siguiente imagen se puede ver la división de tareas por la que ha pasado la elaboración de este proyecto tras proceder a su re-planificación.

	Nombre de tarea	Duración	Comienzo	Fin
1	**OBTENCIÓN DE RESÚMENES AUTOMÁTICOS PARA TEMAS I**	**227 días?**	**mié 11/09/13**	**vie 18/07/14**
2	Reunión director TFM 1	1 día	mié 11/09/13	mié 11/09/13
3	Planificación Inicial	10 días	jue 12/09/13	mié 25/09/13
4	Investigación funcionamiento whatthetrend	14 días	jue 26/09/13	mar 15/10/13
5	Reunion director TFM 2	2,25 días	mié 16/10/13	vie 18/10/13
6	Investigación funcionamiento API Twitter	14 días	vie 18/10/13	mié 06/11/13
7	Investigación funcionamiento whatthetrend	14 días	mié 06/11/13	mar 26/11/13
8	Instalación de aplicaciones en máquina virtual	2 días?	mar 26/11/13	jue 28/11/13
9	**Recopilación de datos (Fase I)**	**165,75 días?**	**jue 28/11/13**	**lun 14/07/14**
10	Diseño BBDD	6 días	jue 28/11/13	vie 06/12/13
11	Reunión 3 y Replanificación Proyecto	1 día	vie 06/12/13	lun 09/12/13
12	Creación BBDD para almacenar listado de Tweets	10,5 días?	lun 09/12/13	vie 20/12/13
13	Desarrollar prototipo de descarga de información	27 días	vie 20/12/13	mar 28/01/14
14	Reunion director TFM 4	1 día	lun 16/12/13	lun 16/12/13
15	Finalización desarrollo prototipo descarga	8,91 días	mié 18/12/13	lun 30/12/13
16	Descarga de Información	140 días?	jue 02/01/14	lun 14/07/14
17	Copia de seguridad de información	140 días	jue 02/01/14	lun 14/07/14
18	Liberación de espacio en disco	140 días?	jue 02/01/14	lun 14/07/14
19	Control de funcionamiento de aplicación de descarga y c	140 días?	jue 02/01/14	lun 14/07/14
20	**Estudio y análisis de algoritmos resumen (Fase II)**	**51 días?**	**mié 15/01/14**	**mar 25/03/14**
21	Estudio y documentación del arte previo en técnicas d	30 días	mié 15/01/14	lun 24/02/14
22	Elección algoritmos de resumen	1 día?	mar 25/03/14	mar 25/03/14
23	Desarrollar Prototipo Generador de resumenes (Fase III)	30 días	jue 24/04/14	mié 04/06/14
24	Reunión Director TFM 5	1 día?	jue 05/06/14	jue 05/06/14
25	Elección Tweets para evaluación	1 día?	sáb 07/06/14	sáb 07/06/14
26	**Evaluación del prototipo (Fase IV)**	**14,5 días?**	**lun 16/06/14**	**vie 04/07/14**
27	Elaboración de intrucciones y fichero de primera evaluac	0,5 días?	lun 16/06/14	lun 16/06/14
28	Evaluación I de resúmenes por curators	6 días?	lun 16/06/14	mar 24/06/14
29	Procesamiento de datos de evaluación I	4 días?	mar 24/06/14	lun 30/06/14
30	Elaboración de intrucciones y fichero de segunda evalua	1 día?	lun 30/06/14	mar 01/07/14
31	Evaluacion II de resúmenes por curators	1 día	mar 01/07/14	mié 02/07/14
32	Procesamiento de datos de evaluacion II	1 día?	mié 02/07/14	jue 03/07/14
33	Análisis y extracción de conclusiones	1 día?	jue 03/07/14	vie 04/07/14
34	**Finalización de documentación y presentación (Fase**	**30 días**	**lun 09/06/14**	**vie 18/07/14**
34	**Finalización de documentación y presentación (Fase**	**30 días**	**lun 09/06/14**	lun 30/06/14
35	Elaboración borrador documentación	16 días	lun 09/06/14	lun 07/07/14
36	Reunión director TFM 6	1 día	vie 04/07/14	lun 07/07/14
37	Revisión y finalización documentación	5 días	mar 01/07/14	lun 07/07/14
38	Entregar Documentación	1 día	lun 07/07/14	mar 15/07/14
39	Hacer transparencias para la presentación del proyecto	6 días	mar 08/07/14	mié 16/07/14
40	Ensayar presentación	1 día	mié 16/07/14	vie 18/07/14
41	Presentar proyecto	1 día	vie 18/07/14	vie 18/07/14

Figura 3.3 Listado de tareas del Trabajo Fin de Máster

Id	Nombre de tarea	Duración
1	**OBTENCIÓN DE RESÚMENES AUTOMÁTICOS PARA TEMAS DE.**	**227 días?**
2	Reunión director TFM 1	1 día
3	Planificación inicial	10 días
4	Búsqueda alternativas a Whatthetrend	14 días
5	Reunion director TFM 2	1 día
6	Investigación funcionamiento API Twitter	14 días
7	Investigación funcionamiento whatthetrend	14 días
8	Instalación de aplicaciones en máquina virtual	2 días?
9	**Recopilación de datos (Fase I)**	**167 días?**
10	Diseño BBDD	6 días
11	Reunión 3 y Replanificación Proyecto	1 día
12	Creación BBDD para almacenar listado de Tweets	10,5 días?
13	Desarrollar prototipo de descarga de información	27 días
14	Reunion director TFM 4	1 día
15	Finalización desarrollo prototipo descarga	8,91 días
16	Descarga de información	140 días?
17	Copia de seguridad de información	140 días
18	Liberación de espacio en disco	140 días?
19	Control de funcionamiento de aplicación de descarga y correo	140 días?
20	**Estudio y análisis de algoritmos resumen (Fase II)**	**51 días?**
21	Estudio y documentación del arte previo en técnicas de re	30 días
22	Elección algoritmos de resumen	1 día?
23	Desarrollar Prototipo Generador de resumenes (Fase III)	30 días
24	Reunión Director TFM 5	1 día?
25	Elección Tweets para evaluación	1 día?
26	**Evaluación del prototipo (Fase IV)**	**14,5 días?**
27	Elaboración de intrucciones y fichero de primera evaluación	0,5 días?
28	Evaluación I de resúmenes por curators	6 días?
29	Procesamiento de datos de evaluación I	4 días?
30	Elaboración de intrucciones y fichero de segunda evaluación	1 día?
31	Evaluacion II de resúmenes por curators	1 día
32	Procesamiento de datos de evaluacion II	1 día?
33	Análisis y extracción de conclusiones	1 día?
34	**Finalización de documentación y presentación (Fase V)**	**30 días**
35	Elaboración borrador documentación	16 días
36	Reunión director TFM 6	1 día
37	Revisión y finalización documentación	5 días
38	Entregar Documentación	1 día
39	Hacer transparencias para la presentación del proyecto	6 días
40	Ensayar presentación	1 día
41	Presentar proyecto	1 día

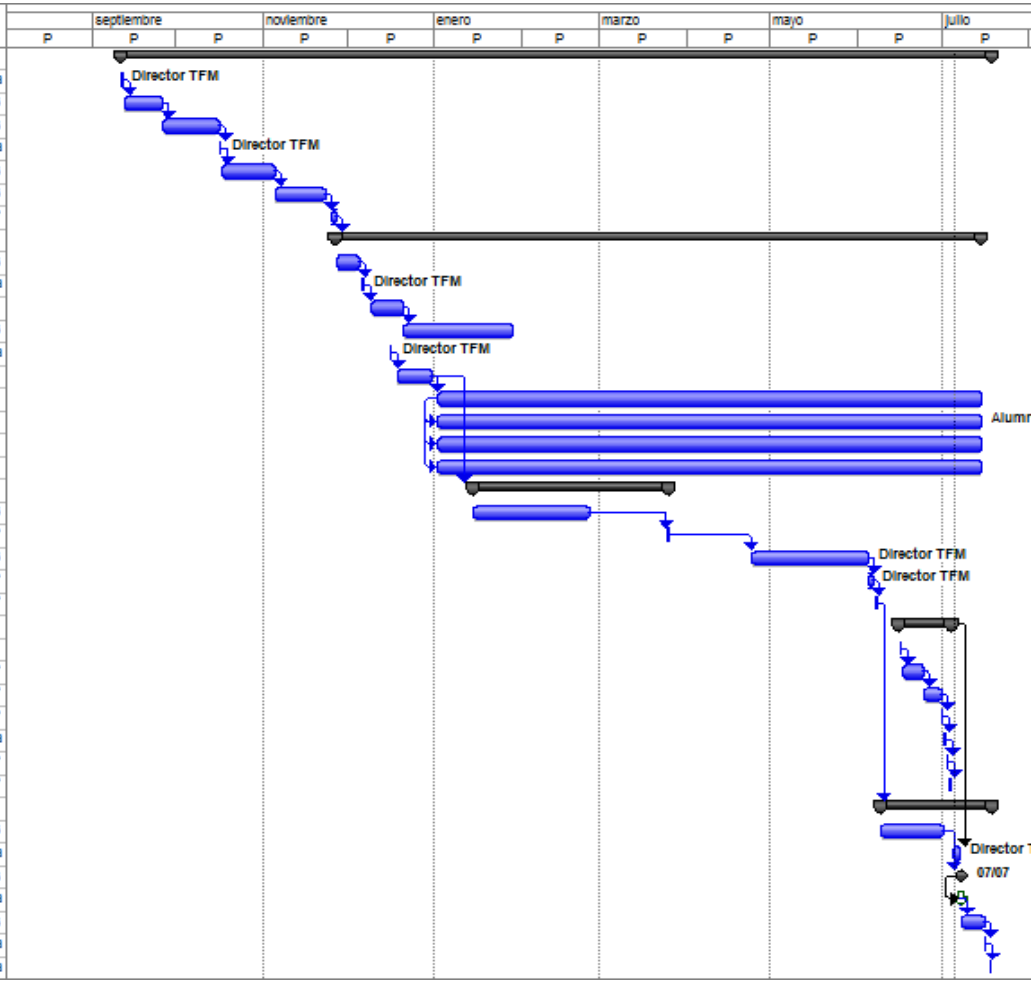


Figura 4.1.Diagrama de Gantt

Capítulo 4 Desarrollo de prototipos

4.1 Introducción

En primer lugar hay que destacar que se ha trabajado

En este capítulo se explicarán los aspectos más representativos de los prototipos implementados

4.2 Herramientas y Programas Usados para el Desarrollo

A continuación se describirán las herramientas utilizadas durante el desarrollo de los prototipos

4.2.1 AppServ

Herramienta de código libre para Windows con Apache, MySQL, PHP y otras funcionalidades que permite la ejecución de un servidor Web completo.

Esta herramienta se ha utilizado para el prototipo de descarga de información de Twitter en su versión 2.5.10.

En la siguiente Figura pueden verse las versiones de los paquetes que contiene la versión utilizada en el prototipo.

Figura 4.1 Versiones de los paquetes de la aplicación

4.2.2 Eclipse

Entorno de desarrollo integrado open source, que permite trabajar con gran variedad de ficheros, véase Java, C, C++, HTML, JSP, CSS,… Permite el uso de herramientas como ANT y JUnit, la comparación de archivos, etc.…
Es una herramienta muy completa, es por ello que ha sido el programa fundamental para el desarrollo del prototipo de generación de resúmenes. Una de sus grandes ventajas es que su funcionalidad es fácilmente extensible pudiendo ampliar sus características mediante el empleo de plug-ins.

4.2.3 MySQL

Sistema de administración de Bases de Datos, que ofrece una gran portabilidad ya que es soportado en múltiples plataformas, en este sistema se almacenará la base de datos del proyecto.
Se ha utilizado la versión 6.0.10-alpha-community.

4.2.4 Notepad++

Es un editor de código gratuito utilizado en su versión 6.1.2. Con este programa se han desarrollado principalmente los scripts para el prototipo de la descarga de datos.

4.2.5 MySQL Worbench

Programa que se ha utilizado para administrar y desarrollar la base de datos. Se ha utilizado debido a que permite un desarrollo y administración de modo interactivo. Facilita la gestión de BBDD, de tablas, triguers, funciones y procedimientos.
Se ha utilizado la versión 5.2.

4.3 Descripción de los prototipos

4.3.1 Prototipo de descarga de datos

Este prototipo se ha desarrollado en PHP y es el encargado de la descarga de datos de Twitter mediante la utilización de su API. El proceso que se sigue es el siguiente:

- Cada 15 minutos se consulta la lista de Trending Topics de Twiter para España y EEUU.
- Se almacenan en una Base de datos MySQL estos trending topic
- Posteriormente se dejan descargando en un fichero todos los tuits posibles para esos trending topics usando la Streaming API de Twitter.

- Se procesan todos esos tuits para almacenar cada uno en un fichero cuyo nombre será el trending topic que le corresponde, a través de la utilización de un script que los introduce en una carpeta con la fecha en que fueron descargados.
- Este proceso se registra en la BBDD indicando la ruta del fichero donde se guardan los archivos descargados

Para el funcionamiento de este prototipo, previamente se ha tenido que montar una máquina virtual en Windows para desplegarlo en ella y que funcionase ininterrumpidamente.

Hay que destacar que en el proceso de descarga de datos se tuvo que hacer un proceso diario de copia de seguridad de los datos de la máquina virtual, así como la liberación de espacio en disco.

Se utilizaron las siguientes tablas para la BBDD.

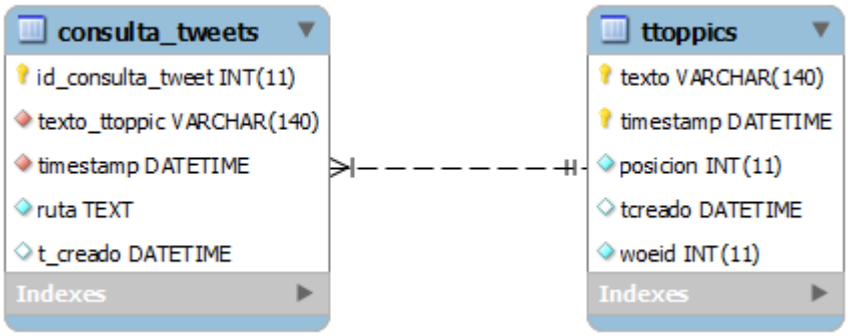


Figura 4.2 Tablas de la BBDD del recopilador

Los Tuits se almacenaron en archivos debido a las limitaciones con la máquina virtual a la hora de realizar copias de seguridad.

4.3.2 Prototipo de generación de resúmenes

Ente prototipo se ha desarrollado utilizando el lenguaje de programación Java y en él se implementan cuatro algoritmos que serán los que generen los resúmenes extractivos para su posterior evaluación.

Se implementaron los algoritmos cuyo funcionamiento fue explicado en el apartado de aspectos teóricos del capítulo 2, fueron los siguientes:

- Random
- Hybrid TF-IDF
- SumBasic
- TextRank

Esta aplicación contiene un Filtro que procesa los ficheros descargados con los JSON de cada tweet y los filtra de modo que almacena en un fichero el id y el texto de ese trending topic eliminando caracteres que puedan complicar la generación de resúmenes.

En las todos los algoritmos salvo en el Random se eliminan las palabras vacías a la hora de puntuar a los tweets candidatos a ser resumen, éstas se encuentran en la carpeta útil del proyecto, en este caso se guardaron las palabras vacías para el inglés y el castellano.

Pese a ser un simple prototipo se incluyó el patrón de diseño Fachada para los algoritmos con el fin de evitar dependencias en caso de que se reutilizase este prototipo para seguir con la realización de pruebas mejorando el prototipo.

Para la obtención de resúmenes se creó una pequeña interfaz para el programador para que pudiese interactuar con las funcionalidades desarrolladas.

A continuación se muestran los pasos que se siguen para generar resúmenes a través de la interfaz:

```
**********MENÚ**********
1.-Elegir TToppic a resumir
2.-Prueba individual de algoritmos
3.-Salir
************************
```

Figura 4.3 Pantalla Inicial del prototipo de generador de Tuits

En la Figura 4.3 se puede observar el menú principal para la aplicación de descarga de Tuits donde tras pulsar en la primera opción se mostrará un submenú (ver Figura 4.4) que pedirá que se indique el trending topic que se desea resumir con el fin de resumir ese tema con todos los algoritmos generando un fichero resumen del trending topic para cada algoritmo (ver Figura 4.5).

La opción 2 del menú principal se utilizó para hacer pruebas individuales del funcionamiento de los algoritmos.

```
3.-Salir
************************
1
**********TEMAS DISPONIBLES**********
1.-#abdicacion
2.-#APorLaTerceraRepublica
3.-#ElReyAbdica
4.-#FelipeVI
5.-#IIIRepublica
6.-#TATGranada14
7.-#TEDxGijon
8.-#VamosRafa
9.-Brasil-Panamá
10.-Selectividad
11.-Felipe VI
12.-Volver al menú principal
************************
```

Figura 4.4 Submenú con los temas disponibles a resumir

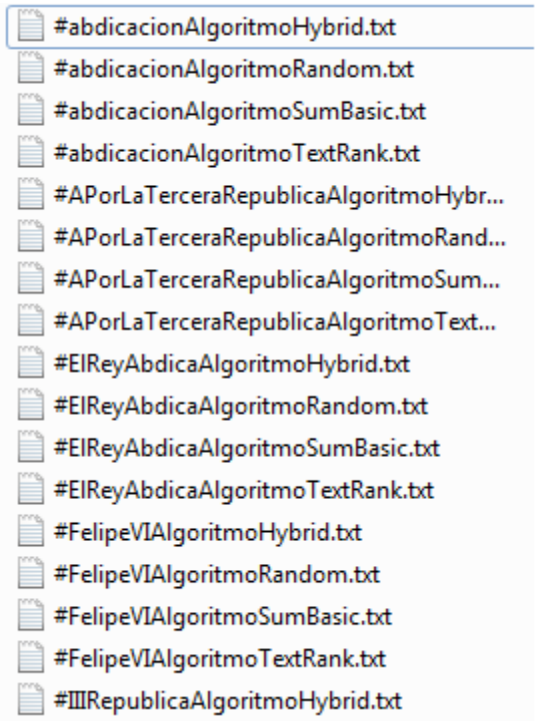


Figura 4.5 Imagen de los archivos generados por el generador de resúmenes

Para su correcto funcionamiento deben almacenarse los trending topic a resumir en la carpeta Tweets guardándolos en su respectiva carpeta con el nombre del trending topic al que pertenecen y añadiéndolos en la clase principal para que sean procesados por los algoritmos.

4.4 Problemas encontrados

A lo largo del proyecto han surgido gran cantidad de imprevistos que han retrasado bastante los tiempos en el desarrollo de las tareas que se tenían planificadas, obligando a reorientar el proyecto y a tener que dejar de hacer partes del proyecto que hubiesen completado de forma razonable el estudio del enfoque previsto.

Algunos de estos problemas han sido:

- Continuos cambios en la API de Twitter que tuvieron como resultado tener que modificar bastantes veces el recopilador de Tuits.
- Inicialmente se tenía planteado comparar los resúmenes generados por el prototipo con las descripciones que producían los usuarios de Whatthetrend, donde surgió el imprevisto de que dicha página web quedó en desuso durante el desarrollo del proyecto.
- El desconocimiento del propio alumno sobre la temática provocó una subestimación del tiempo de desarrollo de algunas tareas que hicieron alargar excesivamente su finalización.

-

4.5 Descripción General de las clases

A Continuación se mostrará la descripción de las clases utilizadas en el prototipo de generación de resúmenes:

4.5.1 Clase AlgoritmoHybrid

Nombre	Tipo	Descripción	Hereda de…
AlgoritmoHybrid	Publico	Procesa Tuits y genera resúmenes aplicando el algoritmo Hybrid	
Responsabilidades			
Número	**Descripción**		
1	Obtención de resúmenes automáticos a partir de un tema en particular mediante el uso del algoritmo Hybrid TF.IDF		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
Público		AlgoritmoHybrid	-
Público		getResumen	String ruta
Privado	void	calcularPalabras	String texto
Privado	void	aparicionesPalabra	String palabra
Privado	void	aparicionesSentencia	List<String> lista
Privado	void	procesarFichero	String ruta
Privado	void	calcularProbabilidades	-
Privado	void	calcularProbabilidadesIDF	-
Privado	double	pesoSentencia	String tweet
Privado	double	pesoPalabra	String palabra
Privado	double	frecuenciaPalabra	String palabra
Privado	double	frecuenciaIDF	String palabra
Privado	void	tweetsTop	Tweet tuit
Privado	void	pesarTweets	String ruta
Público	List <Tweet>	getResumenes	-
Público	void	setResumenes	List<Tweet> resumenes
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
Private		int	numPalabras
Private		List	numPosts
Private		Map <String,Integer>	apariciones
Private		Map<String, Integer>	aparicionesSentencia
Private		Map <String, Double>	Frecuencias
Private		Map <String, Double>	frecuenciasIDF
Private		List <Tweet>	aesúmenes
Private		String	idioma
Private		Stopword	stopword
Observaciones			

-

4.5.2 Clase AlgoritmoRandom

Nombre	**Tipo**	**Descripción**	**Hereda de…**
AlgoritmoRandom	Publico	Procesa Tuits y genera resúmenes aplicando el algoritmo Random	
Responsabilidades			
Número	**Descripción**		
1	Obtención de resúmenes automáticos a partir de un tema en particular mediante el uso del algoritmo Random		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
Público	-	AlgoritmoRandom	
	Void	ejecutarAlgoritmo	String ruta
Público	String	getResumen	String ruta
Público	String	getLinea	Int numLinea, String ruta
Privado	Int	numLineas	String ruta
Público	List<Tweet>	getResumenes	-
Público	Void	setResumenes	List <Tweet> resumenes
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
Private		List<Tweet>	resumenes
Observaciones			
-			

4.5.3 Clase AlgoritmoSumBasic

Nombre	**Tipo**	**Descripción**	**Hereda de…**
AlgoritmoSumBasic	Publico	Procesa Tuits y genera resúmenes aplicando el algoritmo SumBasic	
Responsabilidades			
Número	**Descripción**		
1	Obtención de resúmenes automáticos a partir de un tema en particular mediante el uso del algoritmo SumBasic		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
Público		AlgoritmoSumBasic	
Público	String	getResumen	String ruta
Privado	Void	procesarFIchero	String ruta
Privado	Void	calcularPalabras	String texto
Privado	Void	calcularProbabilidades	-
Privado	Tweet	tweetsTop	Tweet tuit
Privado	Void	pesarTweets	String ruta
Público	List <Tweet>	getResumenes	-
Público	void	setResumenes	List <Tweet> resumenes
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
Private		int	numPalabras
Private		Map <String, Integer>	apariciones
Private		Map<String, Double>	frecuencias
Private		List <Tweet>	resumenes
Private		String	idioma
Private		Stopword	stopword
Observaciones			
-			

4.5.4 Clase AlgoritmoTextRankKeyword

Nombre	**Tipo**	**Descripción**	**Hereda de…**
AlgoritmoTextRankKeywords	Publico	Procesa Tuits y genera resúmenes aplicando el algoritmo TextRank basado en keywords	
Responsabilidades			
Número	**Descripción**		
1	Obtención de resúmenes automáticos a partir de un tema en particular mediante el uso del algoritmo TextRank basado en keywords		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
Público	-	AlgoritmoTextRankKeywords	-
Público	void	ejecutarAlgoritmo	String ruta
Privado	void	cargarTweets	String ruta
Privado	String[]	palabrasSinStopWords	String [] palabras
Privado	void	procesarFrase	String frase
Privado	void	calculoPuntuacion	-
Privado	double	calculoSumatorioConexiones	Set<Nodo> nodos
Privado	void	tweetsTop	Tweet tuit
Privado	void	pesarTweets	String ruta
Público	List<Tweet>	getResumenes	-
Público	void	setResumenes	ListTweet<Tweet> resumenes
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
Private		GrafoRank	grafo
Private		String	idioma
Private		StopWord	stopword
Private	static final	float	DUMPING
Private	static final	float	THRESHOLD
Private		List<Tweet>	resumenes
Observaciones			
-			

4.5.5 Clase Tweet

Nombre	**Tipo**	**Descripción**	**Hereda de…**
Tweet	Publico	Contendrá la información de los Tuits	
Responsabilidades			
Número	**Descripción**		
1	Contener la información de los Tuits		
2	Efectuar el pesado de los mismos		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
Público	-	Tweet	String linea
Público	void	pesarTweet	Map <String,Double> frecuencias
Público	void	setID	String id
Público	String	getID	-
Público	void	setPesp	float peso
Público	float	getPeso	-
Público	void	setTweet	String tweet
Público	String	getTweet	-
Privado	void	calcularNumPalabras	-
Público	int	getNumPalabras	-
Público	int	getNumPalabras2	-
Público	int	compareTo	Tweet tuit
Público	int	hashCode	-
Público	boolean	equals	Object obj
Público	String	toString	-
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
Private		int	numPalabras
Private		int	numPalabras2
Private		String	tweet
Private		Float	peso
Private		String	id
Private		Map<String, Integer>	aparaciones
Observaciones			
-			

4.5.6 Clase StopWord

Nombre	Tipo	Descripción	Hereda de…
StopWord	Publico	Clase que contiene las StopWords para cada idioma	
Responsabilidades			
Número	**Descripción**		
1	Contiene las StopWords necesarias para aplicar los algoritmos		
2	Comprobar si una palabra es StopWord		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
Público		StopWord	String idioma
Público	boolean	isStopWord	String palabra
Privado	void	cargarStop	-
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
Private		String	idioma
Private		String []	stopwords
Private		Set <String>	stopWordSet
Observaciones			
-			

4.5.7 Clase GrafoRank

Nombre	Tipo	Descripción	Hereda de…
GrafoRank	Publico	Será el grafo para el algoritmo TextRankKeywords	
Responsabilidades			
Número	**Descripción**		
1	Ser el grafo utilizado por el algoritmo TextRankKeywords		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
Público		GrafoRank	-
Público	Map <String, Nodo>	getGrafo	
Público	void	anadirNodo	Nodo n
Público	Nodo	getNodo	String key
Público	void	ordenar	-
Público	void	modificarNodo	Nodo n
Público	void	borrarNodo	Nodo n
Público	String	toString	-
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
Private		Map<String,Nodo>	grafo
Observaciones			
-			

4.5.8 Clase Nodo

Nombre	**Tipo**	**Descripción**	**Hereda de…**
Nodo	Publico	Constituirá cada elemento del GrafoRank	
Responsabilidades			
Número	**Descripción**		
1	Constituir el elemento necesitado para construir un grafo		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
Público		Nodo	String texto
Público	void	conectarNodo	Nodo nuevo
Público	void	combinarConexiones	Nodo nuevo
Público	int	getNumConexiones	-
Público	void	desconectarNodo	Nodo n
Público	double	getPuntuacionAnterior	-
Público	void	setPuntuacionAnterior	double puntuacionAnterior
Público	Set <Nodo>	getConexiones	-
Público	void	setConexiones	Set<Nodo> conexiones
Público	void	setTexto	String texto
Público	String	getTexto	-
Público	void	setPuntuacion	double puntuacion
Público	double	getPuntuacion	-
Público	String	toString	-
Público	int	hashCode	-
Público	boolean	equals	Object obj
Público	Int	compareTo	Nodo n
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
Private		String	texto
Private		doublé	puntuacion
Private		doublé	puntuacionAnterior
Private		Set <Nodo>	conexiones
Observaciones			
-			

4.5.9 Clase Filtro Fichero

Nombre	**Tipo**	**Descripción**	**Hereda de…**
FiltroFichero	Publico	Será el encargado de filtrar el fichero de entrada eliminando caracteres que compliquen la ejecución de cada algoritmo	
Responsabilidades			
Número	**Descripción**		
1	Será el encargado de filtrar todos aquellos caracteres que dificulten el posterior procesamiento del mismo por los algoritmos de generación de resúmenes		
2	Generación de ficheros filtrados para su uso posterior por los algoritmos		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
Público	-	FiltroFichero	String ruta, String idioma, boolean filtro
Privado	void	verFicheros	String ruta
Privado	String[]	parsearLinea	String linea
Privado	String	fraseFiltrada	String texto
Privado	void	procesarFichero	String ruta
Privado	void	escribirLinea	String linea
Público	void	getRuta	-
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
Private		String	ruta
		String	rutafinal
		String	idioma
Observaciones			
-			

4.5.10 Clase Principal

Nombre	Tipo	Descripción	Hereda de…
Principal	Publico	Será la clase que servirá para interactuar entre el programador y la aplicación	
Responsabilidades			
Número	**Descripción**		
1	Interfaz del programador		
Métodos			
Acceso | Modo	**Tipo de Retorno**	**Nombre**	**Parámetros y tipos**
Privado	void	mostrarMenuTematica	-
Publico	void	procesarMenuTematica	-
Privado	void	mostrarMenu	-
Privado	void	procesarMenu	Int opcion,String rutaFicheroFiltrado
Privado	void	mostrarTemas	-
Privado	void	generarResumenes	String rutaFicheroFiltrado, String topic
Privado	void	procesarMenuTemas	Int opcion, char filtro
Privado	void	verFicheros	-
Privado	void	fileCopy	String sourceFile, String destination
Privado	void	guardarResumen	List <Tweet> lista,String ruta
Privado	void	algorimoRandom	String ruta, String topic
Privado	void	algoritmoHybrid	String ruta, String topic
Privado	void	algoritmoSumbasic	String ruta, String topic
Privado	void	algoritmoTextRankKeywords	String ruta, String topic
Público	void	main	String [] args
Atributos			
Acceso	**Modo**	**Tipo o Clase**	**Nombre**
Private		Map<String,Nodo>	grafo
Observaciones			
-			

Capítulo 5 Evaluación de resultados

En este capítulo se analizará la evaluación a la que han sido sometidos los resúmenes obtenidos de los algoritmos implementados por una serie evaluadores y los resultados obtenidos de dicho estudio.

5.1 Descripción de los experimentos

Con el fin de obtener unas conclusiones acerca del funcionamiento del prototipo se ha realizado una evaluación humana de los resúmenes generados, para que sirvan como punto de referencia sobre la calidad de éstos y así poder llevar a cabo en un futuro las posibles correcciones que posibiliten la obtención de resúmenes abstractivos con la ayuda de una aplicación de generación de resúmenes automáticos.

Para la evaluación se han seleccionado diez trending topic de diferentes temáticas generando aproximadamente 20 tuits resumen por cada uno y se ha elaborado una plantilla en Excel con éstos ordenados cronológicamente, para que los evaluadores, también llamados “curators” evaluasen distintos aspectos de los resúmenes habiéndoles proporcionado una serie de instrucciones que debían seguir.

A continuación se muestran las instrucciones que debían seguir los usuarios:

5.1.1 Instrucciones para los evaluadores

En primer lugar muchas gracias por cedernos parte de tu tiempo para este proyecto. Por favor, antes de comenzar tu tarea lee con atención las siguientes instrucciones:

- El objeto de esta actividad es verificar la utilidad de un sistema semiautomático para construir resúmenes de eventos de actualidad a partir de datos recogidos en Twitter.
- Aunque en ocasiones los hashtag empleados por los usuarios son suficientemente descriptivos, en la mayor parte de los casos resulta muy difícil para una persona ajena a la temática descubrirlo sólo a partir del tuit.
- Se ha de apuntar la hora de inicio y fin para la evaluación de cada tema.
- En la hoja de cálculo adjunta se ofrecen 10 temas que han sido "trending topic" en Twitter de manera reciente. Para cada uno de esos temas se ofrece una selección de tuits representativos (al menos 40 y nunca más de 100) ordenados de manera cronológica.
- Tu misión es leer esos tuits y construir lo que denominaremos "tuitresúmenes"; esto es, uno o más resúmenes de un máximo de 140 caracteres escritos por ti empleando tus propias palabras.
- Dichos tuitresúmenes deberían cumplir las siguientes condiciones (además del consiguiente límite de 140 caracteres): describir el tema cubierto por la selección de tuits incluyendo la mayor cantidad de información posible, ajustarse a la información

proporcionada por la selección de tuits (es decir, no puedes "inventar" nada ni aportar información que no esté presente en dicha selección), ser susceptibles de usarse en Twitter para "etiquetar" de manera abstracta el evento en cuestión.

- Para cada tema debes proporcionar al menos un tuitresumen y como máximo 5. El orden de los mismos será por relevancia; es decir, se da por supuesto que quien lea el tuitresumen N ya ha leído los tuitresúmenes de 1 a N-1 y, en consecuencia, tiene un contexto para comprender el contenido de ese tuitresumen. Dicho de otra forma, el primer tuitresumen proporcionaría una visión general y el resto ofrecerían visiones más concretas. En este sentido véase el ejemplo de #abdicación.
- Además, cada uno de los tuits de la selección debería etiquetarse del modo siguiente: se aplicará -1 si se considera que el tuit es irrelevante o poco útil para construir los tuitresúmenes; se aplicarán tantos números (del 1 al 5) como sean necesarios para indicar que la información contenida en ese tuit sirvió (total o parcialmente) para construir uno o más de los tuitresúmenes:

  -1 -> Resultó irrelevante para construir ningún tuitresumen

  1 -> Sirvió para construir el tuitresumen nº 1

  2 -> Sirvió para construir el tuitresumen nº2

  3 ->Sirvió para construir el tuitresumen nº3

  4 -> Sirvió para construir el tuitresumen nº4

  5 -> Sirvió para construir el tuitresumen nº5

  Véase el ejemplo de #abdicación. ¡Atención! No se pueden dejar tuits sin etiquetar.

- Por último, hay una sección de comentarios donde deberías escribir cualquier cuestión o duda que te haya surgido así como sugerencias relativas a la tarea o a la selección de tuits ofrecidos.

5.1.2 Hoja de cálculo de evaluación

En la Figura 5.1 se muestra la parte de la plantilla Excel que los evaluadores debían cumplimentar la celda de etiquetado adyacente a un tuit con -1 para indicar si este les había resultado irrelevante o si les había servido para construir su propio tuitresumen, en cuyo caso debían indicar el identificador correspondiente de su tuitresumen en dicha celda.

ID	ETIQUETADO	TWEET
473458061450170368	1	Nuevo rey nueva era Felipe VI
473458794455121920	1	Viva Felipe VI Rey de España
473458794455121920	1	Viva Felipe VI Rey de España
473458919831240704	-1	@SuperFalete Propongo un triunvirato Felipe VI Pablo Iglesias y María Teres
473458991134433280	3	@cnavarroahicart Felipe VI será coronado antes de un mes
473459077054726144	1	@mamenrmateos Felipe VI Rey de España
473459077054726144	1	@mamenrmateos Felipe VI Rey de España
473459309368848384	1	@Jero10 Felipe VI Rey de España
473459309368848384	1	@Jero10 Felipe VI Rey de España
473459359713087488	1	@DavidRMadrid8 Felipe VI REY DE ESPAÑA
473459359713087488	1	@DavidRMadrid8 Felipe VI REY DE ESPAÑA
473462267351429120	-1	Felipe VI Último rey de España
473462267351429120	-1	Felipe VI Último rey de España
473469680578215936	-1	'Felipe VI' suena un poco raro eso si la cuestión es meter contenido a los lil
473471026958176256	-1	Ya en exclusiva en @TheObjectivees Del profesor del Príncipe 'Felipe VI'
473471158965518336	-1	@itxudiaz Ya en exclusiva en @TheObjectivees Del profesor del Príncipe 'Fel
473471537954426880	1	Futuro Rey de España Felipe VI
473471537954426880	1	Futuro Rey de España Felipe VI
473474176875909120	2	@Rubenmtw Oialá la ceremonia de coronación de Felipe VI sea en un Burge

Figura 5.1 Vista de la plantilla de evaluación I

En la Figura 5.2 se muestra la otra parte de la plantilla Excel, donde los evaluadores debían escribir sus propios Tuitresúmenes con un número de caracteres no superior a 140, La hora de inicio y fin de la evaluación, así como los comentarios que creyesen convenientes.

Nº tuitresumen	Longitud	Tuitresumen
1	36	Rey de Espana, Juan Carlos I abdica
2	55	Felipe VI heredara la corona y pasara a se el nuevo rey
3	97	Gran parte de la poblacion reclama la proclamacion de Jordi Hurtado I como legitimo rey de Espana
4	0	
5		

Comentarios	Hora Inicio
Hay tweets repetidos	13:29
	Hora Fin
	13:38

Figura 5.2 Vista de la plantilla de evaluación II

Una vez recopiladas todas las evaluaciones se procedió a juntar todos los tuitresumen por tema de los evaluadores, generando otra plantilla Excel que permitiese puntuar cada tuitresumen con una puntuación del 1 al 10 y poder obtener el mejor resumen generado por un humano en cada tema.

En la siguiente Figura 5.3 se muestra la plantilla que debían rellenar para un tema

	Tweet	Puntuación
2	Abdicacion del rey Juan Carlos I y proclamacion porlas cortes de su hijo Felipe VI	
3	Abemus new rey!!	
4	El "rey del pueblo"	
5	El 18 de junio Felipe VI será coronado	
6	Felipe será nuevo rey de España y pasará a ser llamado Felipe VI tras su coronación.	
7	Felipe VI es el nuevo rey de España.	
8	Felipe VI nuevo Rey de España	
9	Felipe VI nuevo Rey de España	
10	Felipe VI nuevo rey de España	
11	Felipe VI nuevo Rey de España tras la abdicacion de su padre	
12	La gente celebrá la próxima coronación de un nuevo rey: Felipe VI	
13	Las cortes aprobaran la proclamación de Felipe VI el 18 de Junio	
14	Nombramiento y felicitaciones a Felipe VI como nuevo Rey de España una vez que ha abdicado el Rey	
15	Referendum Monarquia o Republica	
16	Rey de Espana abdica en su hijo, Felipe VI futuro rey de Espana.	
17	Se especula sobre las posibles fechas para la coronacion	
18	Usuarios tratan el tema con escepticismo, humor, la mayoria con poco respeto por la institucion (corona).	

Figura 5.3 Vista de la plantilla de clasificación de tuitresumenes

5.2 Resultados

A continuación se explicará, cuáles han sido los resultados obtenidos gracias a la evaluación.

Con respecto al porcentaje de relevancia de lo tuits resumen de cada algoritmo, las Figuras 5.4, 5.5, 5.6 y 5.7 representan en forma de gráfico para cada algoritmo la proporción de relevancia de sus resúmenes en cada trending topic

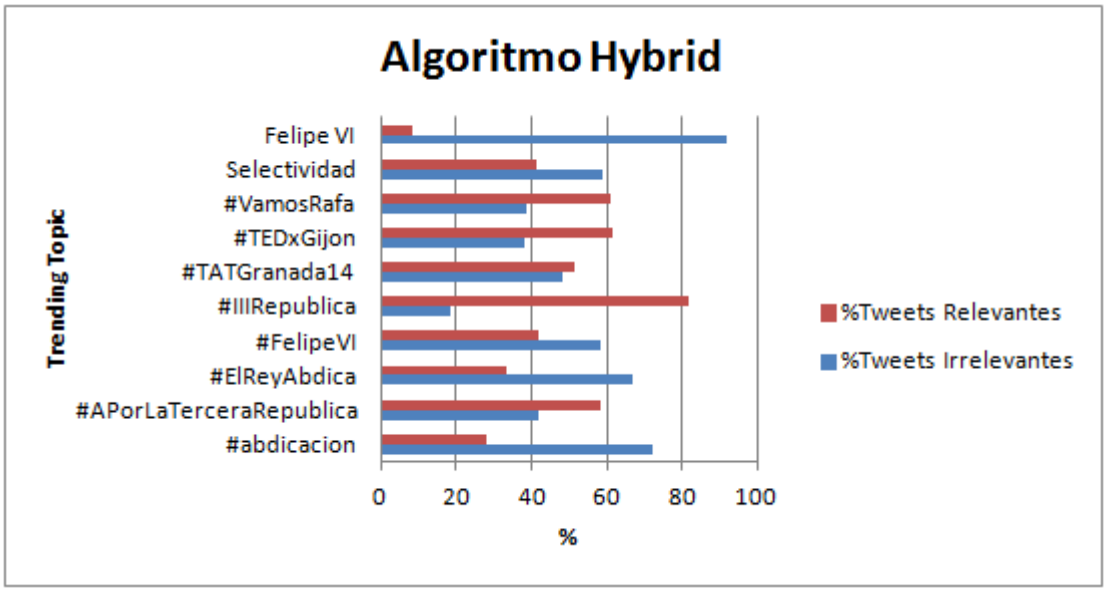


Figura 5.4 Gráfico con los porcentajes de relevancia del Algoritmo Hybrid TF-IDF

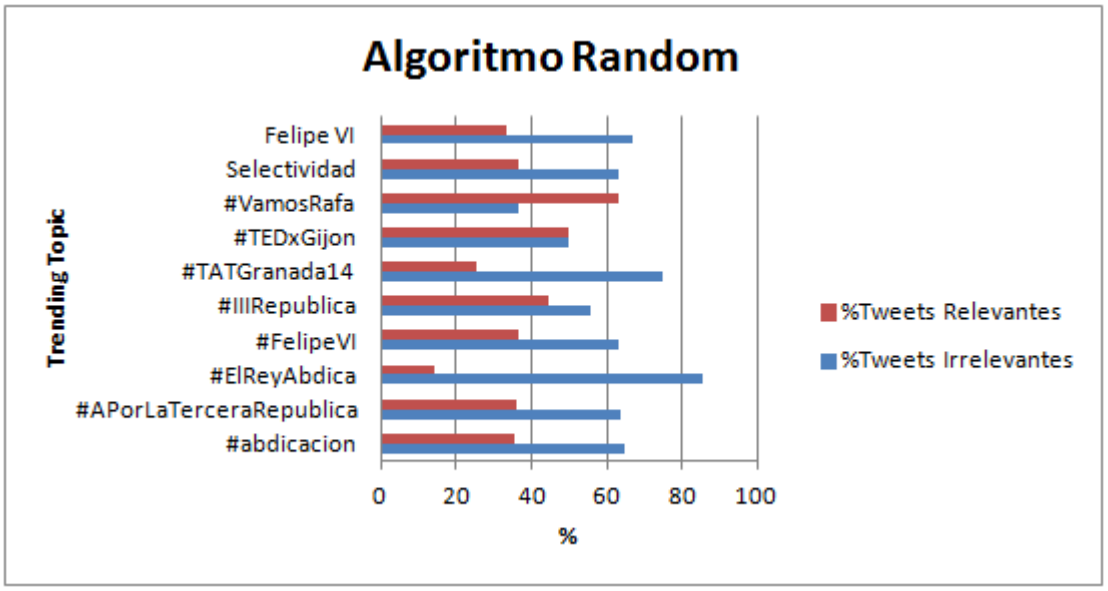


Figura 5.5 Gráfico con los porcentajes de relevancia del Algoritmo Random

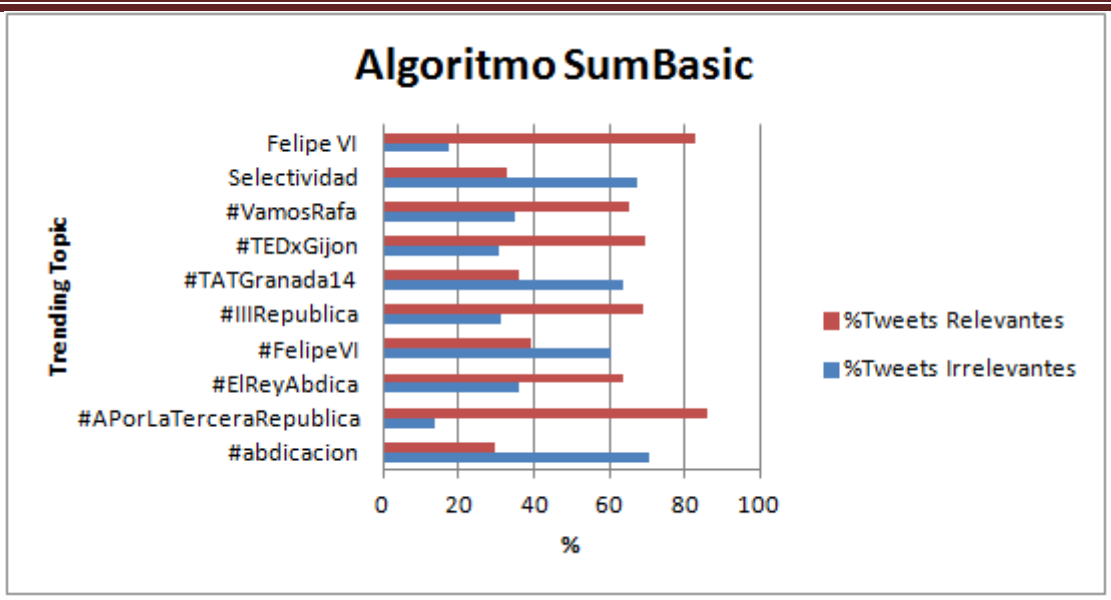


Figura 5.6 Gráfico con los porcentajes de relevancia del Algoritmo SumBasic

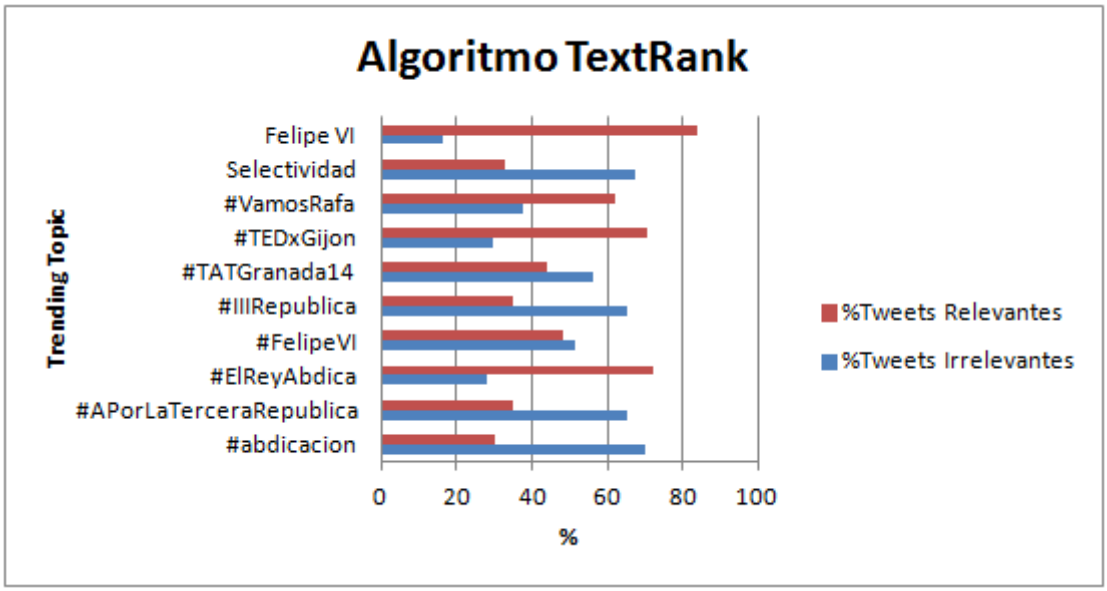


Figura 5.7 Gráfico con los porcentajes de relevancia del Algoritmo TextRank

En la figura 5.8 se muestra para cada algoritmo el porcentaje total de irrelevancia de todos sus resúmenes, donde como se esperaba, destaca el Algoritmo Random con más resúmenes irrelevantes. A su vez el que mejores resultados consigue es el Algoritmo SumBasic con un 57,4% de tuits relevantes.

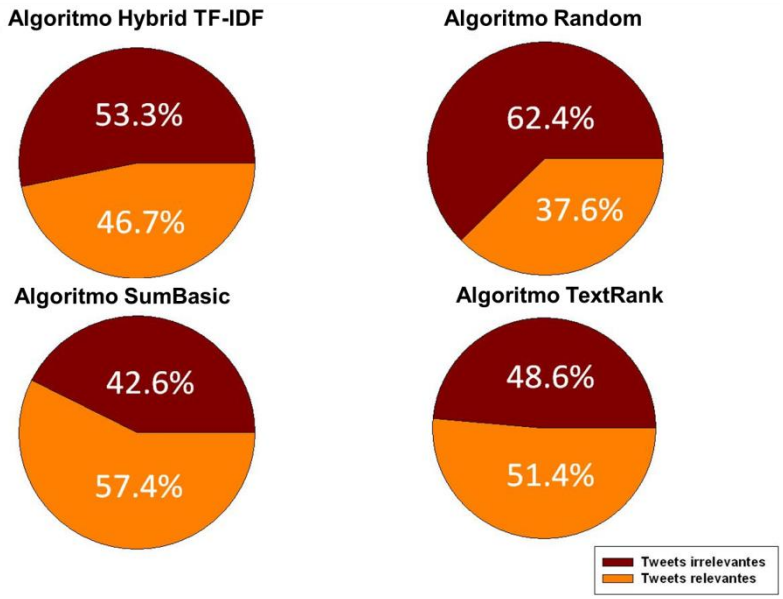


Figura 5.8 Gráfico general con el porcentaje total de irrelevancia

Otro aspecto importante a analizar durante el proceso de evaluación ha sido el tiempo que ha tardado cada usuario por trending topic, donde la media total de minutos utilizados en evaluar un tema ha sido de 11 minutos.

En la siguiente imagen (Figura 5.9) se muestra un gráfico con la media de tiempo utilizada en cada tema.

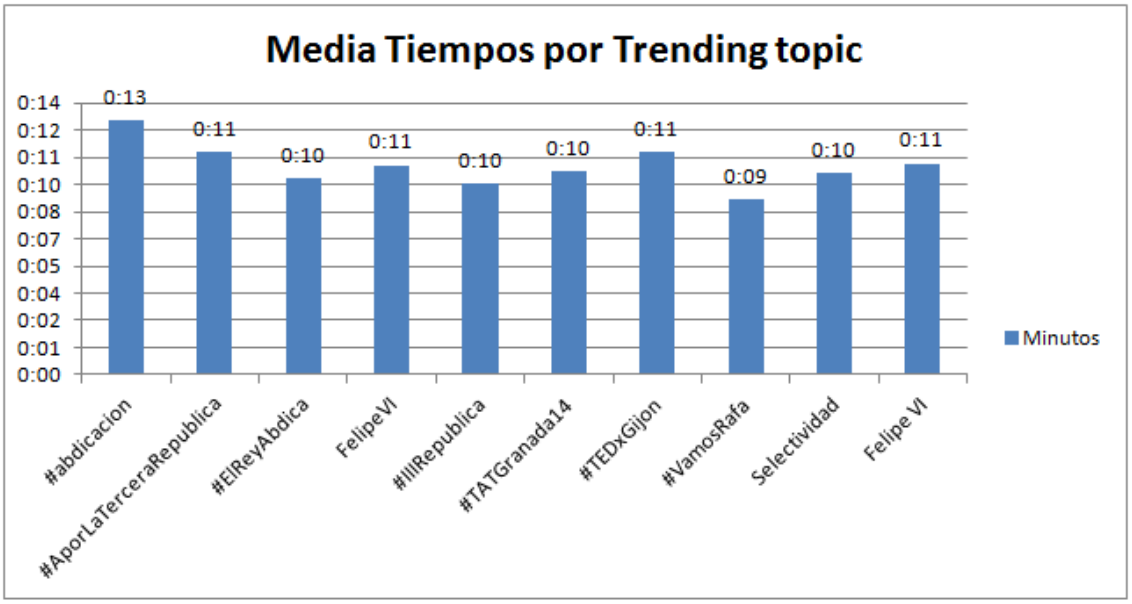


Figura 5.9 Gráfico con las media de tiempos por Trending Topic

En la siguiente imagen (Figura 5.10) se muestra un gráfico con la media de tiempo utilizada por cada evaluador en el resumen de un tema.

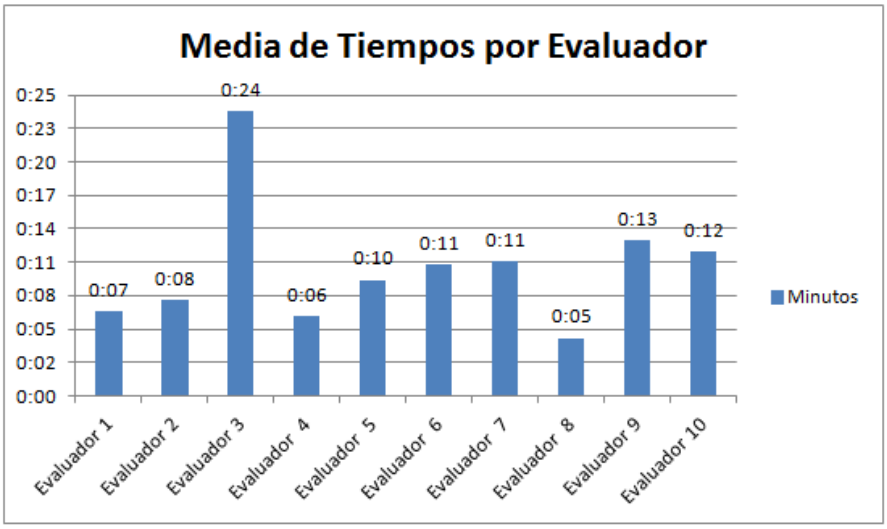


Figura 5.10 Gráfico con la media de tiempos por cada evaluador

A continuación se podrán ver las tablas resultantes para los estudios de similitud y de puntuación de calidad de resúmenes.

5.2.1 Tablas de similitud entre resúmenes humanos por cada trending topic

A continuación se mostrarán las tablas de similitud para cada trending topic obtenidas

#abdicacion	
Origen resumen	**Similitud**
Evaluador 6	0,346474448452
Evaluador 5	0,299363144789
Evaluador 1	0,282306135824
Evaluador 8	0,262960831357
Evaluador 2	0,256582867008
Evaluador 3	0,208670897692
Evaluador 4	0,202976296515
Evaluador 9	0,178365349975
Evaluador 7	0,134839972493
Promedio	0,241393327123

Tabla 5.1 Similitudes entre resúmenes humanos para tema #abdicación

#aporlatercerarepublica	
Origen resumen	**Similitud**
Evaluador 5	0,320442601907
Evaluador 4	0,290983732959
Evaluador 6	0,290398583546
Evaluador 7	0,270044011971
Evaluador 2	0,260881967480
Evaluador 3	0,256121193184
Evaluador 8	0,254665356501
Evaluador 1	0,229796501363
Evaluador 9	0,205916992027
Promedio	0,264361215660

Tabla 5.2 Similitudes entre resúmenes humanos para tema #aporlatercerrepublica

#elreyabdica	
Origen resumen	**Similitud**
Evaluador 5	0,370672671014
Evaluador 3	0,358505323602
Evaluador 9	0,356696149204
Evaluador 8	0,312891358951
Evaluador 7	0,288354918041
Evaluador 2	0,286843392734
Evaluador 6	0,247486342224
Evaluador 1	0,224352060262
Evaluador 4	0,171920476518
Promedio	0,290858076950

Tabla 5.3 Similitudes entre resúmenes humanos para tema #elReyAbdica

#felipevi	
Origen resumen	**Similitud**
Evaluador 9	0,326106221400
Evaluador 4	0,309213859580
Evaluador 3	0,296323605635
Evaluador 7	0,262475766850
Evaluador 8	0,262475766850
Evaluador 5	0,243169043783
Evaluador 6	0,222400213618
Evaluador 1	0,196121714465
Evaluador 2	0,172550454725
Promedio	0,254537405212

Tabla 5.4 Similitudes entre resúmenes humanos para tema #FelipeVI

#iiirepublica	
Origen resumen	**Similitud**
Evaluador 5	0,276504150881
Evaluador 8	0,272741187029
Evaluador 9	0,270300079858
Evaluador 2	0,231182243440
Evaluador 4	0,224506627533
Evaluador 3	0,220154210881
Evaluador 6	0,201508240457
Evaluador 1	0,158576591849
Evaluador 7	0,111376531015
Promedio	0,218538873660

Tabla 5.5 Similitudes entre resúmenes humanos para tema #IIIRespublica

#TATGranada14	
Origen resumen	**Similitud**
Evaluador 6	0,296990347971
Evaluador 4	0,282871578394
Evaluador 5	0,245404019871
Evaluador 3	0,245047548132
Evaluador 8	0,241897628545
Evaluador 9	0,223486793117
Evaluador 1	0,198312946777
Evaluador 2	0,197390189897
Evaluador 7	0,118478027219
Promedio	0,227764342214

Tabla 5.6 Similitudes entre resúmenes humanos para tema #TATGranada14

#TEDxGijon	
Origen resumen	**Similitud**
Evaluador 8	0,262765182431
Evaluador 9	0,257153746579
Evaluador 5	0,252340016119
Evaluador 4	0,234985320150
Evaluador 7	0,230537552075
Evaluador 1	0,230302882323
Evaluador 2	0,214537729388
Evaluador 3	0,098741745438
Evaluador 6	0,082855226500
Promedio	0,207135489000

Tabla 5.7 Similitudes entre resúmenes humanos para tema #TEDxGijon

#VamosRafa	
Origen resumen	**Similitud**
Evaluador 7	0,360648063067
Evaluador 1	0,332889775807
Evaluador 3	0,300086437343
Evaluador 8	0,289754346348
Evaluador 4	0,242972755464
Evaluador 5	0,242482802293
Evaluador 9	0,227267753886
Evaluador 2	0,192398631957
Evaluador 6	0,056980288230
Promedio	0,249497872711

Tabla 5.8 Similitudes entre resúmenes humanos para tema #VamosRafa

selectividad	
Origen resumen	**Similitud**
Evaluador 8	0,290634435166
Evaluador 5	0,270746636694
Evaluador 1	0,248291212801
Evaluador 9	0,238316577288
Evaluador 3	0,234137600310
Evaluador 7	0,221802780894
Evaluador 4	0,221703570060
Evaluador 2	0,173006786321
Evaluador 6	0,117500331490
Promedio	0,224015547892

Tabla 5.9 Similitudes entre resúmenes humanos para tema selectividad

Felipe VI	
Origen resumen	**Similitud**
Evaluador 9	0,307320281864
Evaluador 7	0,280109868554
Evaluador 3	0,265202591042
Evaluador 1	0,260047879096
Evaluador 4	0,244623027395
Evaluador 2	0,233468065324
Evaluador 6	0,233468065324
Evaluador 8	0,233468065324
Evaluador 5	0,232858501847
Promedio	0,254507371752

Tabla 5.10 Similitudes entre resúmenes humanos para tema Felipe VI

5.2.2 Tablas de similitud entre resúmenes de algoritmos y resúmenes humanos por trending topic

#abdicación	
Evaluador	**Similitud**
Evaluador 6	0,346474448452
Evaluador 5	0,299363144789
SumBasic	0,286959612877
Evaluador 1	0,282306135824
Evaluador 8	0,262960831357
Evaluador 2	0,256582867008
Evaluador 3	0,208670897692
Evaluador 4	0,202976296515
Random	0,199703219899
Evaluador 9	0,178365349975
TextRank	0,135088042686
Evaluador 7	0,134839972493
Hybrid	0,078481342407
Promedio Algoritmos vs Humanos	0,220982473998
Promedio Humanos vs Humanos	0,241393327123

Tabla 5.11 Similitud entre resúmenes de algoritmos y resúmenes humanos para el tema #abdicación

#AporLaTerceraRepublica	
Origen Resumen	**Similitud**
Evaluador 5	0,320442601907
Evaluador 4	0,290983732959
Evaluador 6	0,290398583546
Evaluador 7	0,270044011971
Evaluador 2	0,260881967480
Evaluador 3	0,256121193184
Evaluador 8	0,254665356501
Evaluador 1	0,229796501363
Evaluador 9	0,205916992027
SumBasic	0,139494150435
TextRank	0,105130443927
Hybrid	0,104993834943
Random	0,103654092692
Promedio Algoritmos vs Humanos	0,217886420226
Promedio Humanos vs Humanos	0,264361215660

Tabla 5.12 Similitud entre resúmenes de algoritmos y resúmenes humanos para el tema #AporLaTerceraRepublica

#ElReyAbdica	
Origen Resumen	**Similitud**
Evaluador 5	0,370672671014
Evaluador 3	0,358505323602
Evaluador 9	0,356696149204
Evaluador 8	0,312891358951
Evaluador 7	0,288354918041
Evaluador 2	0,286843392734
Evaluador 6	0,247486342224
Evaluador 1	0,224352060262
TextRank	0,189644664985
Evaluador 4	0,171920476518
SumBasic	0,162384718874
Hybrid	0,122194381622
Random	0,080378671737
Promedio Algoritmos vs Humanos	0,244025009982
Promedio Humanos vs Humanos	0,290858076950

Tabla 5.13 Similitud entre resúmenes de algoritmos y resúmenes humanos para el tema #ElReyAbdica

FelipeVI	
Origen Resumen	**Similitud**
Evaluador 9	0,326106221400
Evaluador 4	0,309213859580
Evaluador 3	0,296323605635
Evaluador 7	0,262475766850
Evaluador 8	0,262475766850
Evaluador 5	0,243169043783
Evaluador 6	0,222400213618
Evaluador 1	0,196121714465
TextRank	0,177159143740
Evaluador 2	0,172550454725
Hybrid	0,168517234289
SumBasic	0,149984506533
Random	0,085550538850
Promedio Algoritmos vs Humanos	0,220926774640
Promedio Humanos vs Humanos	0,254537405212

Tabla 5.14 Similitud entre resúmenes de algoritmos y resúmenes humanos para el tema Felipe VI

#IIIRepublica	
Origen Resumen	**Similitud**
Evaluador 5	0,276504150881
Evaluador 8	0,272741187029
Evaluador 9	0,270300079858
Hybrid	0,242163798056
Evaluador 2	0,231182243440
Evaluador 4	0,224506627533
Evaluador 3	0,220154210881
Evaluador 6	0,201508240457
SumBasic	0,168922334622
Evaluador 1	0,158576591849
Random	0,148332150617
TextRank	0,138258178256
Evaluador 7	0,111376531015
Promedio Algoritmos vs Humanos	0,204963563423
Promedio Humanos vs Humanos	0,218538873660

Tabla 5.15 Similitud entre resúmenes de algoritmos y resúmenes humanos para el tema #IIIRepublica

#TATGranada14	
Origen Resumen	**Similitud**
Evaluador 6	0,296990347971
Evaluador 4	0,282871578394
Evaluador 5	0,245404019871
Evaluador 3	0,245047548132
Evaluador 8	0,241897628545
Evaluador 9	0,223486793117
Evaluador 1	0,198312946777
Evaluador 2	0,197390189897
Hybrid	0,182347123599
SumBasic	0,143217888674
TextRank	0,142636082684
Random	0,124914354221
Evaluador 7	0,118478027219
Promedio Algoritmos vs Humanos	0,203307271469
Promedio Humanos vs Humanos	0,227764342214

Tabla 5.16 Similitud entre resúmenes de algoritmos y resúmenes humanos para el tema #TATGranada14

#TEDxGijon	
Origen Resumen	**Similitud**
Evaluador 8	0,262765182431
Evaluador 9	0,257153746579
Evaluador 5	0,252340016119
Evaluador 4	0,234985320150
Evaluador 7	0,230537552075
Evaluador 1	0,230302882323
Hybrid	0,221361868179
Evaluador 2	0,214537729388
Random	0,163028719433
SumBasic	0,117394858564
TextRank	0,117394858564
Evaluador 3	0,098741745438
Evaluador 6	0,082855226500
Promedio Algoritmos vs Humanos	0,191030746596
Promedio Humanos vs Humanos	0,207135489000

Tabla 5.17 Similitud entre resúmenes de algoritmos y resúmenes humanos para el tema #TEDxGijon

#VamosRafa	
Origen Resumen	**Similitud**
Evaluador 7	0,360648063067
Evaluador 1	0,332889775807
Evaluador 3	0,300086437343
Evaluador 8	0,289754346348
Evaluador 4	0,242972755464
Evaluador 5	0,242482802293
Evaluador 9	0,227267753886
Evaluador 2	0,192398631957
Hybrid	0,148765516588
SumBasic	0,144841364876
TextRank	0,144841364876
Random	0,136447695126
Evaluador 6	0,056980288230
Promedio Algoritmos vs Humanos	0,216952061220
Promedio Humanos vs Humanos	0,249497872711

Tabla 5.18 Similitud entre resúmenes de algoritmos y resúmenes humanos para el tema #VamosRafa

Selectividad	
Origen Resumen	**Similitud**
TextRank	0,300917232977
Evaluador 8	0,290634435166
Evaluador 5	0,270746636694
Evaluador 1	0,248291212801
Evaluador 9	0,238316577288
Hybrid	0,235446131811
Evaluador 3	0,234137600310
Evaluador 7	0,221802780894
Evaluador 4	0,221703570060
Random	0,176870720459
Evaluador 2	0,173006786321
SumBasic	0,150710009507
Evaluador 6	0,117500331490
Promedio Algoritmos vs Humanos	0,221544925060
Promedio Humanos vs Humanos	0,224015547892

Tabla 5.19 Similitud entre resúmenes de algoritmos y resúmenes humanos para el tema Selectividad

Felipe VI	
Origen Resumen	**Similitud**
Evaluador 9	0,307320281864
Evaluador 7	0,280109868554
Evaluador 3	0,265202591042
Evaluador 1	0,260047879096
Evaluador 4	0,244623027395
Evaluador 2	0,233468065324
Evaluador 6	0,233468065324
Evaluador 8	0,233468065324
Evaluador 5	0,232858501847
TextRank	0,213886250975
Random	0,191522948091
SumBasic	0,171109000780
Hybrid	0,111220850507
Promedio Algoritmos vs Humanos	0,229100415086
Promedio Humanos vs Humanos	0,254507371752

Tabla 5.20 Similitud entre resúmenes de algoritmos y resúmenes humanos para el tema

5.2.3 Tabla con las puntuaciones medias de cada evaluador para cada trending topic

	#abdicacion	#APorLaTerceraRepublica	#ElReyAbdica	#FelipeVI	#IIIRepublica	#TATGranada14	#TEDxGijon	#VamosRafa	Selectividad	Felipe VI	Promedio
Evaluador 5	8	8,33333	8,44444	8,33333	8,55555	8,44444	8,77777	7,88888	7,66666	7,88888	8,233328
Evaluador 8	7,66666	8,11111	8,22222	8	8,11111	7,33333	8,77777	7,88888	7,66666	8	7,977774
Evaluador 1	7,55555	8,33333	7,66666	8,66666	6,77777	8,88888	6,33333	8,22222	8,77777	8,33333	7,95555
Evaluador 9	6,22222	8	8,22222	8,11111	7,66666	7	8,33333	7,55555	7	8,22222	7,633331
Evaluador 4	7,77777	7,88888	6,88888	8,33333	7	8,66666	7,88888	6,88888	7	8	7,633328
Evaluador 3	6,66666	7,66666	7,88888	8,55555	5,33333	8,88888	4,33333	7,55555	7,44444	8,11111	7,244439
Evaluador 6	8	7,88888	8,11111	8,11111	7,33333	7,66666	3,88888	5,88888	6,11111	8	7,099996
Evaluador 2	4,55555	7,55555	8,11111	5,44444	7,44444	7,77777	5,77777	6,55555	6,66666	8	6,788884
Evaluador 7	7,11111	7,66666	8	8,11111	4,55555	0	8	7,22222	7,77777	8	6,644442
Evaluador 10	5,33333	1,88888	2,44444	4,11111	4,77777	2,33333	1,55555	4,88888	5,77777	3,22222	3,633328

Tabla 5.21 Puntuación medias de resúmenes de evaluadores por tema

5.2.4 Tabla con las puntuaciones medias de cada algoritmo para cada trending topic

	#abdicacion	#APorLaTerceraRepublica	#ElReyAbdica	#FelipeVI	#IIIRepublica	#TATGranada14	#TEDxGijon	#VamosRafa	Selectividad	Felipe VI	Promedio
PROMEDIO HUMANOS	6,888885	7,333328	7,399996	7,577775	6,755551	6,699995	6,366661	7,055549	7,188884	7,577776	7,08444
TEXTRANK	1,77777	5,22222	7,33333	4,22222	5,33333	8,11111	4,55555	7,11111	8,11111	8	5,977775
SUMBASIC	3,11111	7,22222	6,33333	4,11111	6,55555	1,44444	4,55555	7,11111	3,77777	7,33333	5,155552
HYBRID	4,55555	4,22222	7,66666	2,77777	6,11111	1,33333	4,33333	6,11111	5,66666	4,55555	4,733329
RANDOM	4,22222	5,33333	1,11111	6	4,44444	3,77777	4,11111	7,55555	2,66666	1,33333	4,055552

Tabla 5.22 Puntuación medias de resúmenes de algoritmos por tema

5.3 Discusión de los resultados

Se han comparado los resúmenes escritos por los evaluadores entre sí a fin de determinar si los resultados son razonablemente consistentes entre sí.

Para ello se ha recurrido a la similitud del coseno aplicada sobre vectores de 3-gramas de caracteres. El procedimiento consistió en determinar la similitud del coseno entre un vector de 3-gramas generado para los resúmenes de un único usuario y otro vector de 3-gramas generado a partir de los resúmenes del resto de usuarios.

La evaluación se hizo de dos formas diferentes: en primer lugar teniendo en cuenta todos los resúmenes producidos y en segundo lugar teniendo en cuenta sólo el resumen principal.

En ambos casos se calculó la media de las similitudes para todos los usuarios para determinar en qué casos un usuario obtenía una puntuación sustancialmente inferior (un 10%) a la media de dichas similitudes.

Se observó que el evaluador 10 alcanzaba puntuaciones sustancialmente inferiores en el 80% de los temas si se evaluaba con todos los resúmenes y en el 100% de los temas si sólo se evaluaba con el resumen principal. Un análisis de los resúmenes producidos por ese evaluador mostró que sus resúmenes eran tremendamente subjetivos y consistían, fundamentalmente, en interpretaciones personales del evento más que en resúmenes abstractivos de los tuits leídos.

En consecuencia se procedió a eliminar el material del evaluador 10 para el cómputo de las similitudes de los resúmenes del resto de usuarios.

Al hacer eso se observó que aún había evaluadores que ocasionalmente alcanzaban puntuaciones inferiores a la media pero para unos pocos temas por lo que no se eliminó ningún otro usuario de la evaluación.

Similitud promedio entre los resúmenes de los distintos evaluadores al utilizar únicamente el resumen principal:

Similitud Resumen Principal	
Trending Topic	**Similitud promedio**
#abdicacion	0,241393327123
#aporlatercerarepublica	0,264361215660
#elreyabdica	0,290858076950
#felipevi	0,254537405212
#iiirepublica	0,218538873660
#tatgranada14	0,227764342214
#tedxgijon	0,207135489000
#vamosrafa	0,249497872710
felipe vi	0,254507371752
selectividad	0,224015547892
Promedio global	0,243260952217

Tabla 5.23 Similitud media entre resúmenes principales para cada Trending Topic

Similitud promedio entre los resúmenes de los distintos evaluadores al utilizar todos los resúmenes:

Similitud Todos los resúmenes evaluadores	
Trending Topic	**Similitud promedio**
felipe vi	0,244447959896
#abdicacion	0,243839367335
#aporlatercerarepublica	0,264972042689
#elreyabdica	0,276629535316
#felipevi	0,252993321915
#iiirepublica	0,243264054590
#tatgranada14	0,243148156620
#tedxgijon	0,221912487443
#vamosrafa	0,259585612000
selectividad	0,218709001188
Promedio global	0,246950154

Tabla 5.24 Similitud media entre todos los resúmenes humanos para cada Trending Topic

Como se puede comprobar la similitud promedio ronda el 0.25; teniendo en cuenta que la similitud del coseno varía entre 0 (disimilitud total) y 1 (textos idénticos) está claro que aunque hay un grado de coincidencia razonable entre los evaluadores lo cierto es existen discrepancias obvias entre los resúmenes que producen.

En general, el rendimiento de todos los algoritmos es inferior al de los humanos puesto que, en promedio, obtienen valores de similitud muy inferiores (casi un 35%) al del humano promedio.

No hay ningún algoritmo que parezca superior al resto puesto que los resultados no son especialmente consistentes.

De hecho, el algoritmo aleatorio que debería ser el peor obtiene en ocasiones mejores resultados que otros algoritmos.

En un par de casos un algoritmo obtuvo resultados mejores que parte de los humanos (p.ej. sumbasic para #abdicacion, hybrid para #iiirepublica o textrank para selectividad). No obstante, estos casos parecen aislados, no sugieren ninguna pauta y, en consecuencia, no parece razonable optar por un resumen totalmente automático aunque sí parece razonable utilizar una combinación de estos algoritmos para realizar un filtro previo que sirva para que, posteriormente, un ser humano escriba el resumen final.

En cuanto a los resultados obtenidos con las puntuaciones realizadas por los evaluadores a sus propios resúmenes y a los resúmenes generados por los algoritmos se puede observar que la nota media para el resumen de un evaluador es aproximadamente de un 7, mientras que la de los algoritmos son más bajas, en términos generales el algoritmo Random ha conseguido una calificación media de 4,0555, como se esperaba peor que los demás algoritmos, pero hay casos aislados en el que este algoritmo recibía mejores puntuaciones que los otros.

El algoritmo con mayor puntuación ha sido el SumBasic con una valoración media de 5,9777, seguido del TextRank con 5,1555, sin embargo el algoritmo Hybrid TF-IDF no ha conseguido llegar al 5, no ha funcionado como se esperaba.

Pese a que habría que hacer mejoras en el prototipo, se puede observar en este experimento que sería factible desarrollar una aplicación híbrida donde los algoritmos implementados filtran una cantidad ingente de datos que un humano sería incapaz de filtrar en un tiempo razonable, esos datos filtrados generarían una serie de resúmenes donde su nota media rondaría el 5 y se darían a unos evaluadores humanos de forma que siempre tendrían que evaluar una cantidad determinada de tuits donde aplicando su conocimiento se obtendría un resumen abstractivo cuya nota rondaría media entre ellos rondaría el 7.

Fijándonos en la nota que ha sacado el mejor evaluador ha sido de 8,233328, sin embargo, el evaluador que se descartó para comprobar la similitud entre resúmenes ha obtenido una calificación media aproximada de un 3,6. Que aplicándolo a este prototipo que se querría hacer, sería uno de los riesgos a la hora de implementar un sistema de este tipo.

Capítulo 6 Conclusiones y trabajo final

6.1 Conclusiones sobre los prototipos

Partiendo de que no existe nada parecido a nivel de aplicaciones de la obtención de resúmenes abstractivos de forma automática se decidió seguir ese enfoque con las siguientes conclusiones extraídas del desarrollo del trabajo.

- Después del estudio realizado no se puede afirmar que hay un algoritmo claramente superior a los demás en cuanto a la extracción de resúmenes.
- Pese al punto anterior, la utilización de este tipo de aplicación, sí serviría como filtro con el fin de montar un sistema híbrido para generar descripciones, donde el algoritmo proporcionará siempre O(n) posibles resúmenes a un grupo de evaluadores humanos que serían los encargados de formar obtener el resumen abstractivo. Realizar la labor de sintetizar un tema en n tuits de un volumen de información semejante a la que procesa este prototipo resulta inviable para llevarla a cabo por personas, es por eso que este sistema facilitaría la tarea de evaluación para su posterior evaluación humana.
- Los resultados obtenidos en las pruebas de evaluación indican que habría que buscar formas de mejorar los resultados de estos algoritmos y realizar muchas más pruebas
- Hay que realizar muchas más pruebas para avanzar en la obtención de resumen abstractivo aplicando mejoras al prototipo.

6.2 Trabajo futuro

- Hacer implementación de aplicación para la evaluación de resúmenes

- Introducir filtrado de SPAM utilizando clasificador Bayesiano incluyendo un proceso de entrenamiento para este clasificador.

- Experimentar con métodos de detección de cuasi-duplicados para evitar tuits con contenido extremadamente parecido. **[Charikar02],** **[Singh07].**

- Conseguir mucho de este tipo de resúmenes y ver de si algún modo se puede aplicar una técnica de aprendizaje automático que aprenda a seleccionar los mejores tuits.

- ¿Cómo sé que un Trending topic no está cambiando tanto y no merece la pena hacer un resumen de él?

- ¿Cómo serían si esas descripciones cambian geográficamente? ¿Se podrían detectar cambios de opinión?

- ¿Se podría saber si cuando un Trending topic es sobre un tema o alguien en particular los comentarios hacia él son positivos o negativos?

- Ver cómo funcionan en otros idiomas

Capítulo 7 Referencias Bibliográficas

7.1 Libros y Artículos

Libros y artículos usados de alguna forma durante el desarrollo del proyecto o su documentación.

[Chakrabarti11] Chakrabarti, Deepayan; Punera, Kunal. "Event Summarization using Tweets" Sunnyvale 2011 .

[Charikar02] S. Charikar, Moses "Similarity Estimation Techniques from Rounding Algorithms". Princeton University 2011.

[Filippova10] Filippova, Katja. "Multi-Sentence Compression: Finding Shortest Paths in Word Graphs" Google Inc 2010.

[Harabagiu11] Harabagiu, Sanda; Hickl, Andrew. "Relevance Modeling for Microblog Summarization" University of Texas at Dallas 2011.

[Inouye11] Inouye, David; K. Kalita, Jugal "Comparing Twitter Summarization Algorithms for Multiple Post Summaries". University of Colorado 2011

[Lin12] Lin, Chen; Lin, Chun; Li, Jingxuan; Wang, Dingding; Chen, Yang; Li, Tao. "Generating Event Storylines from Microblogs". Florida International University 2012.

[Luhn58] H. P. Luhn, "The automatic creation of literature abstracts," IBM J. Res.

Dev., vol. 2, no. 2, pp. 159–165, 1958.

[Mihalcea04] Mihalcea, Rada; Tarau, Paul. "Text Rank: Bringing Order into Texts". University of Texas 2004.

[Nichols12] Nichols, Jeffrey; Mahmud, Jalal; Drews, Clemens. "Summarizing Sporting Events Using Twitter" IBM Research –Almaden 2012.

[O'Connor10] O'Connor, Brendan; Krieger, Michel; Ahn, David. "TweetMotif: Exploratory Search and Topic Summarization from Twitter" Association for the Advancement of Artificial Intelligence 2010

[Sharifi10] Sharifi,Beaux; Hutton, Mark-Anthony; K. Kalita, Jugal. "Experiments in Microblog Summarization". University of Colorado 2010

[SharifiBeaux10] Sharifi,Beaux; Hutton, Mark-Anthony; K. Kalita, Jugal. "Summarizing Microblogs Automatically". University of Colorado 2010

[Singh07] Singh, Gurmeet; Jain, Arvind; Das Sarma, Anish "Detecting Near-Duplicates for Web Crawling". International World Wide Web Conference Comitee 2007

[Takamura11] Takamura, Hiroya; Yokono, Hikaru; Okumura, Manabu. "Summarizing a Document Stream" Tokyo Institute of Technology 2011.

[Vanderwendea07] Vanderwendea ,Lucy; Suzukia, Hisami ; Brocketta, Chris; Nenkova, Ani. "Beyond SumBasic: Task-Focused Summarization with Sentence Simplification and Lexical Expansion." Columbia University 2007

[Xintian12] Yang, Xintian; Ghoting, Amol; Ruan, Yiye. "A framework for Summarizing and Analyzing Twitter Feeds" The Ohio State University 2012

[Zubiaga12] Zubiaga, Arkaitz; Spina, Damiano; Amigó, Enrique; Gonzalo, Julio. "Towards Real-Time Summarization of Scheduled Events from Twitter Streams" ACM 2012.

7.2 Referencias en Internet

[APITwitter] "Twitter Developers" https://dev.twitter.com/

[Java] "Información sobre Java" http://es.wikipedia.org/wiki/Java_(lenguaje_de_programaci%C3%B3n)

[PHP] "PHP: Manual de PHP - Manual" http://www.php.net/manual/es/index.php

Capítulo 8 Apéndices

8.1 Contenido Entregado en los Anexos

8.1.1 Contenidos

8.1.1.1.1 Estructura de directorios prototipo de generación de resúmenes

Directorio	Contenido
./ Directorio raíz del CD	Contiene un fichero leeme.txt explicando toda esta estructura. Se puede incluir *autorun* e icono del proyecto si existe.
./ObtencionResumenes	Contiene toda la estructura de directorios del proyecto para desarrollo
/ObtencionResumenes/Algoritmos	Contiene toda la estructura de directorios del prototipo de generación de resumenes
/ObtencionResumenes/descargador	Contiene toda la estructura de directorios del prototipo recopilador de información
/ObtencionResumenes/ Script evaluacion Similitudes	Contiene el Script utilizado para el cálculo de similitudes entre resúmenes
./documentacion	Contiene toda la documentación asociada al proyecto. Es necesario incluir un fichero con el documento final del proyecto (en formato *.doc* o *.docx* de *Word* o bien *.sxw* de *Open Office*, por ejemplo) además de un fichero *.PDF*
./documentacion/img	Directorio que contiene las imágenes utilizadas en la documentación. Estas imágenes tendrán formato *.png* si son capturas de pantalla, *.wmf* si son diagramas o esquemas y *.jpg* sólo si son fotografías.
./documentacion/uml	Ficheros que se han generado con los diagramas UML y de entidad relación.
/documentación/evaluacion	Contiene todos los ficheros Excel generados en la evaluación para hacer los cálculos pertinenentes así como los archivos que contienen los resúmenes generados por los algoritmos para la evaluación
./herramientas	Contiene los ficheros de instalación de las herramientas utilizadas para el desarrollo o puesta en marcha del proyecto (lógicamente sólo las que sean distribuibles).
./herram/desarrollo	Ficheros que indican cómo descargar las herramientas para el desarrollo de los prototipos

./herram/explotacion	BD, servidor Web y herramientas en general.

8.1.1.1.2 Estructura de directorios prototipo de generación de resúmenes

Directorio	Contenido
./ Algoritmos	Contiene los ficheros de proyecto del IDE utilizado.
./resumen	Directorio donde se almacenarán los resúmenes generados por el algoritmo
./Tweets	Directorio donde se deberán introducir las carpetas con los trending topics a resumir
/Filtrado	Directorio en el que se almacenaran los ficheros filtrados
./lib	Bibliotecas externas (*.jar*, *.dll*, …) necesarias para compilar y distribuir, de las que depende este proyecto.
/util	Directorio donde se encuentran almacenadas en fichero las stopwords
./src	Ficheros fuente
./src/java	Todos los ficheros *Java*, lógicamente agrupados en los paquetes correspondientes.

Anexo: 1 Código de los prototipos desarrollados

El código fuente tiene que ir dividido por paquetes y por archivos, con un formato que haga que resulte legible, tal y como se muestra a continuación en este ejemplo. En la mayoría de las ocasiones, es posible que no sea conveniente colocar la totalidad del código fuente en esta sección,

1.1 Código fuente Recopilador de información

1.1.1 Script BBDD

```
CREATE TABLE `consulta_tweets` (
  `id_consulta_tweet` int(11) NOT NULL AUTO_INCREMENT,
  `texto_ttoppic` varchar(45) NOT NULL,
  `timestamp` datetime NOT NULL,
  `ruta` text NOT NULL,
  `t_creado` datetime DEFAULT NULL,
  PRIMARY KEY (`id_consulta_tweet`),
  KEY `texto_ttoppic` (`texto_ttoppic`,`timestamp`),
  CONSTRAINT `consulta_tweets_ibfk_1` FOREIGN KEY (`texto_ttoppic`, `timestamp`)
REFERENCES `ttoppics` (`texto`, `timestamp`)
) ENGINE=InnoDB AUTO_INCREMENT=228 DEFAULT CHARSET=utf8

delimiter $$

CREATE TABLE `ttoppics` (
  `texto` varchar(45) NOT NULL,
  `timestamp` datetime NOT NULL,
  `posicion` int(11) NOT NULL,
  `tcreado` datetime DEFAULT NULL,
  `woeid` int(11) NOT NULL,
  PRIMARY KEY (`texto`,`timestamp`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
```

1.1.2 Fichero “index.php”

```
<?php
        require('recott.php');
        require('filter-track.php');
        require('procesadorficheros.php');

        while(true){

                //Se descargan los ttopics y se almacena su lista en una variable
                $consulta=descargartt();

                $timestamp=date("Y-m-d H:i:s");
                $timestampa=$timestamp;
                $timestamp=str_replace(":","",$timestamp);
```

```
                $timestamp=str_replace(" ","",$timestamp);
                //$consulta = utf8_encode ( $consulta );

                //Se llama a la función de descarga de tweets
                descargartw($consulta,$timestamp);

                $ttoppics = explode(",", $consulta);

                //print_r($ttoppics);

                //Se llama a la función que procesará el fichero de descarga
                lectura($timestamp,$ttoppics);

                //Se llama a la función de insertar consultas en la BBDD
                insConsultas($ttoppics,$timestamp,$timestampa);

                //Se llama a la función de Borrado del fichero que guarda la Streaming
API
                //borradoFicherosTotal();

        }

        /**************************************************************************************
************************************/

        //Funcion encargada de la descarga de la descarga de tweets
        function descargartw($consulta,$timestamp){
                descargatweets($consulta,$timestamp);
        }


        //Funcion encargada de la descarga de trending topics
        function descargartt(){
                echo '<p>Descargar Trending Toppic</p>';

                //insertarTT();
                $woeid = "23424950"; //para España
                $consultaSpain=descargar($woeid);
                echo ("Consulta SPAIN twetts ".$consultaSpain);
                $woeid = "23424977"; //para EEUU
                $consultaEEUU=descargar($woeid);
                $consultaFinal=$consultaSpain.','.$consultaEEUU;

                return $consultaFinal;

        }

        //Funcion encargada de la inserción en la BBDD de las consultas realizadas para
la busqueda de tweets
        function insConsultas($ttoppics,$timestamp, $timestampa){
                $cn = mysql_connect('localhost', 'root', 'Proyecto');
                mysql_select_db('proyecto_test', $cn);
                $carpeta=substr($timestamp,0,10);

                for($i = 0; $i < count($ttoppics); $i++)
                {

                        $ttoppic= $ttoppics[$i];
                        $resultado = strpos($texto, $ttoppic);

                        $timestamp = trim($timestamp, 'Z');
                        $rutamysql='datos\\\\'.$carpeta.'\\\\'.$timestamp.'-
'.$ttoppic.'.txt';
                        $consulta="INSERT INTO
CONSULTA_TWEETS(texto_ttoppic,ruta,timestamp) values
('".$ttoppic."','".$rutamysql."','".$timestampa."')";
                                        //echo('<p>'.$consulta.'</p>');

                        mysql_query($consulta);

                }
                mysql_close($cn);

        }

        //Función que efectua el borrado del fichero de descarga de la Streaming API
        function borradoFicherosTotal(){
```

```
                $dir = "datos\\total\\";
                $handle = opendir($dir);

                while (false!=($file = readdir($handle)))
                {
                        if (is_file($dir.$file))
                        {
                                unlink($dir.$file);
                        }
                }

        }

?>
```

1.1.3 Fichero recott.php

```
<?php
	require('conexion.php');
		function descargar($woeid){

			try{
				$connection = getConnectionWithAccessToken();//Realizamos
la conexión
				$response=$connection-
>get('https://api.twitter.com/1.1/trends/place.json', array('id' => $woeid));

			print_r($response);

			echo "####################";
			$consulta=decodificar($response);
			return $consulta;
			}catch(Exception $e){
				echo 'Excepción capturada: ',  $e->getMessage(), "\n";
				return 'Excepción capturada: '.  $e->getMessage();
			}

		}

		function decodificar($datos){

			$data=$datos[0]->trends; //Se obtienen los trends y se introducen
en la variable $data
			$posicion=1; //Se almacena la posición del primer trending toppic

			$locations=$datos[0]->locations;
			$timestamp=$datos[0]->as_of;

			echo ('timestamp: '.$timestamp);
			$woeid=$locations[0]->woeid;

			$consultatweets="";
			foreach ($data as $trend) {

				$texto=utf8_decode($trend->name);
				//$texto=$trend->name;
				$consultatweets=$consultatweets.$texto.",";
				insertarTT($texto,$posicion,$woeid,$timestamp);

				$posicion++; //Se incrementa la posición
			}
			$consultatweets = substr ($consultatweets, 0,
strlen($consultatweets) - 1);
			return $consultatweets;
		}

		//Método para insertar Trending toppic en la Base de datos
		function insertarTT($texto,$posicion,$woeid,$timestamp)
		{
				$cn = mysql_connect('localhost', 'root', 'Proyecto');
				mysql_select_db('proyecto_test', $cn);
				$timestamp = trim($timestamp, 'Z');

				$consulta="INSERT INTO
TTOPPICS(texto,posicion,woeid,timestamp)
values('".$texto."',".$posicion.",".$woeid.",'".$timestamp."')";
				echo('<p>'.$consulta.'</p>');
				mysql_query($consulta);
				echo('<p>'.$texto.' insertado con éxito</p>');
				mysql_close($cn);
		}
?>
```

1.1.4 Fichero "Filtertrack.php"

```
<?php
require_once('lib/Phirehose.php');
require_once('lib/OauthPhirehose.php');

/**
 * Example of using Phirehose to display a live filtered stream using track words
 */
class FilterTrackConsumer extends OauthPhirehose
{
  /**
   * Enqueue each status
   *
   * @param string $status
   */
  public function enqueueStatus($status,$timestamp)
  {

        $carpeta="total";
        guardarFichero($carpeta,$nombre,$status,$timestamp);

  }
}


        define('TWITTER_CONSUMER_KEY','hXTlvpKVTIoYlDrIaHPVg');
        define('TWITTER_CONSUMER_SECRET','FxzQz7PeLucfQpVL1H5UyYRGXfVfM42S1FXPzMuJZY');
        define('OAUTH_TOKEN','381232831-fFj68crxZMj8zLD2SPbq4NjBKGUndcvCckFw3cZo');
        define('OAUTH_SECRET','EAciuSgo0l0JB44AKqWZQT6iW6Clo4ZT9GVIiNCfXNM');

        //Función utilizada para guardar los tweets descargados un fichero
        function guardarFichero($carpeta,$nombre,$json,$timestamp){
                if(!file_exists("datos\\\\"))
                mkdir("datos\\\\",0777);
                //echo ("No hay datos");

                if(!is_dir("datos\\".$carpeta."\\"))
                        mkdir("datos\\".$carpeta."\\", 0777);
                        $ruta='datos\\'.$carpeta.'\\'.$timestamp.'-total.txt';
                        $rutamysql='datos\\\\'.$carpeta.'\\\\'.$timestamp.'-
'.$nombre.'.txt';
                        echo("a");
                        $fp = fopen($ruta, 'a');
                        $json=$json."\n";
                        fputs($fp, $json);
                        fclose($fp);

        }

        //Funcion encargada de realizar la descarfga de Tweets
        function descargatweets($consulta,$timestamp){
                $sc = new FilterTrackConsumer(OAUTH_TOKEN, OAUTH_SECRET,
Phirehose::METHOD_FILTER);

                $sc->setTrack(array($consulta));
                $sc->setTimestamp($timestamp);
                $sc->consume();
        }?>
```

1.1.5 Fichero "ProcesadorFicheros.php"

```
<?php

//Función que comprueba a que trending topic pertenece un tweet determinado
function compruebaTweet($json, $ttoppics,$timestamp){
	$linea = json_decode($json,true);
	$texto=$linea["text"];
	$resultado=false;
	for($i = 0; $i < count($ttoppics)&&$resultado!=true; $i++)
	{

		$ttoppic= $ttoppics[$i];

		$resultado = strpos($texto, $ttoppic);
	}

	if($resultado !== FALSE){
		//se procede a guardar el tweet sabiendo a que trending topic corresponde
		guardarTweet($ttoppic,$json,$timestamp);

	}

}

//Función encargada de guardar en un fichero txt el tweet descargado
	function guardarTweet($nombre,$json,$timestamp){

		$carpeta=substr($timestamp,0,10);
		$timestamp=str_replace("-","",$timestamp);
		//si no existe el directorio se crea
		if(!is_dir("datos\\".$carpeta."\\"))
			mkdir("datos\\".$carpeta."\\", 0777);
		$ruta='datos\\'.$carpeta.'\\'.$timestamp.'-'.$nombre.'.txt';


		$fp = fopen($ruta, 'a');
		fwrite($fp, $json);
		fclose($fp);

	}
//Función encargada de la lectura del fichero descargado por la streaming api para su
procesamiento y obtención de los tweets para cada ttoppic
	function lectura($timestamp,$ttoppics){
		$file = fopen("datos\\total\\".$timestamp."-total.txt", "r") or
exit("Unable to open file!");
		//insConsultas($json,$ttoppics,$timestamp);
		while(!feof($file))
		{
			$json= fgets($file);

			compruebaTweet($json,$ttoppics,$timestamp);

		}
		fclose($file);

	}

?>
```

1.2 Código fuente Generador de resúmenes

1.2.1 Paquete AlgoritmoHybrid:

1.2.1.1 Fichero "AlgoritmoHybrid.java":

```
package impl.miw.AlgoritmoHybrid;

import impl.miw.entidad.StopWord;
import impl.miw.entidad.Tweet;
import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Collections;
import java.util.HashMap;
import java.util.Iterator;
import java.util.List;
import java.util.Map;
import java.util.StringTokenizer;

import com.miw.business.AlgoritmoHybridManagerService;

public class AlgoritmoHybrid  {

	//Número de palabras que contiene el documento
	private int numPalabras=0;

	//Número de post en el documento
	private int numPosts=0;

	//Map con las palabras y sus apariciones
	private Map<String, Integer> apariciones;

	//Map con las palabras y numero de sentencias en las que aparece
	private Map<String, Integer> aparicionesSentencia;

	//Map con las palabras y sus frecuencias
	private Map<String, Double> frecuencias;

	//Map con las palabras y sus frecuencias
	private Map<String, Double> frecuenciasIDF;

	//Lista con los posibles resúmenes
	private List<Tweet> resumenes;

	//Idioma en el que se va aplicar el algoritmo
	private String idioma;

	private StopWord stopword;


	/**
	 * Método constructor de la clase
	 */
	public AlgoritmoHybrid(){
		idioma="es";
		stopword= new StopWord(idioma);
		aparicionesSentencia= new HashMap<String, Integer>();
		apariciones= new HashMap<String, Integer>();
		frecuencias= new HashMap<String, Double>();
		frecuenciasIDF= new HashMap<String, Double>();
		resumenes= new ArrayList<Tweet>();
```

```
	}


	@Override
	public void getResumen(String ruta) {

		procesarFichero(ruta);
		calcularProbabilidades();
		calcularProbabilidadesIDF();
		pesarTweets(ruta);
		System.out.println("Resumenes Hybrid");
		System.out.println(resumenes.toString());

	}



	/**
	 * Método que calcula las palabras
	 * @param texto
	 */
	private void calcularPalabras(String texto) {
		StringTokenizer st = new StringTokenizer(texto);
		List <String> lista= new ArrayList<String>();


		while (st.hasMoreTokens()) {
			String palabra = st.nextToken();
			palabra=palabra.toLowerCase();
				if(!stopword.isStopword(palabra)){
					aparacionesPalabra( palabra);

					if(!lista.contains(palabra)){
						lista.add(palabra);

					}

				numPalabras++;
				}

				aparicionesSentencia(lista);
			}
	}
	/**
	 * Método que incrementa en una unidad el número de veces que aparece una palabra
en el documento de entrada
	 * @param palabra
	 */
	private void aparacionesPalabra(String palabra){
		Integer aparicion = apariciones.get(palabra);
		apariciones.put(palabra, (aparicion == null) ? 1 : aparicion + 1);

	}


	/**
	 * Método que incrementa en una unidad el número de sentencias en el que aparece
la palabra determinada
	 * @param lista
	 */
	private void aparicionesSentencia(List<String> lista) {
		Iterator<String> it = lista.iterator();
		while (it.hasNext()) {
			String palabra=it.next();
			Integer aparicion = aparicionesSentencia.get(palabra);
			aparicionesSentencia.put(palabra, (aparicion == null) ? 1 :
aparicion + 1);

		}

	}


	/**
```

```
         * Método utilizado para el procesamiento del fichero donde se calcula el número
de apariciones por palabra
         * @param ruta
         */
        private void procesarFichero(String ruta){
                        String linea="";
                        try{
                                FileReader fr = new FileReader(ruta);
                                BufferedReader bf = new BufferedReader(fr);


                                //Obtención del texto del tweet
                                while((linea = bf.readLine())!=null){

                                        //Incrementamos el número de post en 1
                                        numPosts++;
                                        String [] campo= linea.split(";");

                                        String tweet=campo[1];


                                        //Calculo del número de apariciones por palabra
                                        calcularPalabras(tweet);


                                }
                                fr.close();
                        } catch (FileNotFoundException fnfe){
                                  fnfe.printStackTrace();
                        } catch (IOException ioe){
                                  ioe.printStackTrace();
                        }

        }
        /**
         * Método que calcula la frecuencia de cada palabra del fichero
         */
        private void calcularProbabilidades(){
                Iterator<?> it = apariciones.entrySet().iterator();

                float total=0;

                while (it.hasNext()) {
                        Map.Entry e = (Map.Entry)it.next();

                        float numero= Float.parseFloat(e.getValue().toString());
                        double frecuenciaActual=(numero/numPalabras);
                        total+=frecuenciaActual;
                        frecuencias.put( e.getKey().toString(),frecuenciaActual );

                        }



        }

        private void calcularProbabilidadesIDF(){
                Iterator<?> it = aparicionesSentencia.entrySet().iterator();

                float total=0;

                while (it.hasNext()) {
                        Map.Entry e = (Map.Entry)it.next();

                        float numero= Float.parseFloat(e.getValue().toString());
                        double frecuenciaActual=(numPosts/numero);

                        frecuenciasIDF.put( (String)e.getKey(),frecuenciaActual );

                        }
        }

        /**
         * Método que calcula el peso de un tweet
         * @param tweet
         * @return
         */
```

```
	private double pesoSentencia(String tweet){
		double peso=0;

		StringTokenizer st = new StringTokenizer(tweet);
		String frase="";
		while (st.hasMoreTokens()) {
			String palabra = st.nextToken();
			palabra=palabra.toLowerCase();

			if(!frase.contains(palabra)){
				peso+=pesoPalabra(palabra);
				frase+=palabra+" ";
			}
		}

		return peso;

	}
	/*
	 * Método que calcula el peso de una palabra dada la fórmula
	 * W(wi) =tf(w)* log2(idf(wi))
	 */
	private double pesoPalabra(String palabra){
		if(stopword.isStopword(palabra))
			return 0;

		double idf=frecuenciaIDF(palabra);
		double tf= frecuenciaPalabra(palabra);
		double peso=0;
		peso=tf*(Math.log(idf) / Math.log(2));

		return peso;


	}

	/**
	 * Método que calcula la frecuencia de una palabra
	 *
	 * @param palabra
	 * @return
	 */
	private double frecuenciaPalabra(String palabra){
		return frecuencias.get(palabra);


	}

	/**
	 * Método que calcula la frecuencia inversa de una palabra
	 * nº de sentencias/sentencias en las  que la palabra ocurre.
	 * @param palabra
	 * @return
	 */
	private double frecuenciaIDF(String palabra){
		return frecuenciasIDF.get(palabra);

	}


	private void tweetsTop(Tweet t){
		if(t.getNumPalabras()>5){

			if(resumenes.size()<20&&!resumenes.contains(t)){
				resumenes.add(t);

			}
			else{
				Collections.sort(resumenes);
				if(resumenes.get(0).compareTo(t)==-
1&&!resumenes.contains(t)){
					resumenes.remove(0);
					resumenes.add(t);
				}
			}
		}
	}
```

```
	private void pesarTweets(String ruta){

		String linea="";

		Tweet t;
		try{
			FileReader fr = new FileReader(ruta);
			BufferedReader bf = new BufferedReader(fr);

			while((linea = bf.readLine())!=null){

				String [] campo= linea.split(";");

				String tweet=campo[1];


				float peso=(float)pesoSentencia(tweet);

				t=new Tweet(linea);
				t.setPeso(peso);
				tweetsTop(t);

			}
		} catch (FileNotFoundException fnfe){
			 fnfe.printStackTrace();
		} catch (IOException ioe){
			 ioe.printStackTrace();
		}
	}


	public List<Tweet> getResumenes() {
		return resumenes;
	}


	public void setResumenes(List<Tweet> resumenes) {
		this.resumenes = resumenes;
	}




}
```

1.2.2 Paquete AlgoritmoRandom:

1.2.2.1 Fichero "AlgoritmoRandom.java":

```
package impl.miw.AlgoritmoRandom;

import impl.miw.entidad.Tweet;
import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;


import com.miw.business.AlgoritmoRandomManagerService;

public class AlgoritmoRandom {
	//Lista con los resúmenes

	private List <Tweet> resumenes;
	public AlgoritmoRandom() {
		resumenes=new ArrayList<Tweet>();
	}

	/**
	 * Metodo que ejecuta todo el proceso para conseguir los resumenes
	 * @param ruta
	 */
	public void ejecutarAlgoritmo(String ruta){
		String res="";
		for (int i=0;resumenes.size()<20;i++){
			res= getResumen(ruta);

			Tweet t=new Tweet(res);
			if(t!=null&&!resumenes.contains(t)){
				resumenes.add(t);
			}
		}

		Iterator<Tweet> it =resumenes.iterator();

		while(it.hasNext()){
			Tweet tweet= (Tweet)it.next();
			System.out.println("ID= "+tweet.getId()+", RESUMEN=
"+tweet.getTweet());

		}

	}
	/**
	 * Metodo que devuelve un resumen aleatorio del fichero
	 */
	@Override
	public String getResumen(String ruta) {
		int numLineas= numLineas(ruta);
		int aleatorio=(int) Math.floor(Math.random()*numLineas+1);
		String frase= getLinea(aleatorio, ruta);

		return frase;
	}


	/**
	 * Metodo que devuelve el numero de linea que se le pasa como parametro de la
ruta que se le pide
	 * @param numLinea numero de linea a devolver
	 * @param ruta del fichero donde se encuentra la linea
	 * @return
	 */
	public String getLinea(int numLinea, String ruta){
		String linea="";
```

```
                try{
                        FileReader fr = new FileReader(ruta);
                        BufferedReader bf = new BufferedReader(fr);

                        int lNumeroLineas = 0;


                        while(lNumeroLineas!=numLinea&&linea!=null){
                                linea = bf.readLine();
                                 lNumeroLineas++;
                        }
                        fr.close();
                        return linea;
                } catch (FileNotFoundException fnfe){
                         fnfe.printStackTrace();
                } catch (IOException ioe){
                         ioe.printStackTrace();
                }
                return "";
        }
        /**
         * Método que devuelve el número de lineas del fichero de entrada
         * @param ruta
         * @return
         */
        private int numLineas(String ruta){
                try{
                        FileReader fr = new FileReader(ruta);
                        BufferedReader bf = new BufferedReader(fr);

                        int lNumeroLineas = 0;
                        String sCadena="";
                        while ((sCadena = bf.readLine())!=null) {
                          lNumeroLineas++;


                        }
                        fr.close();
                        return lNumeroLineas;
                          //Código
                } catch (FileNotFoundException fnfe){
                         fnfe.printStackTrace();
                } catch (IOException ioe){
                         ioe.printStackTrace();
                }
                return 0;

        }

        /**
         * Método que devuelve la lista de resúmenes
         * @return
         */
        public List<Tweet> getResumenes() {
                return resumenes;
        }
        /**
         * Método que asigna un valor a la lista de resúmenes
         * @param resumenes
         */
        public void setResumenes(List<Tweet> resumenes) {
                this.resumenes = resumenes;
        }

}
```

1.2.3 Paquete AlgoritmoSumBasic:

1.2.3.1 Fichero "AlgoritmoSumBasic.java":

```
package impl.miw.AlgoritmoRandom;

import impl.miw.entidad.Tweet;
import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import com.miw.business.AlgoritmoRandomManagerService;

public class AlgoritmoRandom implements {
	//Lista con los resúmenes

	private List <Tweet> resumenes;


	public AlgoritmoRandom() {


		resumenes=new ArrayList<Tweet>();
	}

	/**
	 * Metodo que ejecuta todo el proceso para conseguir los resumenes
	 * @param ruta
	 */
	public void ejecutarAlgoritmo(String ruta){
		String res="";
		for (int i=0;resumenes.size()<20;i++){
			res= getResumen(ruta);

			Tweet t=new Tweet(res);
			if(t!=null&&!resumenes.contains(t)){
				resumenes.add(t);
			}
		}

		Iterator<Tweet> it =resumenes.iterator();

		while(it.hasNext()){
			Tweet tweet= (Tweet)it.next();
			System.out.println("ID= "+tweet.getId()+", RESUMEN=
"+tweet.getTweet());

		}

	}
	/**
	 * Metodo que devuelve un resumen aleatorio del fichero
	 */
	@Override
	public String getResumen(String ruta) {
		int numLineas= numLineas(ruta);
		int aleatorio=(int) Math.floor(Math.random()*numLineas+1);
		String frase= getLinea(aleatorio, ruta);

		return frase;
	}


	/**
	 * Metodo que devuelve el numero de linea que se le pasa como parametro de la
ruta que se le pide
	 * @param numLinea numero de linea a devolver
	 * @param ruta del fichero donde se encuentra la linea
	 * @return
	 */
```

```
	public String getLinea(int numLinea, String ruta){
		String linea="";

		try{
			FileReader fr = new FileReader(ruta);
			BufferedReader bf = new BufferedReader(fr);

			int lNumeroLineas = 0;


			while(lNumeroLineas!=numLinea&&linea!=null){
				linea = bf.readLine();
				 lNumeroLineas++;
			}
			fr.close();
			return linea;
		} catch (FileNotFoundException fnfe){
			  fnfe.printStackTrace();
		} catch (IOException ioe){
			  ioe.printStackTrace();
		}
		return "";
	}
	/**
	 * Método que devuelve el número de lineas del fichero de entrada
	 * @param ruta
	 * @return
	 */
	private int numLineas(String ruta){
		try{
			FileReader fr = new FileReader(ruta);
			BufferedReader bf = new BufferedReader(fr);

			int lNumeroLineas = 0;
			String sCadena="";
			while ((sCadena = bf.readLine())!=null) {
			  lNumeroLineas++;


			}
			fr.close();
			return lNumeroLineas;
			  //Código
		} catch (FileNotFoundException fnfe){
			  fnfe.printStackTrace();
		} catch (IOException ioe){
			  ioe.printStackTrace();
		}
		return 0;

	}
	/**
	 * Método que devuelve la lista de resúmenes
	 * @return
	 */
	public List<Tweet> getResumenes() {
		return resumenes;
	}
	/**
	 * Método que asigna un valor a la lista de resúmenes
	 * @param resumenes
	 */
	public void setResumenes(List<Tweet> resumenes) {
		this.resumenes = resumenes;
	}

}
```

1.2.4 Paquete impl.miw

1.2.4.1 Fichero "Principal.java"

```
package impl.miw;


import impl.miw.AlgoritmoHybrid.AlgoritmoHybridData;
import impl.miw.AlgoritmoRandom.AlgoritmoRandomData;
import impl.miw.AlgoritmoSumBasic.AlgoritmoSumBasicData;
import impl.miw.AlgoritmoTextRankKeywords.AlgoritmoTextRankKeywordsData;
import impl.miw.entidad.Tweet;
import java.io.BufferedReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.util.Iterator;
import java.util.List;

import com.miw.business.AlgoritmoHybridManagerService;
import com.miw.business.AlgoritmoManagerTextRankService;
import com.miw.business.AlgoritmoRandomManagerService;
import com.miw.business.AlgoritmoSumBasicManagerService;




public class Principal {

	/**
	 * Método que muestra el menú principal
	 */
	private void mostrarMenuTematica(){

		System.out.println("**********MENÚ**********");
		System.out.println("1.-Elegir TToppic a resumir");
		System.out.println("2.-Prueba individual de algoritmos");
		System.out.println("3.-Salir");
		System.out.println("************************");
	}

	/**
	 * Método que procesa el menú principal
	 * @throws NumberFormatException
	 * @throws IOException
	 */
	public void procesarMenuTematica() throws NumberFormatException, IOException{

		InputStreamReader isr = new InputStreamReader(System.in);
		BufferedReader br = new BufferedReader (isr);
		String rutaFicheroFiltrado="";

		mostrarMenuTematica();
		int opcion=Integer.parseInt(br.readLine());

		switch(opcion){
			case 1:
				mostrarTemas();
				opcion=Integer.parseInt(br.readLine());
				System.out.println("¿Desea usar Filtro?(S/N)");
				char filtro=br.readLine().charAt(0);
				procesarMenuTemas(opcion,filtro);

				break;
			case 2:
				mostrarMenu();
				opcion=Integer.parseInt(br.readLine());
				procesarMenu( opcion,rutaFicheroFiltrado);
```

```
                        break;
                    default:
                        break;



                }

        }
        /**
         * Método que muestra el submenu para cada algoritmo
         */
        private void mostrarMenu(){
                System.out.println("**********MENÚ PRUEBA DE ALGORITMOS**********");
                System.out.println("1.-Ejecución Algoritmo RANDOM");
                System.out.println("2.-Ejecución Algoritmo HYBRID-TFIDF");
                System.out.println("3.-Ejecución Algoritmo SUMBASIC");
                System.out.println("4.-Ejecución Algoritmo TEXTRANK_KEYWORDS");
                System.out.println("6.-Salir");
                System.out.println("*******************************************");
        }
        /**
         * Método que procesa el Submenu para cada algoritmo
         * @param opcion
         * @param rutaFicheroFiltrado
         */
private void procesarMenu(int opcion,String rutaFicheroFiltrado) {
                String topic="";
                switch(opcion){

                case 1:
                        System.out.println("1.-Ejecución Algoritmo RANDOM");
                        algoritmoRandom(rutaFicheroFiltrado,topic);
                        //verFicheros();
                        break;
                case 2:
                        System.out.println("2.-Ejecución Algoritmo HYBRID-TFIDF");
                        algoritmoHybrid(rutaFicheroFiltrado,topic);
                        break;
                case 3:
                        System.out.println("3.-Ejecución Algoritmo SUMBASIC");
                        algoritmoSumBasic(rutaFicheroFiltrado,topic);
                        break;

                case 4:
                        System.out.println("5.-Ejecución Algoritmo TEXTRANK_KEYWORDS");
                //      algoritmoTextRank(rutaFicheroFiltrado);
                        algoritmoTextRankKeywords(rutaFicheroFiltrado,topic);
                        break;
                default:
                        System.exit(0);
                        break;

                }
        }
        /**
         * Método que muestra los temas a resumir
         */
        private void mostrarTemas(){
                System.out.println("**********TEMAS DISPONIBLES**********");
                System.out.println("1.-#abdicacion");
                System.out.println("2.-#APorLaTerceraRepublica");
                System.out.println("3.-#ElReyAbdica");
                System.out.println("4.-#FelipeVI");
                System.out.println("5.-#IIIRepublica");
                System.out.println("6.-#TATGranada14");
                System.out.println("7.-#TEDxGijon");
                System.out.println("8.-#VamosRafa");
                System.out.println("9.-Brasil-Panamá");
                System.out.println("10.-Selectividad");
                System.out.println("11.-Felipe VI");
                System.out.println("12.-Volver al menú principal");
                System.out.println("**********************");


        }
        /**
```

```
         * Método que genera todos los resúmenes de un trending topic
         * @param rutaFicheroFiltrado
         * @param topic
         */
        private void generarResumenes(String rutaFicheroFiltrado,String topic){
                algoritmoRandom(rutaFicheroFiltrado,topic);
                algoritmoHybrid(rutaFicheroFiltrado,topic);
                algoritmoSumBasic(rutaFicheroFiltrado,topic);
                algoritmoTextRankKeywords(rutaFicheroFiltrado,topic);


        }
        /**
         * Método que procesa el menú de cada tema individualmente
         * @param opcion
         * @param filtro
         */
private void  procesarMenuTemas(int opcion,char filtro) {
                boolean filtrar=false;
                if(filtro=='S'||filtro=='s'){
                        filtrar=true;
                        System.out.println("entra aki");
                }
                String rutaFicheroFiltrado="";
                FiltroFichero ff;
                String ruta="Tweets\\";
                switch(opcion){


                case 1:
                        System.out.println("1.-Generando resúmenes #abdicacion");
                        ff=new FiltroFichero(ruta+"#abdicacion","es",filtrar);
                        rutaFicheroFiltrado=ff.getRuta();
                        generarResumenes(rutaFicheroFiltrado,"#abdicacion");

                        break;
                case 2:
                        System.out.println("2.-Generando resúmenes
#APorLaTerceraRepublica");
                        ff=new FiltroFichero(ruta+"#APorLaTerceraRepublica","es",filtrar);
                        rutaFicheroFiltrado=ff.getRuta();
                        generarResumenes(rutaFicheroFiltrado,"#APorLaTerceraRepublica");
                        break;
                case 3:
                        System.out.println("3.-Generando resúmenes #ElReyAbdica");
                        ff=new FiltroFichero(ruta+"#ElReyAbdica","es",filtrar);
                        rutaFicheroFiltrado=ff.getRuta();
                        generarResumenes(rutaFicheroFiltrado,"#ElReyAbdica");
                        break;
                case 4:
                        System.out.println("4.-Generando resúmenes #FelipeVI");
                        ff=new FiltroFichero(ruta+"#FelipeVI","es",filtrar);
                        rutaFicheroFiltrado=ff.getRuta();
                        generarResumenes(rutaFicheroFiltrado,"#FelipeVI");
                        break;
                case 5:
                        System.out.println("5.-Generando resúmenes #IIIRepublica");
                        ff=new FiltroFichero(ruta+"#IIIRepublica","es",filtrar);
                        rutaFicheroFiltrado=ff.getRuta();
                        generarResumenes(rutaFicheroFiltrado,"#IIIRepublica");
                        break;
                case 6:
                        System.out.println("6.-Generando resúmenes #TATGranada14");
                        ff=new FiltroFichero(ruta+"#TATGranada14","es",filtrar);
                        rutaFicheroFiltrado=ff.getRuta();
                        generarResumenes(rutaFicheroFiltrado,"#TATGranada14");
                        break;
                case 7:
                        System.out.println("7.-Generando resúmenes #TEDxGijon");
                        ff=new FiltroFichero(ruta+"#TEDxGijon","es",filtrar);
                        rutaFicheroFiltrado=ff.getRuta();
                        generarResumenes(rutaFicheroFiltrado,"#TEDxGijon");
                        break;
                case 8:
                        System.out.println("8.-Generando resúmenes #VamosRafa");
                        ff=new FiltroFichero(ruta+"#VamosRafa","es",filtrar);
                        rutaFicheroFiltrado=ff.getRuta();
```

```
                        generarResumenes(rutaFicheroFiltrado,"#VamosRafa");
                        break;
                case 9:
                        System.out.println("9.-Generando resúmenes Brasil-Panamá");
                        ff=new FiltroFichero(ruta+"Brasil-Panamá","es",filtrar);
                        rutaFicheroFiltrado=ff.getRuta();
                        generarResumenes(rutaFicheroFiltrado,"Brasil-Panamá");
                        break;
                case 10:
                        System.out.println("10.-Generando resúmenes Selectividad");
                        ff=new FiltroFichero(ruta+"Selectividad","es",filtrar);
                        rutaFicheroFiltrado=ff.getRuta();
                        generarResumenes(rutaFicheroFiltrado,"Selectividad");
                        break;
                case 11:
                        System.out.println("10.-Generando resúmenes Felipe VI");
                        ff=new FiltroFichero(ruta+"Felipe VI","es",filtrar);
                        rutaFicheroFiltrado=ff.getRuta();
                        generarResumenes(rutaFicheroFiltrado,"Felipe VI");
                        break;
                default:

                        break;
                }
                }



        /**
         * Método que guarda un fichero con los resumenes de la lista que se le pasa como
parametro
         * @param lista
         * @param ruta
         */
        private void guardarResumen(List <Tweet> lista,String ruta){


                        FileWriter fichero = null;
                PrintWriter pw = null;
                try
                {

                    fichero = new FileWriter(ruta,true);

                    pw = new PrintWriter(fichero);
                    Iterator<Tweet> it=lista.iterator();

                    while(it.hasNext()){
                       Tweet t= it.next();
                       String linea="";
                       if(t!=null){
                               linea=t.getId()+","+t.getTweet();
                               pw.println(linea);
                       }

                    //Se escribe la línea en el fichero

                    }


                } catch (Exception e) {
                    e.printStackTrace();
                } finally {
                   try {
                   // Nuevamente aprovechamos el finally para
                   // asegurarnos que se cierra el fichero.
                   if (null != fichero)
                      fichero.close();
                   } catch (Exception e2) {
                      e2.printStackTrace();
                   }
                }

                }


        private void algoritmoRandom(String ruta,String topic){
```

```
                AlgoritmoRandomManagerService ar;
                System.out.println("RESUMENES RANDOM:");

                ar= new AlgoritmoRandomData(ruta);
                String rutaResumen="resumen\\"+topic+"AlgoritmoRandom.txt";
                guardarResumen(ar.getResumenes(),rutaResumen);



        }
        private void algoritmoHybrid(String ruta,String topic){

                AlgoritmoHybridManagerService ah= new AlgoritmoHybridData(ruta);

                String rutaResumen="resumen\\"+topic+"AlgoritmoHybrid.txt";
                guardarResumen(ah.getResumenes(),rutaResumen);

        }
        private void algoritmoSumBasic(String ruta,String topic){
                AlgoritmoSumBasicManagerService asb= new AlgoritmoSumBasicData(ruta);

                String rutaResumen="resumen\\"+topic+"AlgoritmoSumBasic.txt";
                guardarResumen(asb.getResumenes(),rutaResumen);
        }


        private void algoritmoTextRankKeywords(String ruta,String topic){
                System.out.println("RESUMENES TEXTRANK:");
                AlgoritmoManagerTextRankService atrkw =new
AlgoritmoTextRankKeywordsData(ruta);

                String rutaResumen="resumen\\"+topic+"AlgoritmoTextRank.txt";
                guardarResumen(atrkw.getResumenes(),rutaResumen);

        }

        /**
         * @param args
         */
        public static void main(String[] args) {

                InputStreamReader isr = new InputStreamReader(System.in);
                BufferedReader br = new BufferedReader (isr);

                try {
                        Principal p=new Principal();

                        String ruta2="#APorLaTerceraRepublica.txt";
                        String
nombre3="D:\\entorno\\proyectos\\Proyecto\\notas\\20140602151659-#IIIRepublica.txt";

                        while(true){

                                p.procesarMenuTematica();


                        }


                } catch (IOException e) {
                        System.out.println("Error de lectura escritura");
                        e.printStackTrace();
                }


        }

}
```

1.2.5 Paquete AlgoritmoTextRankKeywords:

1.2.5.1 Fichero "AlgoritmoTextRankKeywords.java":

```
package impl.miw.AlgoritmoTextRankKeywords;

import impl.miw.entidad.StopWord;
import impl.miw.entidad.Tweet;
import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Iterator;
import java.util.List;
import java.util.Map;
import java.util.Map.Entry;
import java.util.Set;

public class AlgoritmoTextRankKeywords {


	private GrafoRank grafo;

	//Idioma en el que se va aplicar el algoritmo
	private String idioma;

	private StopWord stopword;

	//Lista con las palabras clave
	//private List <Nodo> keywords;

	//Variable Dumping del algoritmo
	private static final float DUMPING=0.85f;

	//Umbral definido como punto donde existe convergencia
	private static final float THRESHOLD=0.0001f;

	//Lista con los resúmenes
	private List <Tweet> resumenes;

	public AlgoritmoTextRankKeywords(){
		idioma="es";
		stopword= new StopWord(idioma);
		grafo=new GrafoRank();
		resumenes=new ArrayList<Tweet>();
	}
	/**
	 * Método utilizado para la ejecución del algoritmo, pasándole la ruta como
parámetro del fichero
	 * a procesar
	 * @param ruta
	 */
	public void ejecutarAlgoritmo(String ruta){

		//Se genera el grafo
		cargarTweets(ruta);

		//Calculo de Puntuaciones
		calculoPuntuacion();

		pesarTweets(ruta);

		System.out.println("Resumenes");
		System.out.println(resumenes.toString());

	}

	//Construcción del Grafo a partir de la carga de Tweets
	/**
	 * Método que procesa los tweets de la ruta que se le pasa como parámetro y que
genera el grafo de palabras
```

```
         */
                private void cargarTweets(String ruta){
                String linea="";

                try{
                        FileReader fr = new FileReader(ruta);
                        BufferedReader bf = new BufferedReader(fr);

                        while((linea= bf.readLine())!=null){

                                linea=linea.toLowerCase();

                                String []campo= linea.split(";");
                                procesarFrase(campo[1]);
                        }
                        fr.close();

                } catch (FileNotFoundException fnfe){
                         fnfe.printStackTrace();
                } catch (IOException ioe){
                         ioe.printStackTrace();
                }

}

        /**
         * Método que genera un array de strings sin stopwords del array de strings que
se le pasa como parámetro
         * @param palabras
         * @return
         */
        private String [] palabrasSinStopwords(String [] palabras){
                String palabrasSW []=null;
                String cadena="";

                for (int i=0; i<palabras.length;i++){
                        if(!stopword.isStopword(palabras[i])){
                                cadena+=palabras[i]+" ";
                        }
                }
                palabrasSW=cadena.split(" ");

                return palabrasSW;
        }

        /**
         * Método que procesa la frase que se pasa como parámetro y que añade como Nodos
         * al grafo las palabras de dicha frase así como sus conexiones entre ellos
         * @param frase
         */
        private void procesarFrase(String frase){
                String palabras []=frase.split(" ");
                palabras=palabrasSinStopwords(palabras);
                Nodo actual=null;
                Nodo ant=null;
                Nodo sig=null;
                if(palabras!=null){
                        for (int i=0; i<palabras.length;i++){
                                actual= new Nodo(palabras[i]);

                                if(palabras.length>2){
                                        if(i==0){
                                                ant=null;
                                                sig=new Nodo(palabras[i+1]);
                                                actual.conectarNodo(sig);


                                        }else if (i==(palabras.length-1)){
                                                sig=null;
                                                ant=new Nodo(palabras[i-1]);
                                                actual.conectarNodo(ant);


                                        }else{
                                                ant=new Nodo(palabras[i-1]);
                                                sig=new Nodo(palabras[i+1]);
                                                actual.conectarNodo(ant);
```

```
                                        actual.conectarNodo(sig);

                                }
                        }

                        grafo.anadirNodo(actual);

                }
        }
}

/**
 * Método que calcula la puntuación de cada Nodo hasta que se produzca
convergencia
 */
private void calculoPuntuacion(){
        boolean convergencia=false;
        while(!convergencia){
                Map <String, Nodo> nodos=grafo.getGrafo();
                Iterator<Entry<String, Nodo>> it = nodos.entrySet().iterator();
                System.out.println("Calculo de probabilidades");
                double puntuacion=0;


                while (it.hasNext()) {

                        Map.Entry e = (Map.Entry)it.next();
                        Nodo actual=(Nodo)e.getValue();
                        actual.setPuntuacionAnterior(actual.getPuntuacion());
                        puntuacion=(1-DUMPING)*DUMPING;

puntuacion*=calculoSumatorioConexiones(actual.getConexiones());
                        if(puntuacion<THRESHOLD)
                                convergencia=true;
                        actual.setPuntuacion(puntuacion);
                        grafo.modificarNodo(actual);
                        //Vi=(1-d)*d SUM(1/|Vout|)*Vj

                }

        }


}

/**
 * Método que calcula el sumatorio del conjunto de nodos que se pasa como
 *  parámetro necesario en la formula de TextRank
 * @param nodos
 * @return
 */
private double calculoSumatorioConexiones(Set<Nodo> nodos){
        double sumatorio=0;
        if(nodos!=null){
                Iterator<Nodo> it = nodos.iterator();

                while (it.hasNext()) {
                        Nodo e =it.next();
                        Nodo g=grafo.getNodo(e.getTexto());
                        double numConexiones=g.getNumConexiones();
                        sumatorio+=(1/numConexiones)*g.getPuntuacionAnterior();

                }
        }
        return sumatorio;
}



/**
 * Método que va generando la lista de los resumenes que tengan mas peso
 * @param t
 */
private void tweetsTop(Tweet t){
        if(t.getNumPalabras2()>5){
                if(resumenes.size()<20&&!resumenes.contains(t)){
                        resumenes.add(t);
```

```
                        }
                        else{
                                Collections.sort(resumenes);
                                if(resumenes.get(0).compareTo(t)==-
1&&!resumenes.contains(t)){
                                        resumenes.remove(0);
                                        resumenes.add(t);

                                }
                        }
                }



        }



        /**
         * Método que pesa todos los tweets del archivo cuya ruta se le pasa como
parámetro
         * @param ruta
         */
        private void pesarTweets(String ruta){

                String linea="";

                Tweet t;
                try{
                        FileReader fr = new FileReader(ruta);
                        BufferedReader bf = new BufferedReader(fr);

                        while((linea = bf.readLine())!=null){

                                t=new Tweet(linea);

                                t.pesarTweetKeywords(grafo.getGrafo());

                                tweetsTop(t);

                        }

                        fr.close();

                } catch (FileNotFoundException fnfe){
                         fnfe.printStackTrace();
                } catch (IOException ioe){
                         ioe.printStackTrace();
                }

        }

        /**
         * Método que devuelve la lista de resúmenes
         * @return
         */
        public List<Tweet> getResumenes() {
                return resumenes;
        }
        /**
         * Método que asigna a la lista de resúmenes un valor determinado
         * @param resumenes
         */
        public void setResumenes(List<Tweet> resumenes) {
                this.resumenes = resumenes;
        }

}
```

1.2.5.2 Fichero "GrafoRank.java":

```
package impl.miw.AlgoritmoTextRankKeywords;
```

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.HashMap;
import java.util.Iterator;
import java.util.List;
import java.util.Map;
import java.util.Map.Entry;

public class GrafoRank {

	private Map <String, Nodo> grafo;



	public GrafoRank(){

		grafo=new HashMap<String,Nodo>();

	}

	public Map<String, Nodo> getGrafo() {
		return grafo;
	}

	/**
	 * Metodo utilizado para añadir un nodo al grafo
	 * @param n
	 */
	public void anadirNodo(Nodo n){

			//Si existe el Nodo en el grafo
		if(n!=null){
			if(grafo.containsKey(n.getTexto())){
				Nodo old=grafo.get(n.getTexto());

				old.combinarConexiones(n);
				//System.out.println("entra aki "+n.getTexto());

				grafo.put(n.getTexto(), old);
			}else{
				grafo.put(n.getTexto(),n);
			}

		}



	}
	/**
	 * Metodo que devuelve el nodo cuya clave se le pasa como parámetro
	 * @param key
	 * @return
	 */
	public Nodo getNodo(String key){
		if(key!=null){
			return grafo.get(key);


		}else return null;

	}

	/**
	 * Método que ordena los elementos del grafo
	 */
	public void ordenar(){

		Iterator<Entry<String, Nodo>> it = grafo.entrySet().iterator();
		List <Nodo> keywords=new ArrayList<Nodo>();
		while (it.hasNext()) {
			Map.Entry e = (Map.Entry)it.next();
			Nodo actual=(Nodo)e.getValue();
			if(keywords.size()<10){
				keywords.add(actual);

			}
			else{
```

```
                                Collections.sort(keywords);
                                if(keywords.get(0).compareTo(actual)==-
1&&!keywords.contains(actual)){
                                        keywords.remove(0);
                                        keywords.add(actual);

                                }
                        }

                }



        }
        /**
         * Método que modifica el nodo que se le pasa como parámetro del grafo
         * @param n
         */
        public void modificarNodo (Nodo n){
                if(n!=null&&grafo.containsKey(n.getTexto())){
                        grafo.put(n.getTexto(),n);
                }

        }

        /**
         * Método que borra el nodo que se le pasa como parámetro del grafo
         * @param n
         */
        public void borrarNodo(Nodo n){
                if(n!=null){
                        grafo.remove(n.getTexto());

                }


        }

        /**
         * Método que devuelve en una cadena los componentes del grafo
         */
        @Override
        public String toString() {
                return "GrafoRank [grafo=" + grafo + "]";
        }


}
```

1.2.5.3 Fichero "Nodo.java":

```
package impl.miw.AlgoritmoTextRankKeywords;

import java.util.HashSet;
import java.util.Iterator;
import java.util.Set;

public class Nodo implements Comparable<Nodo>{

	private String texto;

	private double puntuacion;

	private double puntuacionAnterior;

	private Set<Nodo> conexiones;

	public Nodo(String texto){
		this.setTexto(texto);
		puntuacion=1;
		puntuacionAnterior=1;
		conexiones=new HashSet<Nodo>();

	}

	/**
	 * Método que añade una conexion con el nodo que se le pasa como parámetro al propio nodo
	 * @param nuevo
	 */
	public void conectarNodo(Nodo nuevo){

		if(nuevo!=null&&!conexiones.contains(nuevo))
			conexiones.add(nuevo);

	}
	/**
	 * Método que combina las conexiones del nodo que se le pasa como parámetro con el propio nodo
	 * @param nuevo
	 */
	public void combinarConexiones(Nodo nuevo){
		if(nuevo!=null&&texto.compareToIgnoreCase(nuevo.getTexto())==0){
			Set<Nodo> conexionesNuevo=nuevo.getConexiones();

			Iterator<Nodo> it = conexionesNuevo.iterator();
			while (it.hasNext()) {
				Nodo e =it.next();
				//System.out.println("   Combina "+e.getTexto());
				conectarNodo(e);
			}


		}

	}
	/**
	 * Método que devuelve el número de conexiones que tiene un nodo
	 * @return
	 */
	public int getNumConexiones(){

		return conexiones.size();
	}
	/**
	 * Método que desconecta el nodo que se le pasa como parámetro del propio nodo
	 * @param n
	 */
	public void desconectarNodo(Nodo n){
		conexiones.remove(n);
```

```
	}

/**
 * Método que devuelve la puntuacion anterior del nodo
 * @return
 */
	public double getPuntuacionAnterior() {
		return puntuacionAnterior;
	}

/**
 * Método que fija la puntuacionAnterior con el parámetro que se le pasa al metodo
 * @param puntuacionAnterior
 */
	public void setPuntuacionAnterior(double puntuacionAnterior) {
		this.puntuacionAnterior = puntuacionAnterior;
	}

/***
 * Método que devuelve las conexiones del Nodo
 * @return
 */
	public Set<Nodo> getConexiones() {
		return conexiones;
	}

/**
 * Método que fija las conexiones del propio nodo a un valor que se le pasa como
parámetro
 * @param conexiones
 */
	public void setConexiones(Set<Nodo> conexiones) {
		this.conexiones = conexiones;
	}

/**
 * Método que fija el texto del nodo al elemento que se le pasa como parámetro
 * @param texto
 */
	public void setTexto(String texto) {
		this.texto = texto;
	}

	/**
	 * Método que devuelve el texto del nodo
	 * @return
	 */
	public String getTexto() {
		return texto;
	}
/**
 * Método que fija la puntuacion del propio nodo con el valor que se le pasa como
parametro
 * @param puntuacion
 */
	public void setPuntuacion(double puntuacion) {
		this.puntuacion = puntuacion;
	}
/**
 * Método que devuelve la puntuación del nodo
 * @return
 */
	public double getPuntuacion() {
		return puntuacion;
	}


	@Override
	public String toString() {
		return "Nodo [texto=" + texto + ", puntuacion=" + puntuacion+"]\n";
	}


	@Override
	public int hashCode() {
		final int prime = 31;
```

```
                int result = 1;
                result = prime * result + ((texto == null) ? 0 : texto.hashCode());
                return result;
        }


        @Override
        public boolean equals(Object obj) {
                if (this == obj)
                        return true;
                if (obj == null)
                        return false;
                if (getClass() != obj.getClass())
                        return false;
                Nodo other = (Nodo) obj;
                if (texto == null) {
                        if (other.texto != null)
                                return false;
                } else if (!texto.equals(other.texto))
                        return false;
                return true;
        }


        /**
         * Método utilizado para comparar un nodo con otro
         */
        @Override
        public int compareTo(Nodo n) {
                 if (this.puntuacion > n.puntuacion) {
                            return 1;
                        }
                        else if (this.puntuacion < n.puntuacion) {
                            return -1;
                        }
                        else {
                            return 0;//this.texto.compareTo(n.texto);
                        }
        }

}
```

1.2.6 Paquete impl.miw

1.2.6.1 Fichero "FiltroFichero.java"

```
import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.StringTokenizer;

import org.apache.commons.validator.UrlValidator;

import net.sf.json.JSONObject;
import net.sf.json.JSONSerializer;

public class FiltroFichero {

	//Ruta del fichero de salida que generará el filtro con los tweets limpios
	private String ruta;
	private String idioma;
	private String rutafinal;

	/**
	 * Método constructor del Filtro
	 * @param ruta Parámetro donde se encuentra la ruta del fichero a filtrar
	 */
	public FiltroFichero(String ruta,String idioma,boolean filtrar){
		this.idioma=idioma;
		String s[]=ruta.split("\\\\");
		int numCarpetas=s.length;
		this.rutafinal="Filtrado\\"+s[numCarpetas-1]+".txt";
		if(filtrar)
			verFicheros(ruta);
		//procesarFichero(ruta);



	}


	private void verFicheros(String ruta){

		String sDirectorio = ruta;
		System.out.println(ruta);
		File f = new File(sDirectorio);
		if (f.exists()){ // Directorio existe
			File[] ficheros = f.listFiles();
			for (int x=0;x<ficheros.length;x++){
			  System.out.println(ficheros[x].getName());
			  procesarFichero(ruta+"\\"+ficheros[x].getName());
			}
		}

			else { //Directorio no existe



			}

	}

	/**
	   * Método que obtiene el campo text de la linea que se le pasa como parámetro
en formato json
	   * @param linea
	   * @return Devolverá el texto del Tweet en cuestión
```

```
         */
        private String []parsearLinea(String linea) {
                String campo[]=new String[2];
                String texto="";
                String id="";
                if(linea.compareToIgnoreCase("")==0){

                        campo[0]="";
                         campo[1]="";
                         return campo;

                }
                 JSONObject json = (JSONObject) JSONSerializer.toJSON( linea );
                 String lang=json.getString("lang");
                // System.out.println("Lang: "+lang+", Idioma: "+idioma);
                 if(lang!=null&&lang.compareToIgnoreCase(idioma)!=0){
                         campo[0]="";
                         campo[1]="";
                         return campo;
                 }
                 else{
                         texto = json.getString( "text" );
                         id=json.getString( "id" );
                         campo[0]=id;
                         campo[1]=texto;
                 return campo;
            }

        }

        /**
         * Método utilizado para el filtrado de un Tweet, eliminando URLs, términos
innecesarios de la nomenclatura utilizada por twitter
         * y también eliminará otros símbolos que no formen únicamente palabras
         * @param texto
         * @return
         */
        private String fraseFiltrada(String texto) {
                StringTokenizer st = new StringTokenizer(texto);
                String[] schemes = {"http","https"}; // DEFAULT schemes = "http",
"https", "ftp"
                UrlValidator urlValidator = new UrlValidator(schemes);

                String frase="";
                while (st.hasMoreTokens()) {
                        String palabra = st.nextToken();

                        if(urlValidator.isValid(palabra)||palabra.startsWith("http:")){

                                frase+="";
                        }else{

                                palabra=palabra.replace("(","");
                                palabra=palabra.replace(")","");
                                palabra=palabra.replace("\"","");
                                palabra=palabra.replace("“","");
                                palabra=palabra.replace("¨","");


                                if(!urlValidator.isValid(palabra)&&palabra.length()>0){


                                //se comprueba si es url, hashtag, usuario de twitter, o
retweet para eliminar la palabra

        if(/*||palabra.charAt(0)=='@'||palabra.charAt(0)=='#'||*/palabra.compareToIgnoreC
ase("RT")==0)
                                                frase+="";
                                        else{
                                                //Se reemplazan por blancos caracteres
innecesarios
                                                palabra=palabra.replace(":","");
                                                palabra=palabra.replace(".","");
                                                palabra=palabra.replace(",","");
                                                palabra=palabra.replace("…","");
                                                palabra=palabra.replace("?","");
                                                palabra=palabra.replace("¿","");
```

```
                                        palabra=palabra.replace("?","");

                                        palabra=palabra.replace("¡","");
                                        palabra=palabra.replace("!","");
                                        palabra=palabra.replace("-","");
                                        palabra=palabra.replace("_","");

                                        frase+=palabra+" ";
                                }
                        }else{
                                        //si es una url no se cuenta
                                frase+="";
                        }
                }
        }

        return frase;
    }

    /**
     * Método utilizada para el procesamiento del fichero a filtrar, donde se leerá
linea a línea el fichero
     * y se realizará el filtrado de dicha línea para escribirla ya filtrada en el
fichero que almacenará los Tweets filtrados
     * @param ruta
     */
    private void procesarFichero(String ruta){
        String linea="";
        String campo[];
        String texto="";
        try{
                FileReader fr = new FileReader(ruta);
                BufferedReader bf = new BufferedReader(fr);

                while((linea = bf.readLine())!=null){

                        //Obtención del texto del tweet
                        campo=parsearLinea(linea);


                        //Se filtra el texto del Tweet y se almacena en la variable
texto
                        texto=fraseFiltrada(campo[1]);

                        if(texto.length()>2)
                        //Se escribe la linea filtrada en el fichero
                        escribirLinea(campo[0]+";"+texto);

                        //Calculo del número de apariciones por palabra

                }
                System.out.println("*****Fin Procesado Fichero*****");

                fr.close();
        } catch (FileNotFoundException fnfe){
                 fnfe.printStackTrace();
        } catch (IOException ioe){
                 ioe.printStackTrace();
        }

    }

    /**
     * Método utilizado para escribir la linea previamente filtrada en el fichero
resultante de los Tweets procesados
     * @param linea Parámetro que contendrá la línea a almacenar en el fichero
     */
    private void escribirLinea(String linea){

        FileWriter fichero = null;
     PrintWriter pw = null;
     try
     {

         fichero = new FileWriter(rutafinal,true);

         pw = new PrintWriter(fichero);
```

```
            //Se escribe la línea en el fichero
                pw.println(linea);


         } catch (Exception e) {
             e.printStackTrace();
         } finally {
            try {
            // Nuevamente aprovechamos el finally para
            // asegurarnos que se cierra el fichero.
            if (null != fichero)
               fichero.close();
            } catch (Exception e2) {
               e2.printStackTrace();
            }
         }

        }

        /**
         * Método que retorna la ruta del fichero ya filtrado
         * @return
         */
        public String getRuta() {
                return rutafinal;
        }

}
```

1.2.7 Paquete impl.miw.entidad

1.2.7.1 Fichero "StopWord.java"

```
package impl.miw.entidad;
import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.util.Arrays;
import java.util.HashSet;
import java.util.Set;
public class StopWord {
	public static String[] stopwords;
	public static Set<String> stopWordSet ;
	private String idioma;

	public StopWord (String idioma){
		this.idioma=idioma;
		cargarStop();
		stopWordSet = new HashSet<String>(Arrays.asList(stopwords));
	}
	/**
	 * Método que comprueba si una palabra es stopWord
	 * @param word
	 * @return
	 */
	public boolean isStopword(String word) {
		word=word.toLowerCase();
		if(word.length() < 2) return true;
		if(word.charAt(0) >= '0' && word.charAt(0) <= '9') return true; //remove
numbers, "25th", etc
		if(word.charAt(0)=='#'||word.charAt(0)=='@') return true;
		if(word.startsWith("http:"))return true;
		if(stopWordSet.contains(word)) return true;
	//	if(word.matches("\\b(https?|ftp|file)://[-A-Z0-9+&@#/%?=~_|!:,.;]*[-A-Z0-
9+&@#/%=~_|]")) return true;
		else return false;
	}
	/**
	 * Método que carga el array de stopwords en castellano
	 */
	private void cargarStop(){
		String linea="";
		String texto="";
		String fichero;
		try{

			if(idioma.compareToIgnoreCase("es")==0)
					fichero="util\\stopwords-es.txt";
			else
				fichero="util\\stopwords-en.txt";
			FileReader fr = new FileReader(fichero);
			BufferedReader bf = new BufferedReader(fr);

			while((linea= bf.readLine())!=null){
				texto+=linea;
			}
			texto=texto.toLowerCase();
			stopwords = texto.split(",");

		} catch (FileNotFoundException fnfe){
			 fnfe.printStackTrace();
		} catch (IOException ioe){
			 ioe.printStackTrace();
		}
	}
}
```

1.2.8 Paquete impl.miw.entidad

1.2.8.1 Fichero "Tweet.java"

```
package impl.miw.entidad;

import impl.miw.AlgoritmoTextRankKeywords.Nodo;

import java.util.HashMap;
import java.util.Map;
import java.util.StringTokenizer;


public class Tweet implements Comparable <Tweet>{
	private String tweet;
	private float peso;
	private String id;
	private int numPalabras;
	private int numPalabras2;

	//Map con las palabras y sus apariciones
	private Map<String, Integer> apariciones;

	public Tweet (String linea){
		this.tweet="";
		String [] campo= linea.split(";");
		if(campo.length==2){
			this.tweet=campo[1];
			this.id=campo[0];
			calcularNumPalabras();
		}


	}

	/**
	 * Método utilizado  para pesar Tweets
	 * @param frecuencias
	 */
	public void pesarTweet(Map<String, Double> frecuencias){
		float pesoaux=0;
		numPalabras=0;
		numPalabras2=0;
		apariciones= new HashMap<String, Integer>();
		StringTokenizer st = new StringTokenizer(tweet);

		while (st.hasMoreTokens()) {
			String palabra = st.nextToken();
			palabra=palabra.toLowerCase();
		//	System.out.println("Tweet: "+tweet);
			//System.out.println("palabra: "+palabra);
			if(frecuencias.containsKey(palabra)){
				Integer aparicion = apariciones.get(palabra);
				apariciones.put(palabra, (aparicion == null) ? 1 :
aparicion + 1);
				if(apariciones.get(palabra)<2){
					pesoaux+=frecuencias.get(palabra);
				}
			}
			numPalabras2++;


		}
		peso=pesoaux/numPalabras2;
	}
	/**
	 * Método utilizado para pesar los Tweets
	 * @param pesos
	 */

	public void pesarTweetKeywords(Map<String, Nodo> pesos ){

		float pesoaux=0;
```

```
                apariciones= new HashMap<String, Integer>();
                StringTokenizer st = new StringTokenizer(tweet);

                while (st.hasMoreTokens()) {
                        String palabra = st.nextToken();
                        palabra=palabra.toLowerCase();

                        if(pesos.containsKey(palabra)){
                                Integer aparicion = apariciones.get(palabra);
                                apariciones.put(palabra, (aparicion == null) ? 1 :
aparicion + 1);
                                if(apariciones.get(palabra)<2){
                                        Nodo n=pesos.get(palabra);
                                        if(n!=null){
                                                pesoaux+=n.getPuntuacion();
                                        }
                                }
                        }
                        numPalabras2++;


                }
                peso=pesoaux/numPalabras2;

        }

        //GETTERS Y SETTERS
        public void setId(String id) {
                this.id = id;
        }
        public String getId() {
                return id;
        }
        public void setPeso(float peso) {
                this.peso = peso;
        }
        public float getPeso() {
                return peso;
        }
        public void setTweet(String tweet) {
                this.tweet = tweet;
        }
        public String getTweet() {
                return tweet;
        }

        /**
         * Método que calcula el número de Palabras de un Tweet
         */
        private void calcularNumPalabras(){

                StringTokenizer st = new StringTokenizer(tweet);

                while (st.hasMoreTokens()) {
                        st.nextToken();
                        numPalabras++;
                }



        }
        /**
         * Método que devuelve el numero de palabras del tweet
         * @return
         */
        public int getNumPalabras() {

                return numPalabras;
        }
        /**
         * Método que devuelve el numero de palabras del tweet
         * @return
         */
        public int getNumPalabras2() {

                return numPalabras2;
        }
```

```
	/**
	 * Método que compara un Tweet con otro
	 * @return
	 */
	@Override
	public int compareTo(Tweet o) {
		float resul=this.peso-o.peso;
		if(resul>0)
			return 1;
		else if(resul<0)
			return -1;
		else
			return 0;


	}



	@Override
	public int hashCode() {
		final int prime = 31;
		int result = 1;
		result = prime * result + ((id == null) ? 0 : id.hashCode());
		result = prime * result + ((tweet == null) ? 0 : tweet.hashCode());
		return result;
	}



	@Override
	public boolean equals(Object obj) {
	//	System.out.println("Entra equals");

		if (obj == null)
			return false;

		Tweet other = (Tweet) obj;
		if (id == null) {
			if (other.id != null)
				return false;
		} else if (id.equals(other.id))
			return true;
		if (tweet == null) {
			if (other.tweet != null)
				return false;
		} else if (tweet.compareToIgnoreCase(other.tweet)!=0){
			//System.out.println(tweet+" ->"+ other.tweet);
			return false;
		}
		return true;
	}



	@Override
	public String toString() {
		return "Tweet [id="+id+" tweet=" + tweet + ", peso=" + peso + "]\n";
	}

}
```

1.2.9 Paquete impl.miw

1.2.9.1 Fichero "FiltroFichero.java"

```
package impl.miw.entidad;

import impl.miw.AlgoritmoTextRankKeywords.Nodo;

import java.util.HashMap;
import java.util.Map;
import java.util.StringTokenizer;


public class Tweet implements Comparable <Tweet>{
	private String tweet;
	private float peso;
	private String id;
	private int numPalabras;
	private int numPalabras2;

	//Map con las palabras y sus apariciones
	private Map<String, Integer> apariciones;

	public Tweet (String linea){
		this.tweet="";
		String [] campo= linea.split(";");
		if(campo.length==2){
			this.tweet=campo[1];
			this.id=campo[0];
			calcularNumPalabras();
		}


	}

	/**
	 * Método utilizado  para pesar Tweets
	 * @param frecuencias
	 */
	public void pesarTweet(Map<String, Double> frecuencias){
		float pesoaux=0;
		numPalabras=0;
		numPalabras2=0;
		apariciones= new HashMap<String, Integer>();
		StringTokenizer st = new StringTokenizer(tweet);

		while (st.hasMoreTokens()) {
			String palabra = st.nextToken();
			palabra=palabra.toLowerCase();
		//	System.out.println("Tweet: "+tweet);
			//System.out.println("palabra: "+palabra);
			if(frecuencias.containsKey(palabra)){
				Integer aparicion = apariciones.get(palabra);
				apariciones.put(palabra, (aparicion == null) ? 1 :
aparicion + 1);
				if(apariciones.get(palabra)<2){
					pesoaux+=frecuencias.get(palabra);
				}
			}
			numPalabras2++;


		}
		peso=pesoaux/numPalabras2;
	}
	/**
	 * Método utilizado para pesar los Tweets
	 * @param pesos
	 */
```

```
	public void pesarTweetKeywords(Map<String, Nodo> pesos ){

		float pesoaux=0;
		apariciones= new HashMap<String, Integer>();
		StringTokenizer st = new StringTokenizer(tweet);

		while (st.hasMoreTokens()) {
			String palabra = st.nextToken();
			palabra=palabra.toLowerCase();

			if(pesos.containsKey(palabra)){
				Integer aparicion = apariciones.get(palabra);
				apariciones.put(palabra, (aparicion == null) ? 1 :
aparicion + 1);
				if(apariciones.get(palabra)<2){
					Nodo n=pesos.get(palabra);
					if(n!=null){
						pesoaux+=n.getPuntuacion();
					}
				}
			}
			numPalabras2++;


		}
		peso=pesoaux/numPalabras2;

	}

	//GETTERS Y SETTERS
	public void setId(String id) {
		this.id = id;
	}
	public String getId() {
		return id;
	}
	public void setPeso(float peso) {
		this.peso = peso;
	}
	public float getPeso() {
		return peso;
	}
	public void setTweet(String tweet) {
		this.tweet = tweet;
	}
	public String getTweet() {
		return tweet;
	}

	/**
	 * Método que calcula el número de Palabras de un Tweet
	 */
	private void calcularNumPalabras(){

		StringTokenizer st = new StringTokenizer(tweet);

		while (st.hasMoreTokens()) {
			st.nextToken();
			numPalabras++;
		}



	}
	/**
	 * Método que devuelve el numero de palabras del tweet
	 * @return
	 */
	public int getNumPalabras() {

		return numPalabras;
	}
	/**
	 * Método que devuelve el numero de palabras del tweet
	 * @return
	 */
```

```
	public int getNumPalabras2() {

		return numPalabras2;
	}
	/**
	 * Método que compara un Tweet con otro
	 * @return
	 */
	@Override
	public int compareTo(Tweet o) {
		float resul=this.peso-o.peso;
		if(resul>0)
			return 1;
		else if(resul<0)
			return -1;
		else
			return 0;


	}



	@Override
	public int hashCode() {
		final int prime = 31;
		int result = 1;
		result = prime * result + ((id == null) ? 0 : id.hashCode());
		result = prime * result + ((tweet == null) ? 0 : tweet.hashCode());
		return result;
	}



	@Override
	public boolean equals(Object obj) {
	//	System.out.println("Entra equals");

		if (obj == null)
			return false;

		Tweet other = (Tweet) obj;
		if (id == null) {
			if (other.id != null)
				return false;
		} else if (id.equals(other.id))
			return true;
		if (tweet == null) {
			if (other.tweet != null)
				return false;
		} else if (tweet.compareToIgnoreCase(other.tweet)!=0){
			//System.out.println(tweet+" ->"+ other.tweet);
			return false;
		}
		return true;
	}



	@Override
	public String toString() {
		return "Tweet [id="+id+" tweet=" + tweet + ", peso=" + peso + "]\n";
	}

}
```

8.1.2 Script para el cálculo de similitudes entre palabras

1.2.9.2 Fichero evaluación-resumenes.php

```
<?php

// Este archivo contiene todos los tuits de los usuarios
//
$input="evaluacion-resumenes.txt";

// Este sólo contiene el primer tuitresumen de cada usuario
//
$input="evaluacion-resumenes-1tuit.txt";

$tmp=explode("\n",trim(file_get_contents($input)));

// Array para poder "vetar" usuarios en la evaluación
//
$blackList=array("ev10");

foreach ($tmp as $line) {
	$line=trim($line);
	list($topic,$summary,$evaluator)=explode("\t",$line);

	if (!in_array($evaluator,$blackList)) {
		$summaries[$topic][$evaluator][]=trim($summary);
	}
}

foreach ($summaries as $topic=>$array) {
	$evaluators=array_keys($array);
	foreach ($evaluators as $evaluator) {
		$similarity=compareEvaluatorVSothers($topic,$evaluator);
		$report[$topic][$evaluator]=$similarity;
		$allSimilarities[]=$similarity;
	}
}

foreach ($report as $topic=>$array) {
	arsort($array);
	$report[$topic]=$array;
	$avg=array_sum($array)/sizeof($array);

	foreach ($array as $user=>$score) {
		if ($score<0.9*$avg) // Si la puntuación del usuario es sustancialmente
inferior a la media (< 90%)
			$penalties[$user][]=$topic;
	}
	$report[$topic]["Average"]=$avg;
}

print_r($report);

echo "Avg. similarity: ".(array_sum($allSimilarities)/sizeof($allSimilarities))."\n";

// print_r($penalties); // Para ver en qué topics un usuario tiene puntuaciones
inferiores a la media

return;

function compareEvaluatorVSothers($topic,$evaluator) {
	global $summaries;

	$localSummaries=$summaries[$topic];

	foreach ($localSummaries as $e=>$array) {
		foreach ($array as $summary) {
```

```
                        if ($e==$evaluator)
                                $evaluatorSummaries[]=$summary;
                        else
                                $restSummaries[]=$summary;
                }
        }

        $evaluatorSummaries=implode(" ",$evaluatorSummaries);
        $restSummaries=implode(" ",$restSummaries);

        $evaluatorSummaries=vectorize($evaluatorSummaries);
        $restSummaries=vectorize($restSummaries);

        return cosine($evaluatorSummaries,$restSummaries);
}

function cosine($v,$w) {
        foreach ($v as $key=>$value) {
                if (isset($w[$key]))
                        $intersect[$key]=min($value,$w[$key]);
        }

        return array_sum($intersect)/sqrt(array_sum($v)*array_sum($w));
}

function vectorize($string) {
        $ngrams=extract3grams($string);
        foreach ($ngrams as $ngram) {
                if (!isset($vector[$ngram]))
                        $vector[$ngram]=0;
                $vector[$ngram]++;
        }

        arsort($vector);

        return $vector;
}

function extract3grams ($string) {
        $string=mb_strtolower($string,"UTF-8");
        for ($i=0;$i<(mb_strlen($string,"UTF-8")-2);$i++) {
                $ngrams[]=mb_substr($string,$i,3,"UTF-8");
        }
        return $ngrams;
}

?>
```

Anexo: 2 Puesta en producción de los prototipos

2.1 Introducción

En este Anexo se tratarán de explicar las principales características de los pasos a seguir para la puesta en producción de los prototipos tratados en el proyecto, así como la inclusión de una aplicación web para la evaluación de los algoritmos por parte de usuarios.

2.2 Análisis de alto nivel

A continuación se expondrán los requisitos que debería cumplimentar el sistema.

2.2.1 Especificación de requisitos

2.2.1.1 Requisitos funcionales

2.2.1.1.1 Requisitos Funcionales para las aplicación de descarga de datos

Código	Nombre Requisito	Descripción del Requisito
R1.1	Descarga de datos de Twitter	Se implementará un sistema que descargue datos de Twitter
R1.2	Establecimiento periodos de descarga	Posibilidad de establecer duración de los periodos de descarga. Por ejemplo poner fase de 15 minutos
R1.3	Almacenamiento de datos	Los información descargada deben almacenarse en una Base de datos

2.2.1.1.2 Requisitos Funcionales para las aplicación de generación de resúmenes

Código	Nombre Requisito	Descripción del Requisito
R2.1	Algoritmos de resumen	Se implementará los algoritmos que permitan generar resúmenes a partir de los datos descargados
R2.2	Filtro de SPAM	Se incluirá un filtro de SPAM Bayesiano, que previamente a su inclusión será entrenado
R2.3	Filtro de palabras vacías	Se incluirá un filtro que detecte las palabras vacías en los procesos de puntuación
R2.4.	Almacenamiento de resúmenes generados	Se podrán almacenar los resúmenes generados en una BBDD para su posterior utilización

R2.5	Interfaz para selección de ficheros a filtrar	Se hará una pequeña interfaz donde se podrá seleccionar los Trending topic a resumir

8.1.2.1.1 Requisitos Funcionales para la aplicación de Evaluación

8.1.2.1.1 Cliente Anónimo

Código	Nombre Requisito	Descripción del Requisito
R3.1	Ver página de Bienvenida	Podrá ver la página de Bienvenida de la aplicación de evaluación
R3.2	Registro de Usuarios	Dispondrá de una opción de registro, donde a través de rellenar un formulario con sus datos personales, se confirme como usuario registrado de la aplicación y así poder disponer de las funciones de un usuario registrado.

8.1.2.1.1 Cliente Registrado

Código	Nombre Requisito	Descripción del Requisito
R4.1	Funcionalidad cliente anónimo	Dispondrá de las funciones que tiene el usuario anónimo.
R4.2	Acceder a su perfil	Podrá acceder a su perfil previo acceso a través de su Nick y contraseña.
R4.3	Modificar datos personales	Podrá modificar sus datos personales.
R4.4	Cambiar contraseña	Podrá cambiar su contraseña.
R4.5	Evaluar resúmenes automáticos	Dispondrá de una opción para realizar las evaluaciones de resúmenes automáticos.
R4.6	Evaluar resúmenes de otros evaluadores	Dispondrá de una opción para realizar las evaluaciones de resúmenes de otros evaluadores.
R4.7	Historial de evaluaciones	Podrá comprobar las evaluaciones realizadas .a través de una página habilitada dentro del propio perfil del usuario
R4.4	Ver Instrucciones	Podrá consultar las instrucciones para realizar una evaluación a través de una página para dicho fin

8.1.2.1.1 Administrador

Código	Nombre Requisito	Descripción del Requisito
R5.1	Gestión contenido.	Deberá poder gestionar los contenidos de la aplicación
R5.2	Gestión de resúmenes	Podrá Insertar/Borrar/Actualizar los resúmenes para evaluar
R5.3	Gestión usuarios	Podrá Insertar/Borrar/Actualizar usuarios en la web

8.1.2.2 Requisitos no funcionales

8.1.2.2.1 Tecnológicos

Código	Nombre Requisito	Descripción del Requisito
R6.1	Web dinámica.	Deberá ser una web dinámica que combine las tecnologías JSP, xHTML, CSS, Javascript y el framework Struts2.
R6.2	Web 2.0	Deberá ser una web 2.0
R6.3	Estándar	Deberá cumplir los estándares del W3C: -xHtml 1.1 -CSS -AAA

8.1.2.2.2 Hardware

Código	Nombre Requisito	Descripción del Requisito
R7.1	Servidor	La aplicación deberá ser desplegada en ordenador que posea un servidor Tomcat superior a la versión 5, dicho ordenador debe poseer los requerimientos mínimos del servidor tomcat.

2.2.1.1.3 Usabilidad

Código	Nombre Requisito	Descripción del Requisito
R8.1	Usable	Deberá ser una web que aporte una interfaz amigable y fácil de usar para el usuario.
R8.2	Accesible	Sus contenidos deben ser accesibles por tanto deberá seguir el estándar AAA

2.2.1.1.4 Seguridad

Código	Nombre Requisito	Descripción del Requisito
R9.1	Cifrado contraseña	La contraseña deberá ser almacenada de forma encriptada en la base de datos.
R9.2	Validación de formularios	Se deberán validar los datos introducidos en los formularios.
R9.3	Distintos perfiles en la BBDD	Se deberán crear distintos perfiles de acceso a datos dentro de la BBDD.
R9.4	Cumplimiento de la LOPD	Debido a que la aplicación almacenará datos personales de usuarios, deberá cumplir los

		requerimientos de la ley orgánica de protección de datos (LOPD); como esta aplicación es utilizada para el comercio deberá cumplir también la ley de servicios de la sociedad de la información y del comercio electrónico (LSSI).

2.2.1.1.5 Adaptabilidad

Código	Nombre Requisito	Descripción del Requisito
R10.1	Adaptable	Debido a la continua evolución de la web y a los continuos cambios que se hacen dentro de las empresas, deberá ser una web cuyas modificaciones sean fácilmente adaptables.

2.2.1.1.6 Portabilidad

Código	Nombre Requisito	Descripción del Requisito
R11.1	Portabilidad	Debe ser susceptible a cambios de plataforma

2.2.1.1.7 Tiempo de respuesta

Código	Nombre Requisito	Descripción del Requisito
R12.1	Tiempo Razonable	La aplicación debe responder a las consultas en un tiempo razonable.

2.3 Esquema de BBDD

En la siguiente imagen se puede observar un diagrama de alto nivel para la BBDD de la aplicación de Evaluación:

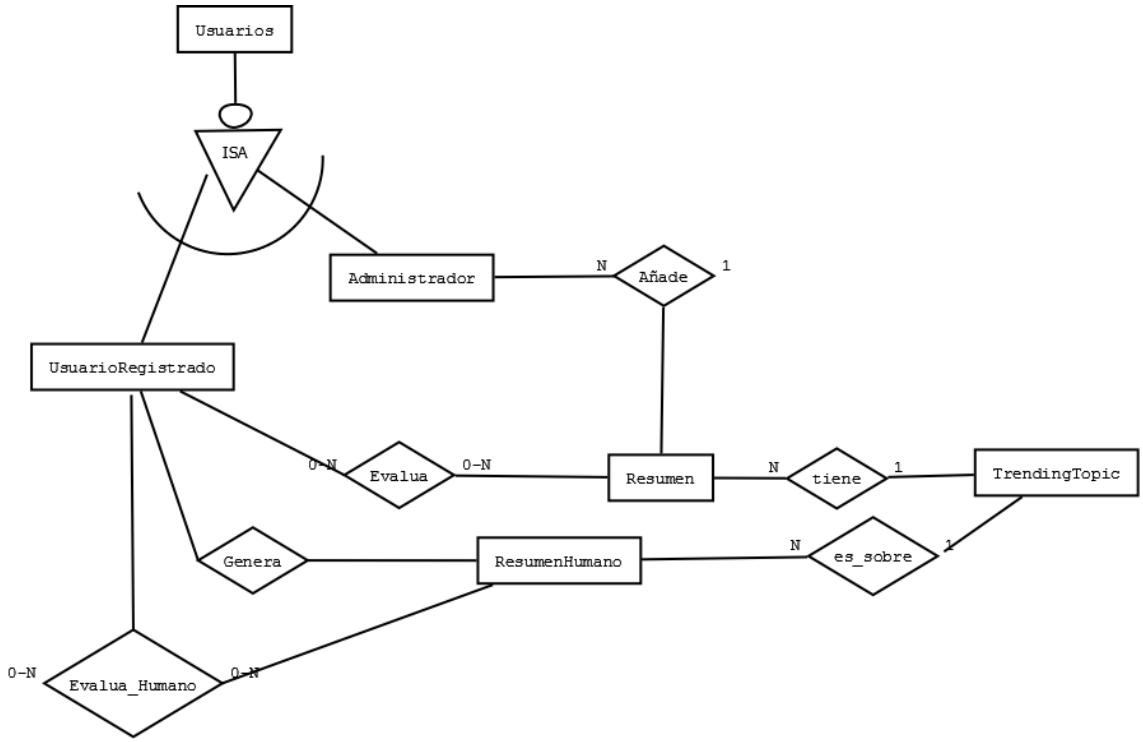


Ilustración 1 Diagrama de la BBDD para la aplicación de evaluación

2.4 Escenarios

2.4.1 Identificación de actores del sistema

La aplicación tendría 3 tipos de usuarios que representarían roles diferentes:

- Por una parte está el **usuario anónimo**: que sólo podrá acceder a la página principal y registrarse.
- Por otra parte está el **usuario registrado** que dispondrá por una parte de la funcionalidad que tenga el usuario anónimo y por otra podrá realizar las siguientes funciones:
    - Evaluación de resúmenes generados automáticamente por la aplicación de resúmenes y de resúmenes generados por humanos.
    - Consulta de su perfil para realizar diversas acciones como ver su historial de resúmenes, cambiar su contraseña…
- Por último se encuentra el **administrador**, que tendrá el control de todos los contenidos de la web como habilitar y deshabilitar trending topics para su evaluación y administrar los usuarios.

2.4.2 Especificación y diagramas de casos de uso

A continuación se puede observar el diagrama de casos de uso para todos los actores del sistema:

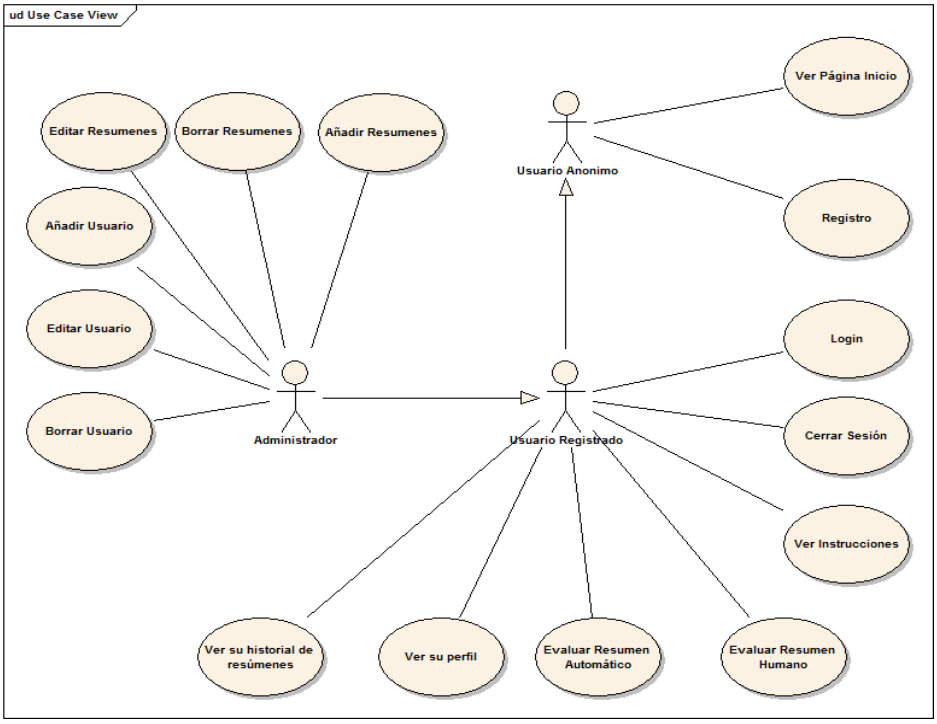


Ilustración 2 Casos de uso generales para la aplicación de evaluación

2.4.2.1 Caso de uso 1: Registro en el sistema

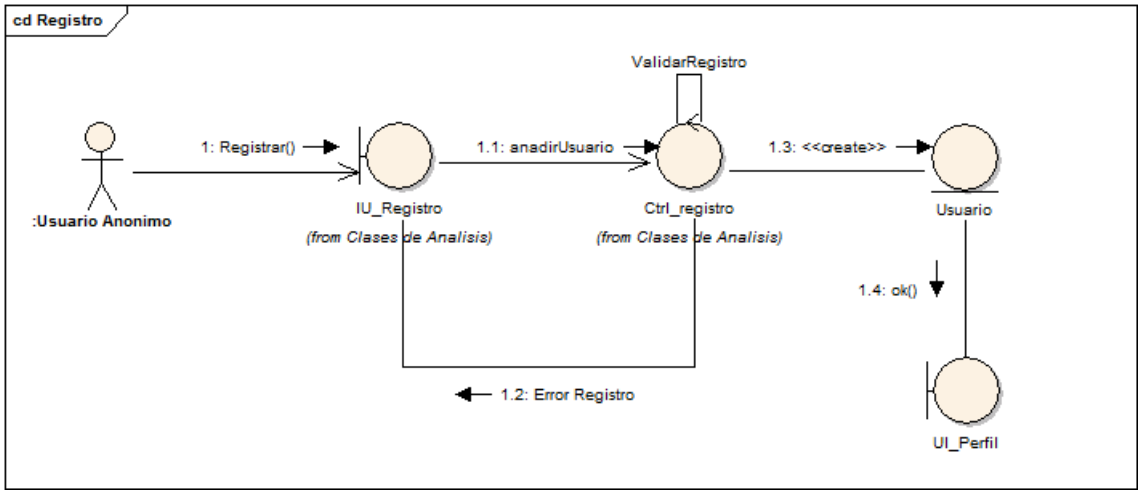


Ilustración 3 Caso de uso Registro en el sistema

Registro en el Sistema	
Precondiciones	No
Postcondiciones	El usuario debe estar registrado y con un rol asignado
Actores	Iniciado por un usuario anónimo finalizado por un usuario registrado.
Descripción	El usuario anónimo: ○ Accede a la página de registro ○ Rellena el formulario con sus datos ○ Guarda sus datos en el sistema ○ Es redirigido a su espacio personal donde tendrá su perfil como usuario registrado.
Variaciones (escenarios secundarios)	**Escenario Alternativo 2.1:** El usuario Anónimo: ○ Accede a la página de registro ○ Rellena el formulario con sus datos personales ○ Falla la validación de sus datos ○ Es redirigido otra vez hacia el formulario de registro
Excepciones	1. **La base de datos no está disponible:** ○ Notificar un error informando sobre el problema en cuestión
Notas	-

2.4.2.2 Caso de uso 2: Login

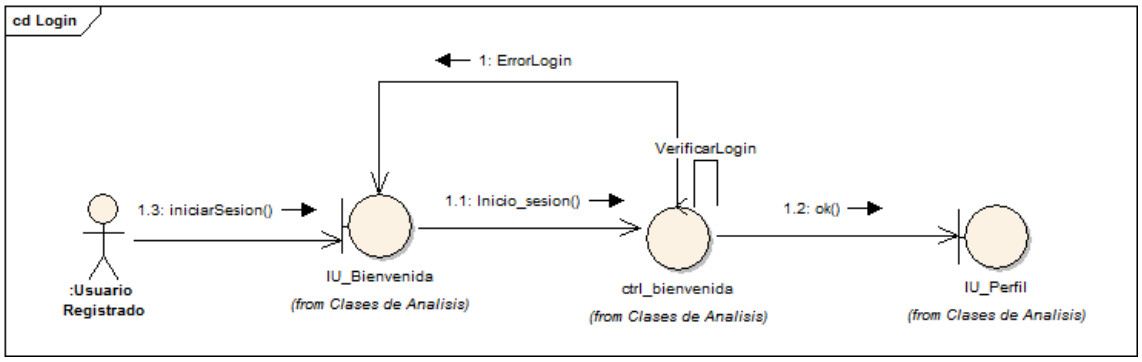


Ilustración 4 Caso de uso Login

Login	
Precondiciones	Estar registrado en el sistema
Post-condiciones	El usuario debe poder iniciar sesión
Actores	Usuario registrado
Descripción	El usuario registrado: ○ Accede a la pantalla de login ○ Introduce su usuario y contraseña ○ Accede a su espacio personal
Variaciones (escenarios secundarios)	**Escenario Alternativo 3.1:** ○ **El usuario registrado:** ○ Accede a la pantalla de login ○ Introduce su usuario y contraseña ○ No valida sus datos de usuario ○ Sigue en la página de login recibiendo un mensaje de que los datos son incorrectos
Excepciones	**1. La base de datos no está disponible:** Notificar un error informando sobre el problema en cuestión
Notas	-

2.4.2.3 Caso de uso 3: Cerrar Sesión

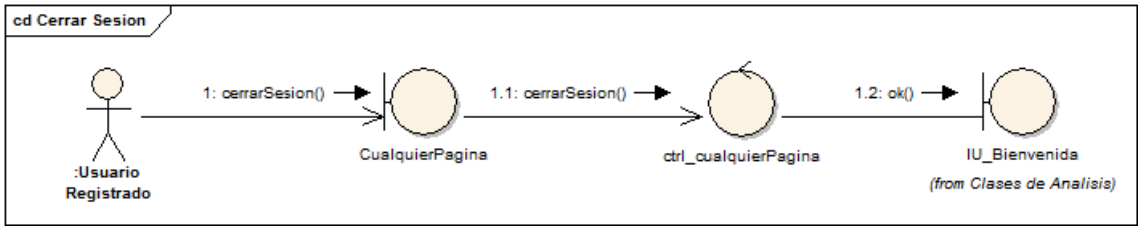


Ilustración 5 Caso de uso Cerrar Sesión

Cerrar Sesión	
Precondiciones	El usuario registrado debe estar en sesión
Post-condiciones	El usuario registrado estará desconectado en su sesión
Actores	Usuario registrado
Descripción	El Usuario registado: ○ Desde cualquier página de la interfaz principal ○ Pulsará el link de cerrar sesión ○ Será redirigido a la pantalla de inicio de sesión previo cierre de su sesión
Variaciones (escenarios secundarios)	-
Excepciones	**-**
Notas	-

2.4.2.4 Caso de uso 4: Ver Perfil

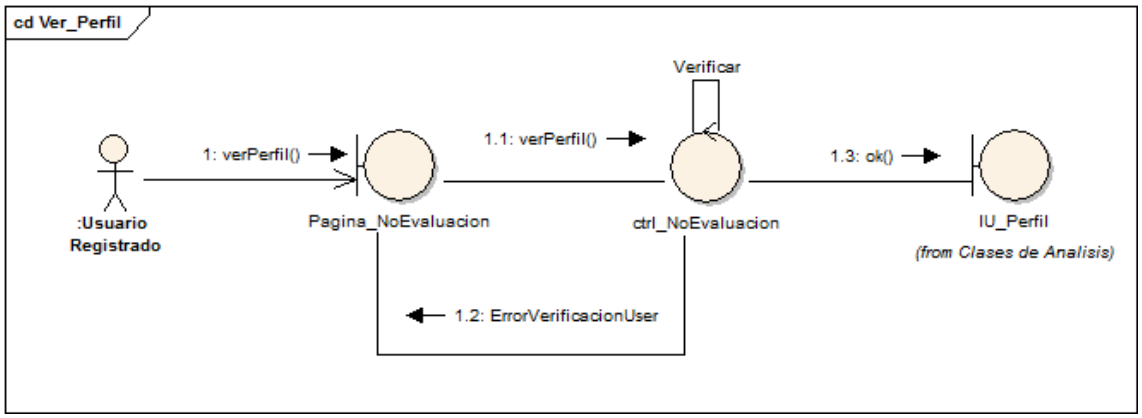


Ilustración 6 Caso de uso Ver Perfil

Ver Perfil	
Precondiciones	El usuario registrado debe estar en sesión
Post-condiciones	El usuario registrado debe poder haber visto su perfil
Actores	Usuario registrado
Descripción	El Usuario registrado: ○ Podrá ver su perfil desde cualquier Pantalla de la aplicación que no sea de evaluación pulsando sobre un enlace de ver perfil
Variaciones (escenarios secundarios)	-
Excepciones	**1. - La base de datos no está disponible:** Notificar un error informando sobre el problema en cuestión
Notas	-

2.4.2.5 Caso de uso 5: Evaluar Resumen Humano

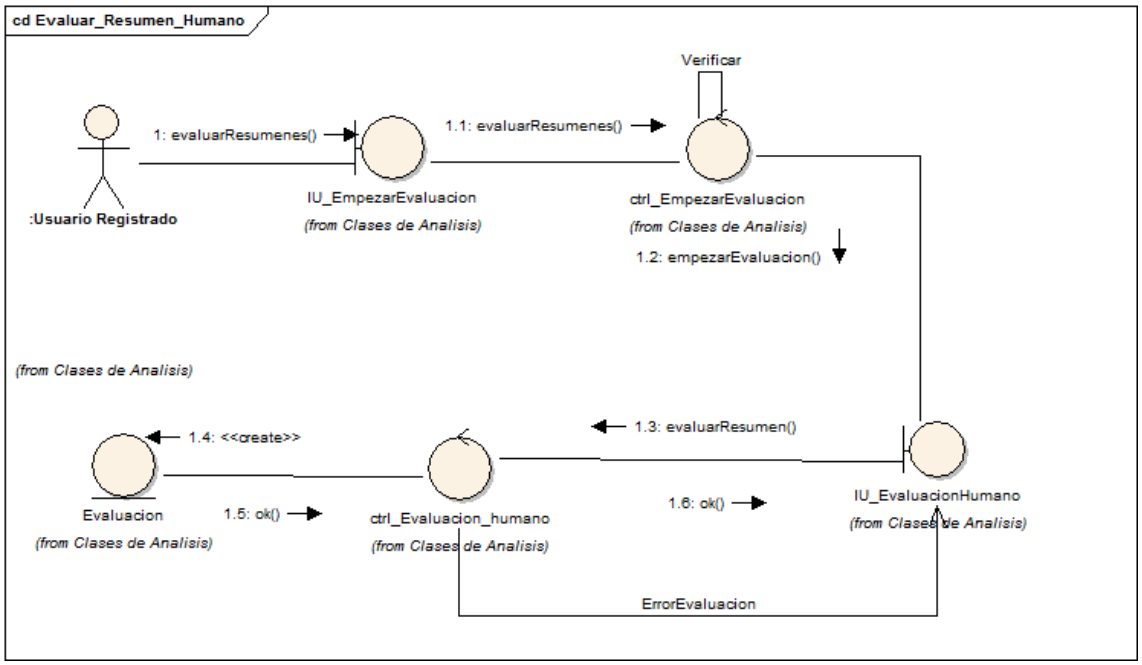


Ilustración 7 Caso de uso Evaluar Resumen Humano

Evaluar Resumen Humano	
Precondiciones	El usuario registrado debe estar en sesión
Post-condiciones	El usuario debe haber podido realizar la evaluación del resumen humano
Actores	Usuario registrado
Descripción	El usuario registrado debe situarse primero en la página de evaluación humana: ○ Accediendo previamente a su perfil dispondrá de un enlace para Realizar evaluaciones ○ Pulsará sobre ese enlace y le redirigirá a una ventana donde deberá elegir el tipo de evaluación desea efectuar ○ Seleccionará la opción de evaluación de resúmenes humanos ○ Le aparecerá un formulario que deberá rellenar con respecto a los tuits que tenga que evaluar. ○ Una vez rellenado confirmará la evaluación ○ El sistema le indicará que la evaluación ha sido efectuado con éxito
Variaciones (escenarios secundarios)	-
Excepciones	**1. - La base de datos no está disponible:** Notificar un error informando sobre el problema en cuestión
Notas	-

2.4.2.6 Caso de uso 6 Evaluar Resumen Automático

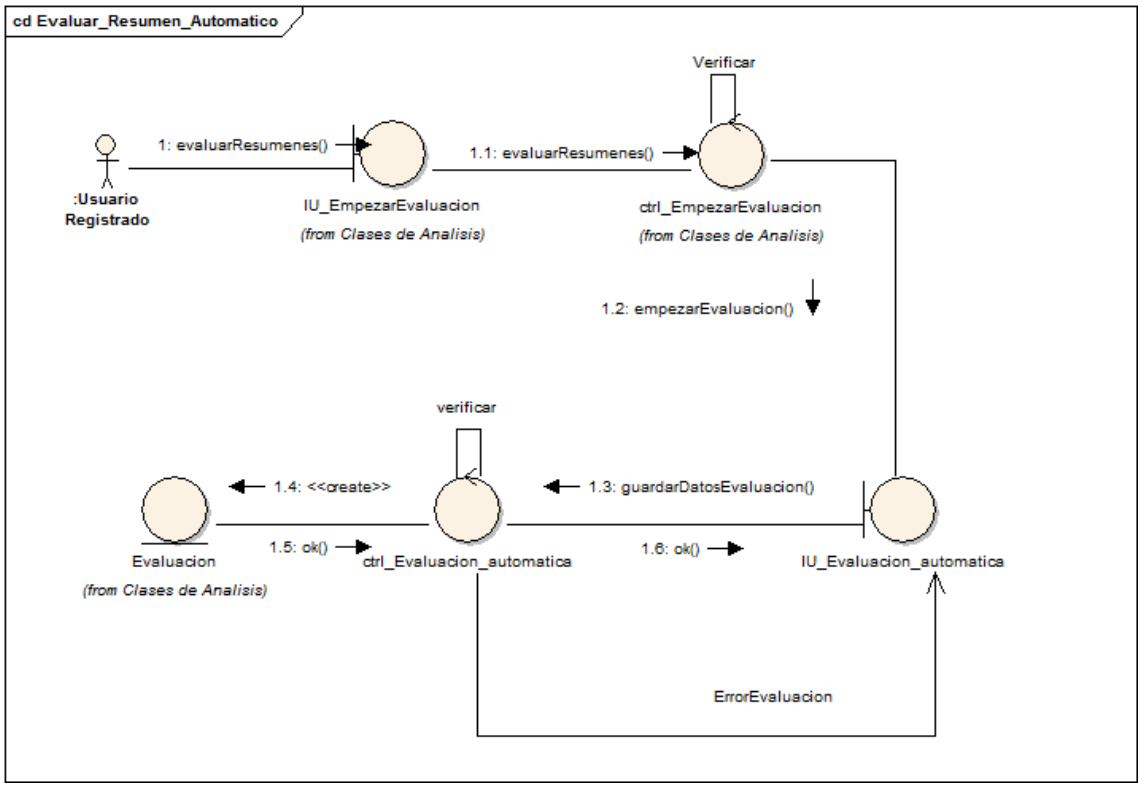


Ilustración 8 Caso de uso Evaluar Resumen Automático

Evaluar Resumen Automático	
Precondiciones	El usuario registrado debe estar en sesión
Post-condiciones	El usuario debe haber realizado una evaluación de resúmenes automáticos
Actores	Usuario registrado
Descripción	El Usuario registrado debe situarse primero en la página de evaluación de resumen automático: ○ Accediendo previamente a su perfil dispondrá de un enlace para Realizar evaluaciones ○ Pulsará sobre ese enlace y le redirigirá a una ventana donde deberá elegir el tipo de evaluación desea efectuar ○ Seleccionará la opción de evaluación de resúmenes automáticos ○ Le aparecerá un formulario que deberá rellenar con respecto a los tuits que tenga que evaluar. ○ Una vez rellenado confirmará la evaluación ○ El sistema le indicará que la evaluación ha sido efectuado con éxito
Variaciones (escenarios secundarios)	-
Excepciones	1. **-- La base de datos no está disponible:** Notificar un error informando sobre el problema en cuestión
Notas	-

2.4.2.7 Caso de uso 7: Cambiar Contraseña

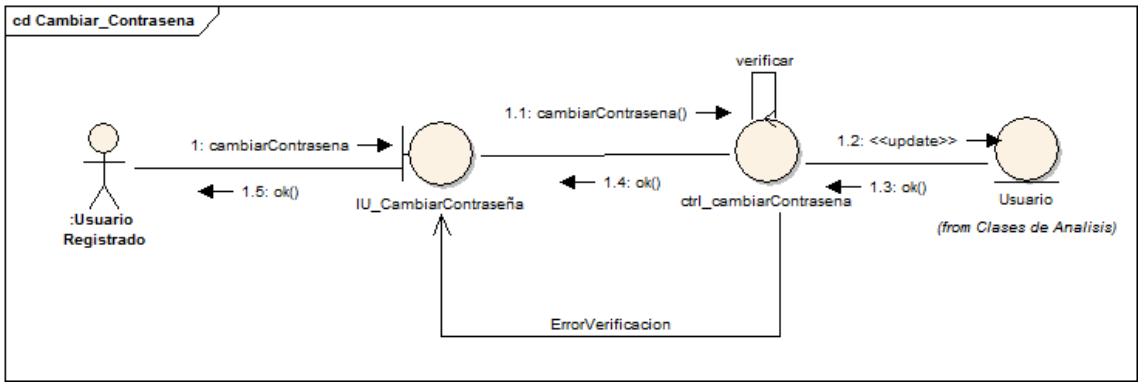


Ilustración 9 Caso de uso Cambiar Contraseña

Cambiar Contraseña	
Precondiciones	El usuario registrado debe estar en sesión
Post-condiciones	El usuario registrado habrá podido cambiar su contraseña
Actores	Usuario registrado
Descripción	El Usuario registrado: ○ Accediendo previamente a su página de perfil ○ Pulsará el link de cambiar contraseña ○ Será redirigido a la pantalla de cambio de contraseña donde deberá especificar la antigua y la nueva contraseña ○ Confirmará la solicitud ○ Le redirigirá a su perfil mostrándole previamente un mensaje que la contraseña ha sido cambiada con éxito
Variaciones (escenarios secundarios)	-
Excepciones	**1. - La base de datos no está disponible:** Notificar un error informando sobre el problema en cuestión
Notas	-

2.4.2.8 Caso de uso 8: Ver Instrucciones

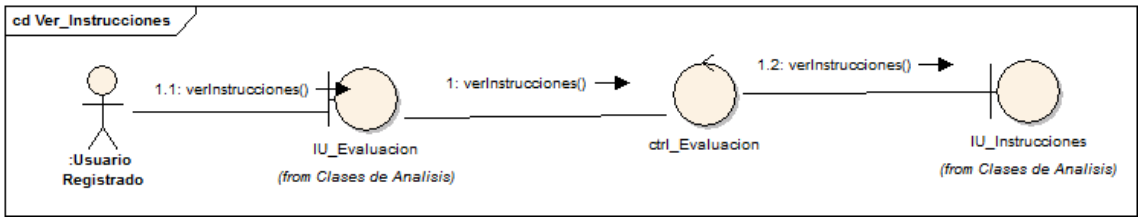


Ilustración 10 Caso de uso Ver Instrucciones

Ver Instrucciones	
Precondiciones	El usuario registrado debe estar en sesión
Post-condiciones	El usuario registrado deberá poder haber visto la página de instrucciones para la evaluación
Actores	Usuario registrado
Descripción	El Usuario registado: ○ Desde cualquier página de evaluación ya sea de resúmenes automáticos como de humanos dispondrán de un enlace para ver las instrucciones de evaluación. ○ Pulsará el link de ver instrucciones ○ Será redirigido a la pantalla en que se le muestran las instrucciones
Variaciones (escenarios secundarios)	-
Excepciones	-
Notas	-

8.1.2.3

2.4.2.9 Caso de uso 9: Ver historial de resúmenes

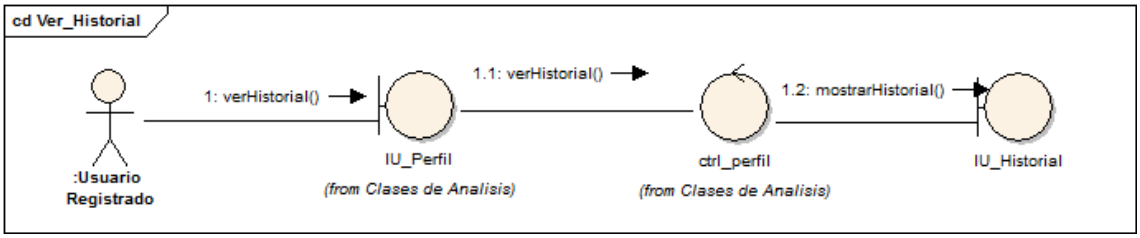


Ilustración 11 Caso de uso Ver historial de resúmenes

Ver historial de resúmenes	
Precondiciones	El usuario registrado debe estar en sesión
Post-condiciones	El usuario habrá visto la página en la que se muestran todas las evaluaciones que ha realizado
Actores	Usuario registrado
Descripción	El Usuario Registrado ○ Partiendo desde su página de perfil dispondrá de un enlace para acceder a su historial de resúmenes ○ Pulsará el link para a ver su historial ○ Será redirigido a la pantalla en que se muestran las instrucciones
Variaciones (escenarios secundarios)	-
Excepciones	**1. -- La base de datos no está disponible:** Notificar un error informando sobre el problema en cuestión
Notas	-

2.4.2.10 Caso de uso 10: Añadir Resúmenes

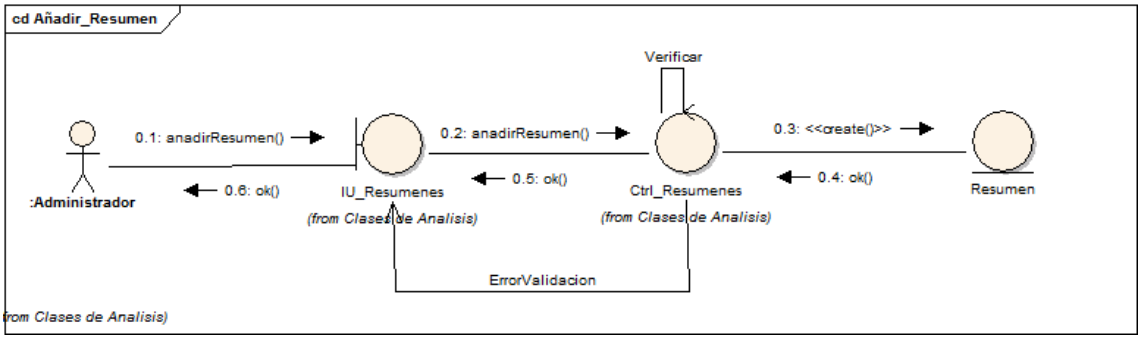


Ilustración 12 Caso de uso Añadir Resúmenes

Añadir resúmenes	
Precondiciones	El administrador debe estar en sesión
Post-condiciones	El administrador podrá haber añadido resúmenes a evaluar a la aplicación
Actores	Usuario administrador
Descripción	El Administrador: ○ Partiendo desde su página de inicio de la web de administración ○ Dispondrá de una opción para añadir resúmenes ○ Al elegirla le redirigirá a una pantalla para poder subir resúmenes ○ Desde ahí el usuario podrá subir un fichero con los resúmenes a añadir y confirmando la operación pulsando en un botón Añadir Resúmenes ○ Será redirigido a la pantalla principal de la web de administración mostrándose en su historial que ha añadido resúmenes a una hora concreta
Variaciones (escenarios secundarios)	-
Excepciones	**2. -- La base de datos no está disponible:** Notificar un error informando sobre el problema en cuestión
Notas	-

2.4.2.11 Caso de uso 11: Borrar resúmenes

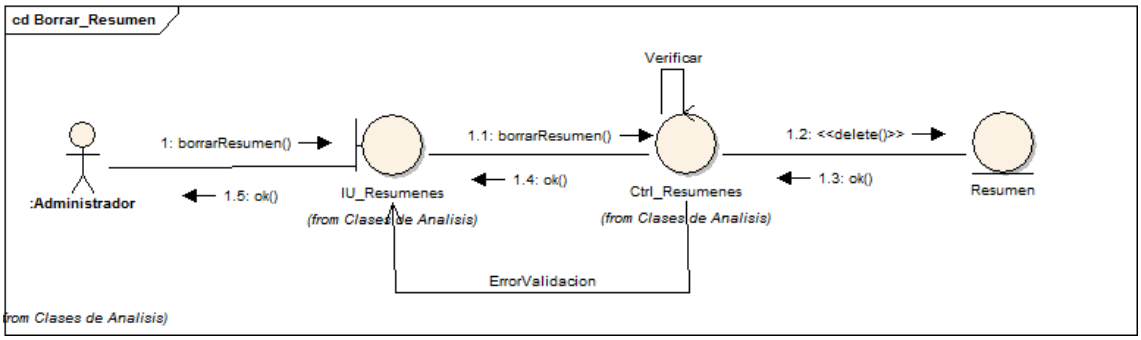


Ilustración 13 Caso de uso Borrar Resúmenes

Borrar resúmenes	
Precondiciones	El administrador debe estar en sesión
Post-condiciones	El usuario podido borrar un/unos resumen/es en particular
Actores	Usuario administrador
Descripción	El Administrador: ○ Partiendo desde su página de inicio de la web de administración ○ Dispondrá de una opción para borrar resúmenes ○ Al elegirla le redirigirá a una pantalla para poder borrar resúmenes ○ Desde ahí el usuario podrá buscar los resúmenes de un tema para borrar ○ Una vez encontrado el tema a borrar el administrador podrá eliminarlo seleccionándolo en una opción adyacente al resumen ○ Una vez haya seleccionado los resúmenes a borrar, confirmará la operación pulsando en un botón que confirme el borrado ○ Una vez confirmado será redirigido a la pantalla principal de la web de administración mostrándose en su historial que ha borrado resúmenes a una hora concreta
Variaciones (escenarios secundarios)	-
Excepciones	**3. -- La base de datos no está disponible:** Notificar un error informando sobre el problema en cuestión
Notas	-

2.4.2.12 Caso de uso 12: Editar resúmenes

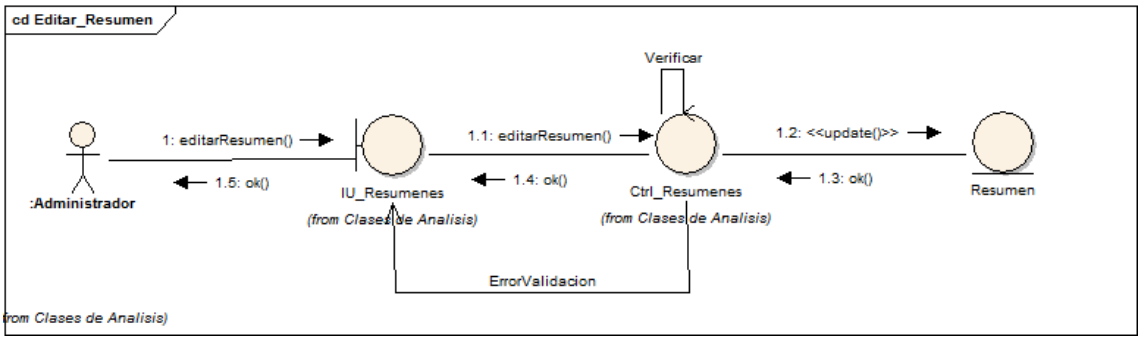


Ilustración 14 Caso de uso Editar resúmenes

Editar resúmenes	
Precondiciones	El administrador debe estar en sesión
Post-condiciones	El usuario podido editar un/unos resumen/es en particular
Actores	Usuario administrador
Descripción	El Administrador: ○ Partiendo desde su página de inicio de la web de administración ○ Dispondrá de una opción para editar resúmenes ○ Al elegirla le redirigirá a una pantalla para poder editar resúmenes ○ Desde ahí el usuario podrá buscar los resúmenes de un tema para editar ○ Una vez encontrado el tema a editar el administrador podrá modificarlo modificando un formulario adaptado al tipo de contenido que contenga cada resumen ○ Una vez haya editado los resúmenes en el formulario, confirmará la operación pulsando en un botón que confirme la edición ○ Una vez confirmado será redirigido a la pantalla principal de la web de administración mostrándose en su historial que ha editado resúmenes a una hora concreta
Variaciones (escenarios secundarios)	-
Excepciones	**4. -- La base de datos no está disponible:** Notificar un error informando sobre el problema en cuestión
Notas	-

2.4.2.13 Caso de uso 13: Añadir usuario

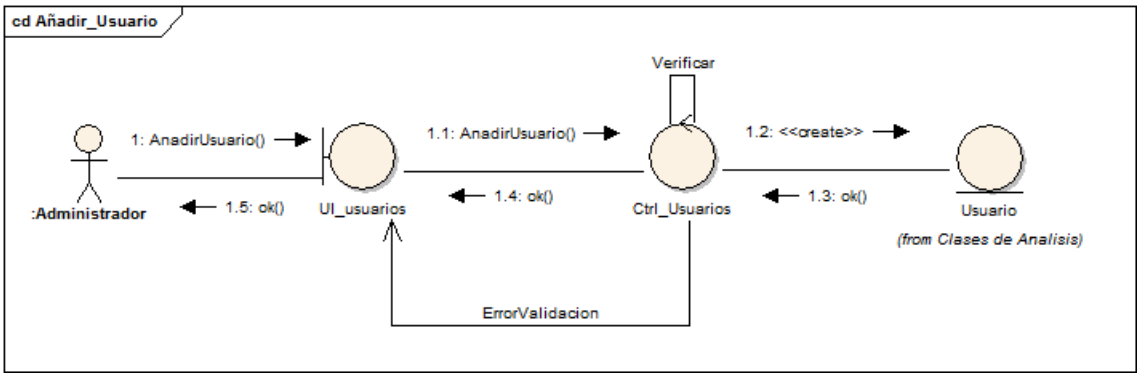


Ilustración 15 Caso de uso Añadir Usuario

Añadir usuario	
Precondiciones	El administrador debe estar en sesión
Post-condiciones	El usuario habrá visto la página en la que se muestran todas las evaluaciones que ha realizado
Actores	Usuario administrador
Descripción	El Administrador: ○ Partiendo desde su página de inicio de la web de administración ○ Dispondrá de una opción para añadir usuarios ○ Al elegirla le redirigirá a una pantalla para poder añadirlos rellenando un formulario ○ Tras rellenar el formulario confirmará el mismo pulsando un botón habilitado para ello ○ Una vez que se haya confirmado la operación será redirigido a la pantalla principal de la web de administración mostrándose en su historial que ha añadido un usuario a una hora concreta
Variaciones (escenarios secundarios)	-
Excepciones	**5. -- La base de datos no está disponible:** Notificar un error informando sobre el problema en cuestión
Notas	-

2.4.2.14 Caso de uso 14: Borrar usuario

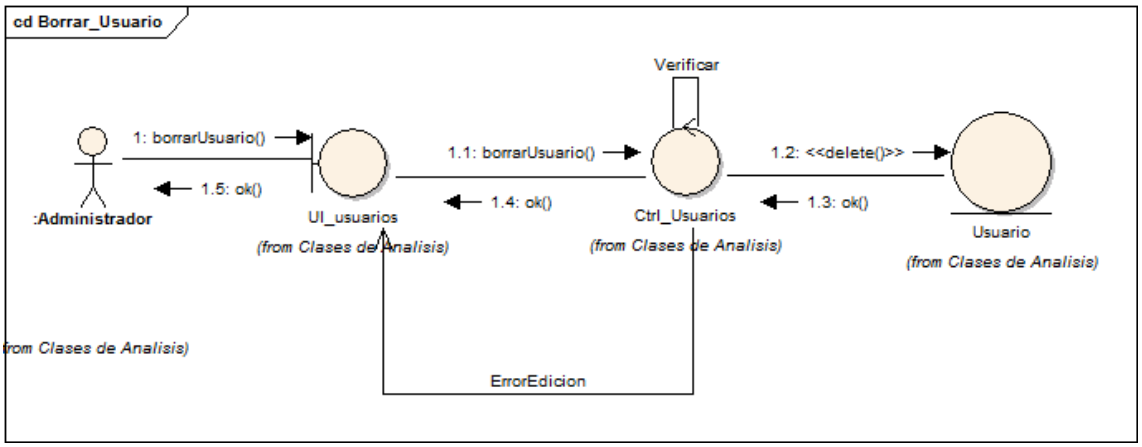


Ilustración 16 Caso de uso Borrar Usuario

Borrar usuario	
Precondiciones	El administrador debe estar en sesión
Post-condiciones	El usuario podrá haber borrado un/unos usuario/s en particular
Actores	Usuario administrador
Descripción	El Administrador: ○ Partiendo desde su página de inicio de la web de administración ○ Dispondrá de una opción para borrar usuarios ○ Al elegirla le redirigirá a una pantalla para poder borrar usuarios ○ Desde ahí el usuario podrá buscar los usuarios que desee borrar ○ Una vez encontrado el usuario a borrar el administrador podrá eliminarlo seleccionándolo en una opción adyacente al él ○ Una vez haya seleccionado el/los usuario/s a borrar, confirmará la operación pulsando en un botón que confirme el borrado ○ Una vez confirmado será redirigido a la pantalla principal de la web de administración mostrándose en su historial que ha borrado usuarios a una hora concreta
Variaciones (escenarios secundarios)	-
Excepciones	**6. -- La base de datos no está disponible:** Notificar un error informando sobre el problema en cuestión
Notas	-

2.4.2.15 Caso de uso 15: Editar Usuario

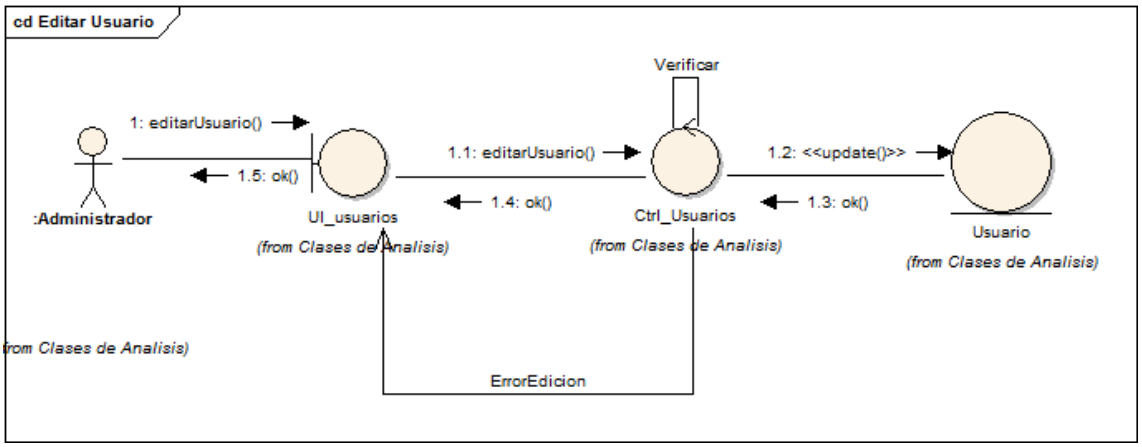


Ilustración 17 Caso de uso Editar Usuario

Editar usuario	
Precondiciones	El administrador debe estar en sesión
Post-condiciones	El usuario podido editar un/unos usuario/s en particular
Actores	Usuario administrador
Descripción	El Administrador: ○ Partiendo desde su página de inicio de la web de administración ○ Dispondrá de una opción para editar usuarios ○ Al elegirla le redirigirá a una pantalla para poder editarlos ○ Desde ahí el usuario podrá buscar los usuarios que desee editar ○ Una vez encontrado, el administrador podrá modificarlo modificando un formulario adaptado a la información que contenga el perfil de usuario ○ Tras haber editado al usuario en el formulario, confirmará la operación pulsando en un botón que confirme la edición ○ Al confirmar será redirigido a la pantalla principal de la web de administración mostrándose en su historial que ha editado resúmenes a una hora concreta
Variaciones (escenarios secundarios)	-
Excepciones	**7. -- La base de datos no está disponible:** Notificar un error informando sobre el problema en cuestión
Notas	-

-

2.5 Bocetos de Interfaz de Usuario

En la siguiente figura se puede observar el boceto de la página de bienvenida a la aplicación de evaluación donde los usuarios podrán iniciar sesión en la aplicación y de no estar registrados, podrán acceder a la pantalla de registro:

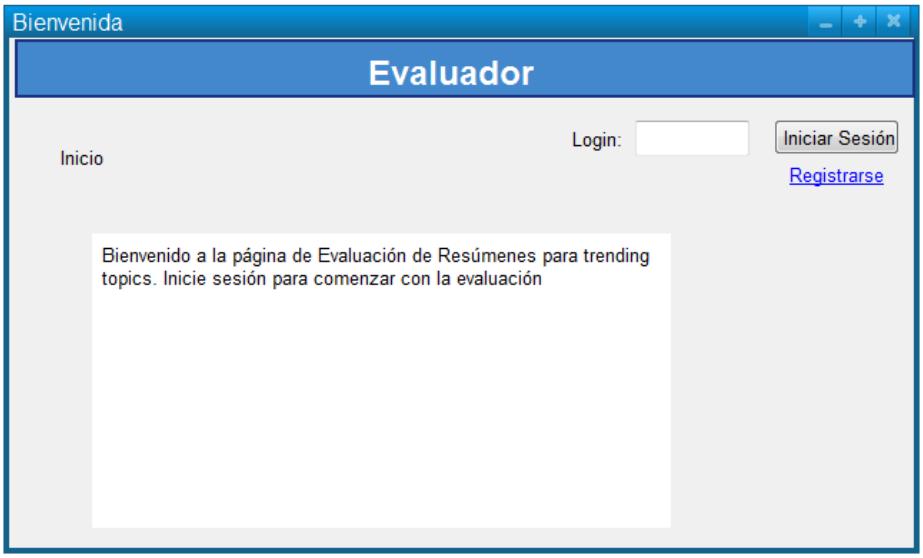


Ilustración 18 Pantalla de "Bienvenida" del Evaluado de resúmenes r

En la siguiente ilustración puede verse el borrador de ventana donde los usuarios podrán registrarse, dependiendo del estudio que desee hacer la empresa puede almacenar mas datos del evaluador.

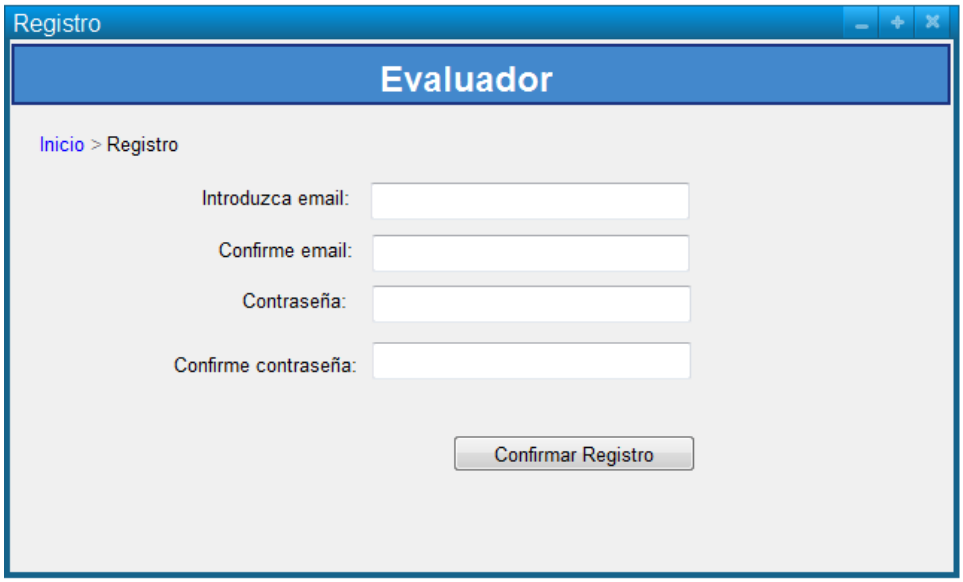


Ilustración 19 Pantalla de "Registro" del Evaluador de resúmenes

En la siguiente Ilustración puede verse el borrador de la ventana del perfil de los usuarios registrados que dispondrá de los accesos necesarios para el control y realización de las propias evaluaciones del usuario como de la gestión de sus datos personales, en este caso de ejemplo se pone el cambio de contraseña:

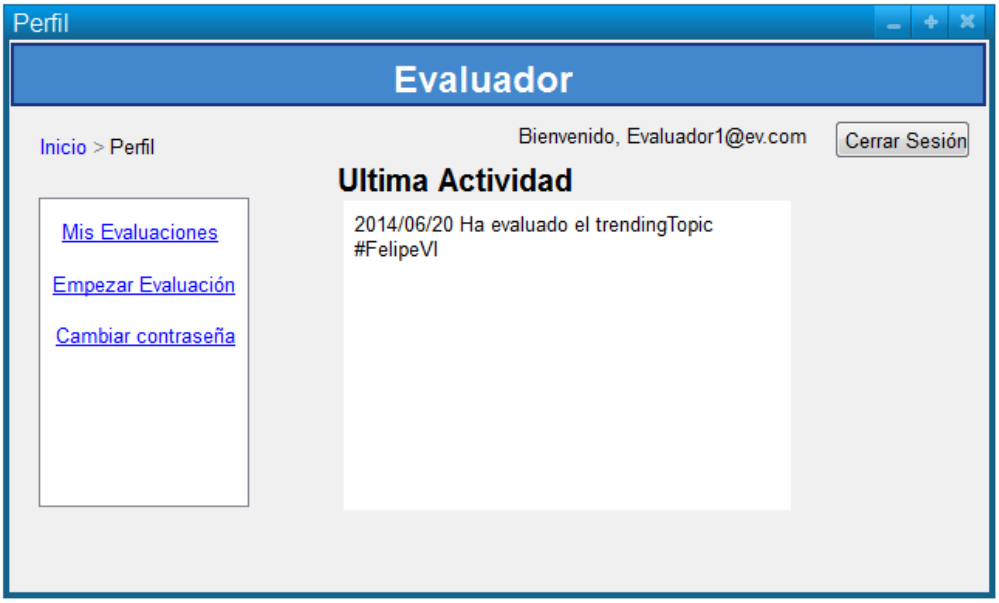


Ilustración 20 Pantalla del "Perfil" de Usuario del Evaluador de resúmenes

En la siguiente figura se muestra la ventana donde el usuario podría acceder a ver el historial de sus evaluaciones y consultar las mismas:

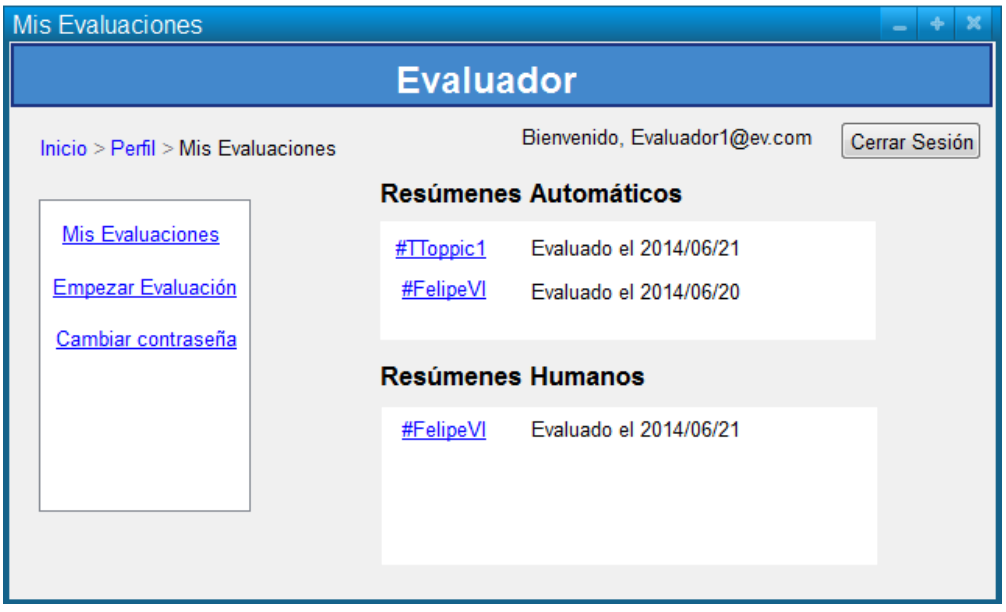


Ilustración 21 Pantalla "Mis Evaluaciones" del Evaluador de resúmenes

En la Ilustración 22 puede observarse el borrador de la pantalla que el usuario podrá elegir los resúmenes que desea Evaluar para efectuar dicho proceso.

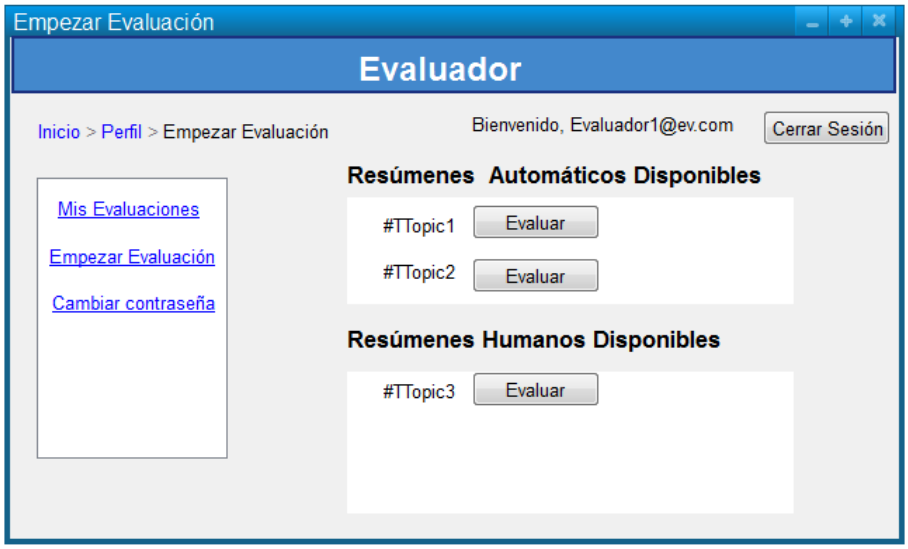


Ilustración 22 Pantalla "Empezar Evaluación" del Evaluador de resúmenes

En la Ilustración 23 se muestra el borrador de la Evaluación de resúmenes automáticos donde el usuario calificaría la relevancia de los Tuits así como produciría sus propios resúmenes acerca del tema en cuestión.

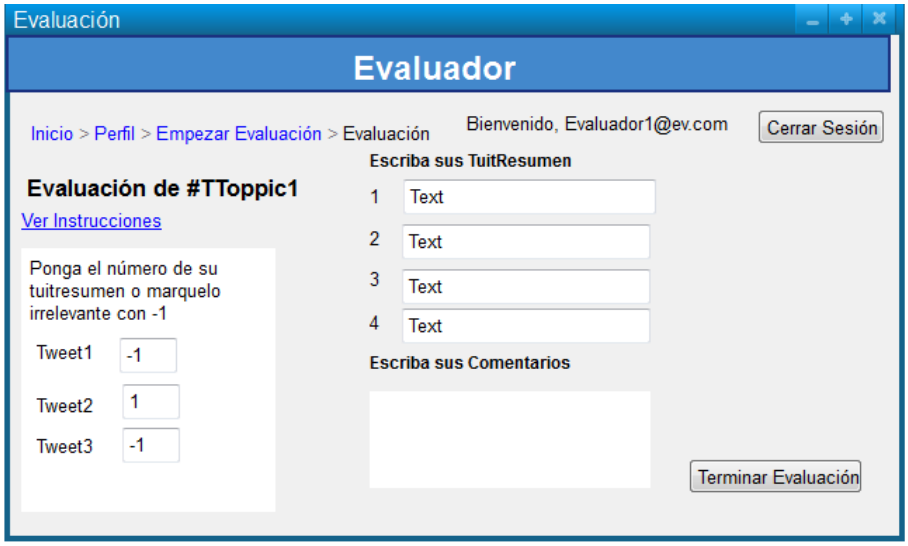


Ilustración 23 Pantalla "Evaluador" del Evaluador de resúmenes

En la Ilustración 24 se muestra el borrador para la ventana de evaluación de tuits resumen generados por los propios usuarios.

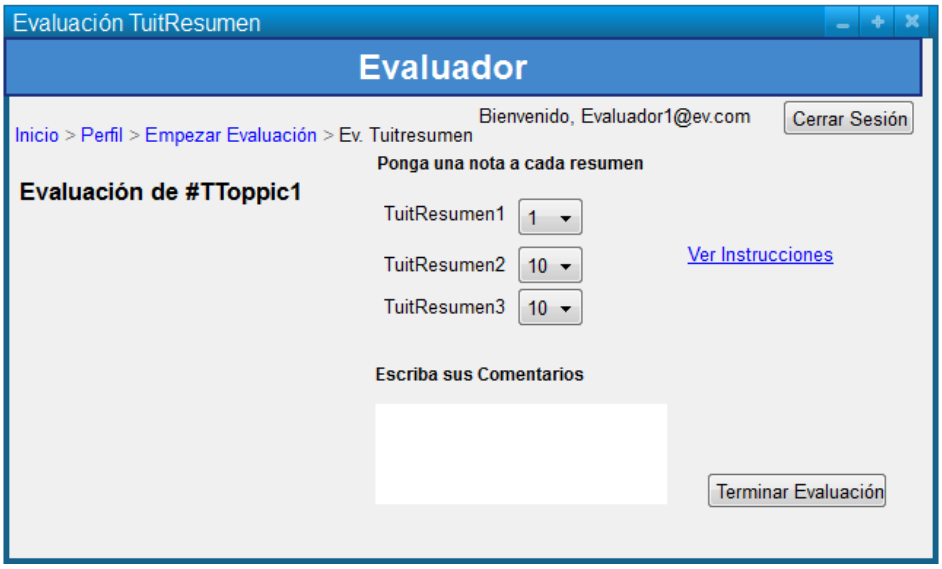


Ilustración 24 Pantalla "Evaluación TuitResumen" del Evaluador de resúmenes

En la Ilustración 25 se puede observar la ventana donde los usuarios podrán ver las instrucciones que deberán seguir para realizar la evaluación correctamente.

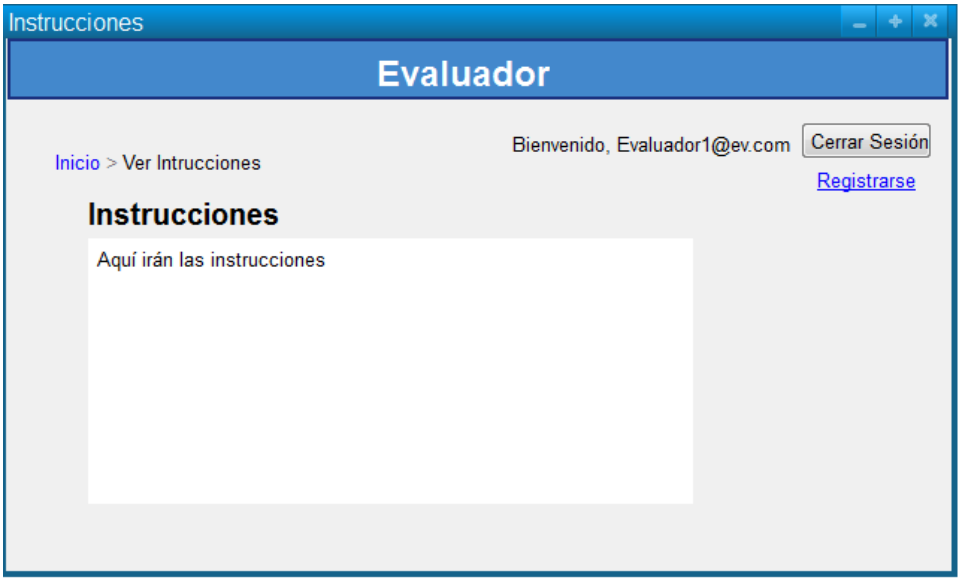


Ilustración 25 Pantalla "Instrucciones" del Evaluador de resúmenes

En la Ilustración 26 se observa el borrador para cambiar información del perfil de usuario, en este caso se ha basado en el cambio de contraseña, pero dependiendo de la información que se desee almacenar de los evaluadores, se tendrán que modificar más o menos la pantalla de edición de datos del perfil.

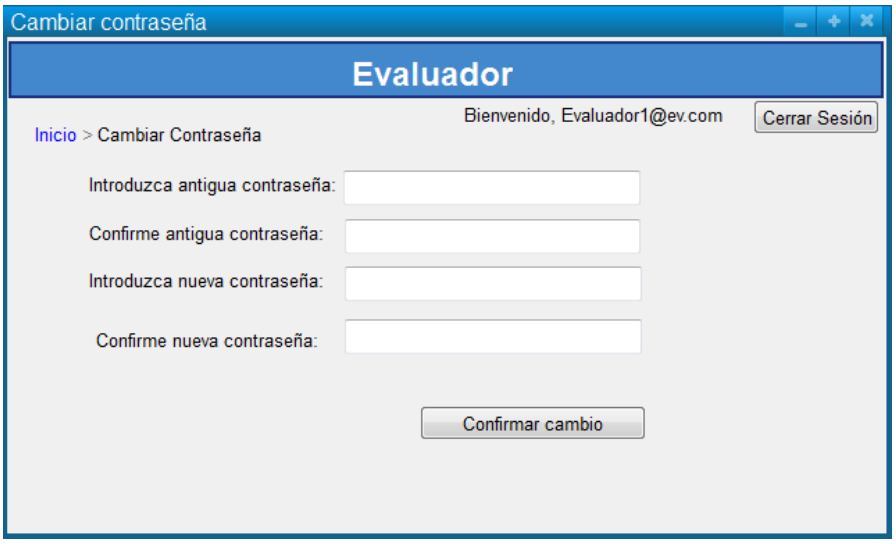


Ilustración 26 Pantalla "Cambiar Contraseña" del Evaluador de resúmenes

2.6 Planificación

A continuación se expondrán las tareas a realizar durante el proyecto y el Diagrama de Gantt

Nombre de tarea	Duración	Comienzo
1 OBTENCIÓN DE RESÚMENES AUTOMÁTICOS PARA TEMAS DE ACTU	**302 días?**	**vie 18/07/14**
1.1 Recopilación de datos (Fase I)	**206 días?**	**vie 18/07/14**
1.1.1 Reunirse con el director de proyecto (Fase I)	1 día	vie 18/07/14
1.1.2 Especificación de requisitos	3 días?	lun 21/07/14
1.1.3 Realizar la planificación	2 días	jue 24/07/14
1.1.4 Análisis de riesgos	35 días	lun 28/07/14
1.1.5 Estudio tecnologías a utilizar	10 días?	mié 17/09/14
1.1.6 Estudio de alternativas	7 días?	mié 01/10/14
1.1.7 Diseño BBDD (Modelo E-R)	5 días?	vie 10/10/14
1.1.8 Desarrollo BBDD	7 días?	vie 17/10/14
1.1.9 Probar Funcionamiento correcto BBDD	3 días?	mar 28/10/14
1.1.10 Desarrollar aplicación de descarga de información	**23 días**	**vie 31/10/14**
1.1.10.1 Diseño de aplicación de descarga de información	4 días	vie 31/10/14
1.1.10.2 Desarrollo de aplicación de descarga	7 días	jue 06/11/14
1.1.10.3 Pruebas para la aplicación de descarga	10 días	lun 17/11/14
1.1.10.4 Desplegar aplicación de descarga	2 días	lun 01/12/14
1.1.11 Documentación Primera parte Fase I	88 días	jue 24/07/14
1.1.12 Almacenar Información en BBDD y en ficheros con la aplicaci	114 días	mar 25/11/14
1.1.13 Realización de copia de seguridad de los datos descargados	114 días	mar 25/11/14
1.2 Análisis de datos (Fase II)	**70 días?**	**mié 03/12/14**
1.2.1 Reunirse con el director de proyecto (Fase II)	3 días?	mié 03/12/14
1.2.2 Elección de algoritmos	**67 días?**	**lun 08/12/14**
1.2.2.1 Estudio del arte previo en algoritmos de resumen	60 días	lun 08/12/14
1.2.2.2 Elección de algoritmos a implementar	7 días?	lun 02/03/15
1.2.3 Documentación Fase II	42 días?	lun 08/12/14
1.3 Desarrollar Aplicación Generador de resumenes (Fase III)	**53 días?**	**mié 11/03/15**
1.3.1 Reunirse con el director del proyecto (Fase III)	1 día	mié 11/03/15
1.3.2 Diseño de la aplicación	5 días?	jue 12/03/15
1.3.3 Desarrollo de la aplicación	40 días	jue 19/03/15
1.4.3 Realización de pruebas del prototipo	10 días	lun 03/08/15
1.5 Evaluación Previa (Fase V)	**18 días?**	**lun 17/08/15**
1.6 Documentación Fases III , IV y V	195 días	mié 03/12/14
1.7 Finalización de documentación y presentación del producto	**9 días?**	**mié 02/09/15**
1.7.1 Reunirse con el director de proyecto (Fase IV)	1 día	mié 02/09/15
1.7.2 Revisar documentación	2 días?	jue 03/09/15
1.7.3 Implantacion	4 días	lun 07/09/15
1.7.4 Presentación del producto al cliente	2 días	vie 11/09/15

Ilustración 27 Tareas para la puesta en producción del sistema de resumen

Id	Nombre de tarea	Duración
1	**1 OBTENCIÓN DE RESÚMENES AUTOMÁTICOS PARA TEMAS DE ACTUALII**	**302 días?**
2	**1.1 Recopilación de datos (Fase I)**	**206 días?**
3	1.1.1 Reunirse con el director de proyecto (Fase I)	1 día
4	1.1.2 Especificación de requisitos	3 días?
5	1.1.3 Realizar la planificación	2 días
6	1.1.4 Análisis de riesgos	35 días
7	1.1.5 Estudio tecnologías a utilizar	10 días?
8	1.1.6 Estudio de alternativas	7 días?
9	1.1.7 Diseño BBDD (Modelo E-R)	5 días?
10	1.1.8 Desarrollo BBDD	7 días?
11	1.1.9 Probar Funcionamiento correcto BBDD	3 días?
12	**1.1.10 Desarrollar aplicación de descarga de información**	**23 días**
13	1.1.10.1 Diseño de aplicación de descarga de información	4 días
14	1.1.10.2 Desarrollo de aplicación de descarga	7 días
15	1.1.10.3 Pruebas para la aplicación de descarga	10 días
16	1.1.10.4 Desplegar aplicación de descarga	2 días
17	1.1.11 Documentación Primera parte Fase I	88 días
18	1.1.12 Almacenar información en BBDD y en ficheros con la aplicación	114 días
19	1.1.13 Realización de copia de seguridad de los datos descargados	114 días
20	**1.2 Análisis de datos (Fase II)**	**70 días?**
21	1.2.1 Reunirse con el director de proyecto (Fase II)	3 días?
22	**1.2.2 Elección de algoritmos**	**67 días?**
23	1.2.2.1 Estudio del arte previo en algoritmos de resumen	60 días
24	1.2.2.2 Elección de algoritmos a implementar	7 días?
25	1.2.3 Documentación Fase II	42 días?
26	**1.3 Desarrollar Aplicación Generador de resumenes (Fase III)**	**53 días?**
27	1.3.1 Reunirse con el director del proyecto (Fase III)	1 día
28	1.3.2 Diseño de la aplicación	5 días?
29	1.3.3 Desarrollo de la aplicación	40 días
30	1.3.4 Realizar pruebas funcionamiento del generador de resumenes	7 días?
31	**1.4 Desarrollar prototipo Evaluación (Fase IV)**	**60 días?**
32	1.4.1 Diseño de la aplicación de evaluación	10 días?
33	1.4.2 Desarrollo de la aplicación de evaluación	40 días?
34	1.4.3 Realización de pruebas del prototipo	10 días
35	**1.5 Evaluación Previa (Fase V)**	**18 días?**
42	1.6 Documentación Fases III , IV y V	195 días
43	**1.7 Finalización de documentación y presentación del producto (Fase V)**	**9 días?**
44	1.7.1 Reunirse con el director de proyecto (Fase IV)	1 día
45	1.7.2 Revisar documentación	2 días?
46	1.7.3 Implantacion	4 días
47	1.7.4 Presentación del producto al cliente	2 días

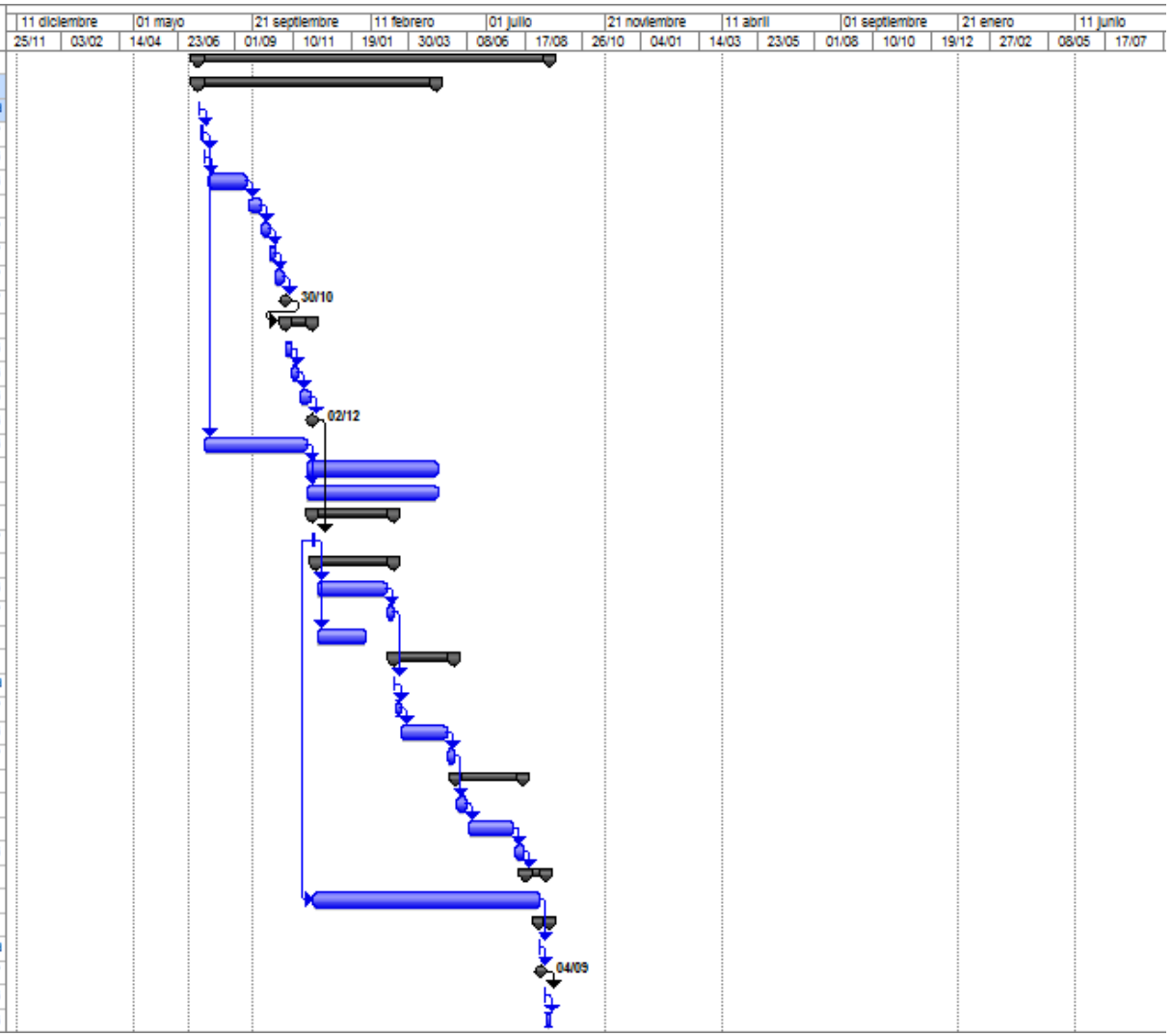


Ilustración 28 Diagrama de Gantt

2.7 Plan de Gestión de Riesgos

2.7.1 Metodología

La metodología que se va a utilizar para la gestión de riesgos será la metodología MAGERIT que se puede apreciar en la siguiente figura.

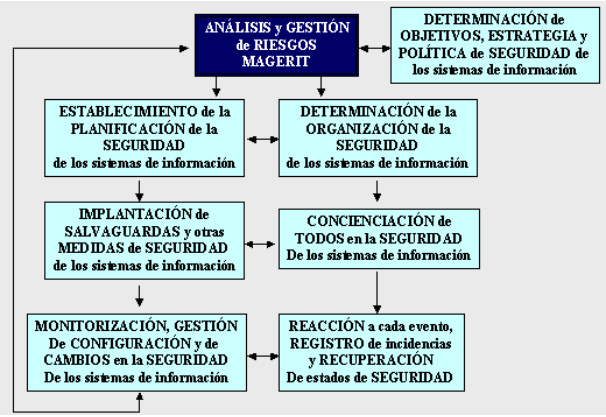


Ilustración 29

La fase de identificación comenzará con la determinación de los objetivos, estrategia y política de seguridad de la aplicación, para que posteriormente en la fase de análisis-planificación se establezca la planificación de seguridad del proyecto, una vez implantadas las medidas de seguridad y estar el alumno concienciado de los riesgos, en cada tarea se hará un registro de incidencias para monitorizar y gestionar los posibles riesgos, este será un proceso continuo.

2.7.2 Calendario

Para el análisis de riesgos se contempló la utilización de 35 días debido al alto grado de riesgo que contiene la elaboración de este proyecto.

Múltiples cambios en la tecnología o aplicaciones que quedaron en desuso y de las cuales se requería su utilización, han hecho que se haya asignado un tiempo razonable a analizar y monitorizar los riesgos pertinentes. La monitorización de dichos riesgos debería hacerse de forma diaria, ya que de no hacerlo se podría poner en peligro la finalización del proyecto.

2.7.3 Roles y Responsabilidades

Dependiendo de las personas que participasen en el proyecto debería de hacerse unos roles y responsabilidades diferentes, en este aspecto se muestra un ejemplo para este trabajo fin de máster:

Rol	Alumno
Descripción	Será la persona que identifique, analice, evite o solucione los riesgos
Responsabilidades	• Identificará el riesgo y hará una breve descripción del mismo, mostrándoselo al director de proyecto. • Hará un estudio de todos los riesgos analizándolos detalladamente verificando su probabilidad e impacto y determinando los métodos de evitación del riesgo, monitorización y gestión. • Solucionará o evitará riesgos en función de las decisiones del director de proyecto
Cualificación	Para cualquier proyecto real lo ideal sería que fuesen las personas que tuviesen los conocimientos adecuados para llevar a cabo sus funciones, pero en este caso será preciso que el alumno haga un esfuerzo estudiando las tecnologías que va a utilizar.

Rol	Director
Descripción	Será el director del proyecto que gestionará los riesgos del proyecto
Responsabilidades	Cuando se alcance alguno de los riesgos, dirigirá al alumno siguiendo las estrategias marcadas en cada uno de los riesgos
Cualificación	Deberá ser una persona responsable con experiencia en gestión de proyectos.

2.7.4 Categorías de riesgo

Se consideraron para los siguientes niveles las categorías de riesgo que se muestran a continuación:

- Nivel técnico
    - Requisitos (Para los riesgos causados por el fallo en los requerimientos)
    - Tecnología (Cambios en la tecnología, comportamientos inesperados de la aplicación)
    - Complejidad (Dificultad en definición de hipótesis)
- Nivel Externo
    - Regulación (Cambios en la legislación)
- Nivel Organizacional
    - Recursos (No disponibilidad de herramientas, Baja productividad)
    - Financiación (Incapacidad de financiación)
- Nivel Gestión de proyecto
    - Estimación (Retrasos en fases del proyecto, subestimación del tamaño)

2.7.5 Definiciones de la probabilidad e impacto de los riesgos

En esta tabla se estipulan los valores para el impacto y la probabilidad de un riesgo dependiendo de su coste o de su grado de probabilidad.

Se fijó un impacto catastrófico en el 50% porque se entiende que el proyecto sería inviable a partir de ese valor y crítico en torno al 20 y 50% ya que son costes que podrían hacer peligrar la finalización del proyecto.

Como probabilidades reseñables cabe decir que una probabilidad muy alta corresponderían a valores superiores al 90% y alta entre el 70 y 90% porque se entiende que a partir de esos valores la probabilidad de que ocurra un riesgo es muy posible.

Atributo	Valor	Descripción
Impacto	Catastrófico	Coste mayor del 50%
	Crítico	Coste mayor del 20% y menor del 50%
	Tolerable	Coste mayor del 10% y menor del 20%
	Bajo	Coste menor del 10% y mayor del 2%
	Muy Bajo	Coste menor del 2%
Probabilidad	Muy baja	Menor del 10%
	Baja	Mayor del 10% y Menor del 20%
	Moderada	Mayor del 20%y menor del 70%
	Alta	Mayor del 70% y menor del 90%
	Muy Alta	Mayor del 90%

2.7.6 Definición de la Matriz de probabilidad e impacto

Se utilizará la siguiente Matriz de probabilidad:

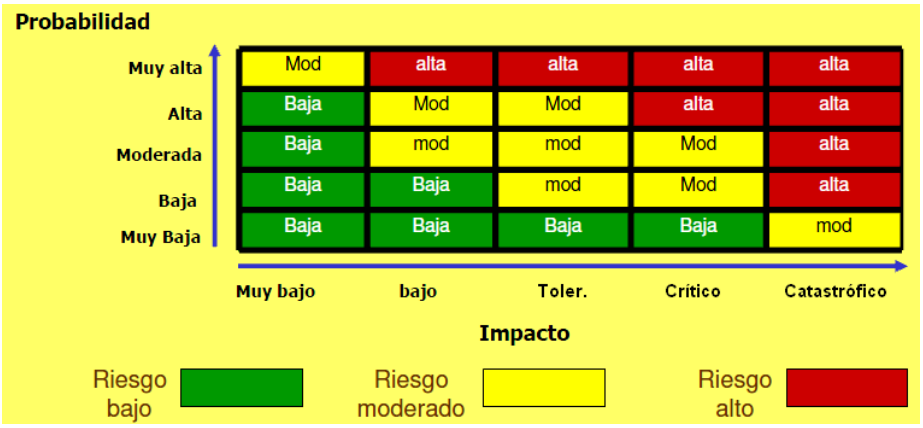


Ilustración 30

2.7.7 Formatos de los informes, del Registro de Riesgos, Informes de riesgos

Formatos utilizados en la práctica

El formato seguido para el registro de los riesgos es el siguiente:

ID Riesgo	Declaración de riesgo	Probabilidad	Impacto				Impacto global	Estrategia	Categoría	P/I
			Alcance	Calidad	Calendario	Coste				

Para el informe de cada riesgo se utilizará el siguiente formato:

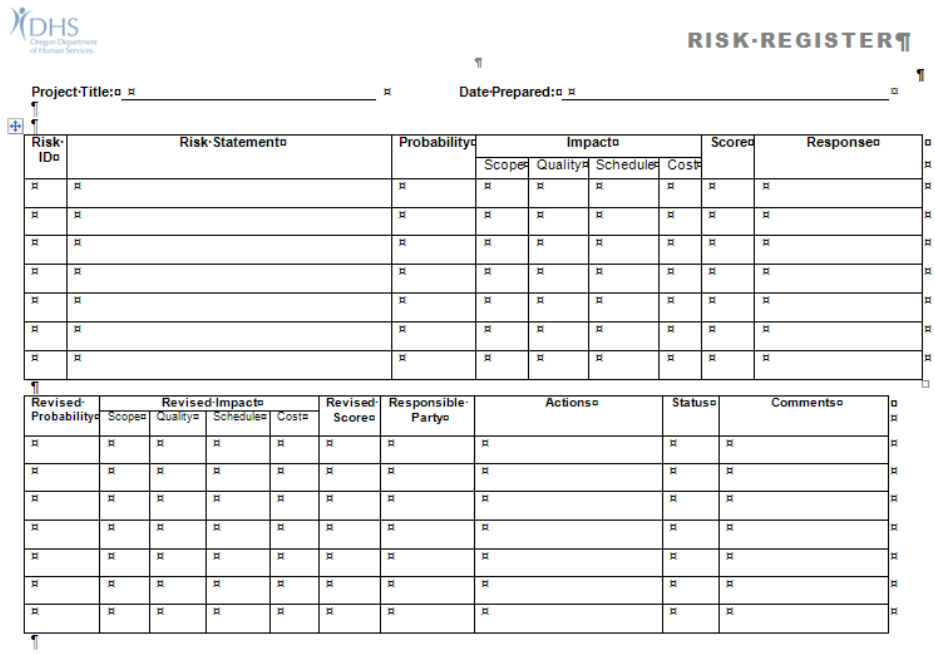
ID Riesgo:	
Nombre del Riesgo	
Categoría	
Tipo	**Clase**
Probabilidad	
Impacto	
Descripción del riesgo	
Escala Afectada por el riesgo	
Excepciones que afectan al riesgo	
-	
Variaciones del mismo riesgo	
.	
Personas o Entidades implicadas en el riesgo	

Evitación del riesgo	
Estrategia General	
Medidas Correctoras	
Monitorización	
Indicador	**Modo de evaluación del indicador**
Gestión	
Plan de Contingencias	
Consideraciones Especiales	

Las plantillas que se utilizarán en el transcurso del proyecto serán:

2.7.8 Registro de riesgos:

Conforme a la metodología en esta plantilla se establecerá un resumen de la planificación de seguridad para cada riesgo.



DHS
Oregon Department of Human Services

RISK REGISTER

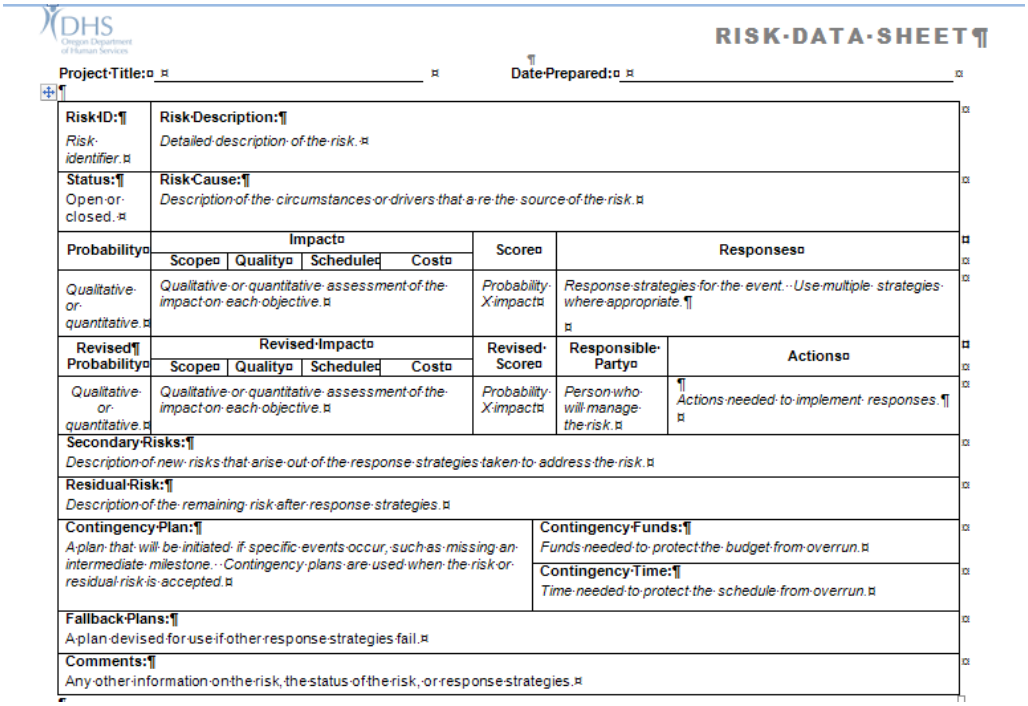
Project Title: _____ Date Prepared: _____

Risk ID	Risk Statement	Probability	Impact				Score	Response
			Scope	Quality	Schedule	Cost		

Revised Probability	Revised Impact				Revised Score	Responsible Party	Actions	Status	Comments
	Scope	Quality	Schedule	Cost					

2.7.9 Plantilla de Informe de riesgos:

Conforme a la metodología se procederá al informe de actuación de cada riesgo para implantar las medidas de seguridad.



RISK·DATA·SHEET

Project·Title:

Date·Prepared:

Risk·ID: *Risk· identifier.*	**Risk·Description:** *Detailed·description· of·the·risk.*					
Status: Open·or· closed.	**Risk·Cause:** *Description·of·the· circumstances·or·drivers·that·a·re·the· source·of·the·risk.*					
Probability	**Impact**				**Score**	**Responses**
	Scope	**Quality**	**Schedule**	**Cost**		
Qualitative· or· quantitative.	*Qualitative·or·quantitative· assessment·of·the· impact·on· each·objective.*				*Probability· X·impact*	*Response·strategies·for·the·event.··Use·multiple· strategies· where·appropriate.*
Revised Probability	**Revised·Impact**				**Revised· Score**	**Responsible· Party**	**Actions**
	Scope	**Quality**	**Schedule**	**Cost**			
Qualitative· or· quantitative.	*Qualitative·or·quantitative· assessment·of·the· impact·on· each·objective.*				*Probability· X·impact*	*Person·who· will·manage· the·risk.*	*Actions·needed· to·implement· responses.*
Secondary·Risks: *Description·of·new· risks·that·arise·out·of·the·response·strategies·taken·to· address·the·risk.*							
Residual·Risk: *Description·of·the· remaining· risk·after·response· strategies.*							
Contingency·Plan: *A·plan· that· will· be·initiated· if· specific·events·occur,·such·as·missing·an· intermediate· milestone.··Contingency·plans·are·used·when· the·risk·or· residual·risk·is·accepted.*					**Contingency·Funds:** *Funds·needed·to·protect·the· budget·from·overrun.*		
					Contingency·Time: *Time·needed·to·protect·the· schedule·from·overrun.*		
Fallback·Plans: A·plan·devised·for·use·if·other·response·strategies·fail.							
Comments: Any·other·information·on·the·risk,·the·status·of·the·risk,·or·response·strategies.							

2.7.10 Plan de Seguimiento

Cada día se revisarán los riesgos conforme avance el proyecto y en caso de que se llegue a un punto en que se deba tratar un riesgo se actualizará el estado del riesgo en el registro de riesgos y en su informe, tomando las medidas oportunas para solucionarlo; en caso de aparecer riesgos no contemplados, estos se añadirán al registro y se creará un informe del nuevo riesgo para posteriormente tomar las medidas pertinentes para su solución.

2.8 Presupuesto

En este apartado se puede observará el presupuesto de la puesta en producción así como el análisis de otros factores que pueden incrementar el precio del mismo; en este caso se contempla que no sería necesario aportar nada más que la infraestructura necesaria para alojar la web de evaluación en un host.:

Item	SubItem	Concepto	Cantidad(días)	Precio Unitario (€/día)	Total SubItems (€)	Total(€)
001		**Desarrollo de las aplicaciónes**	**302**	**150,00 €**		**45.300,00 €**
	01	Recopilación de datos (Fase I)	92	**150,00 €**	13.800,00 €	
	02	Análisis de datos (Fase II)	70	**150,00 €**	10.500,00 €	
	03	Desarrollo Generador Resumenes (Fase III)	53	**150,00 €**	7.950,00 €	
		Desarrollo Aplicación Evaluacion Fase (IV)	60	**150,00 €**	9.000,00 €	
	04	Evaluacion Previa (Fase V)	18	**150,00 €**	2.700,00 €	
	05	*Finalización de documentación y presentación del producto (Fase VI)*	*9*	**150,00 €**	1.350,00 €	
002		**Recursos tecnológicos**				**151,60 €**
	02	Hosting (duración 2 años)	24 meses	5,90€/mes	151,60 €	
003		**Recursos Software**		**(€/año)**		**2.100,00 €**
	01	Licencias Office	1	560,00 €	560,00 €	
	02	SqlSrv	1	1.140,00 €	1.140,00 €	
	02	Licencias Call SqlSrv	2	200,00 €	400,00 €	
004		**Gastos Empresa**				**750,00 €**
	01	Material Oficina			250,00 €	
	02	Imprevistos			500,00 €	
005		**Mantenimiento**		**(€/Año)**		
	01	**Mantenimiento (duración 2 años, el primero gratuito)**	**1**	**8.694,29 €**		

SUBTOTAL (I.V.A no incluido)	**56.995,89 €**
I.V.A (21%)	**11.969,14 €**
SUBTOTAL (I.V.A. incluido)	**68.965,02 €**

Ilustración 31 Presupuesto de la aplicación

En el caso de que la empresa no contase con la infraestructura necesaria para montar las aplicaciones de descarga y de generación de resúmenes, habría que determinar dónde prefiere alojar la empresa esas aplicaciones y cuánto está dispuesta a gastar.

Una buena opción en la propia empresa sería como la que aparece en la ilustración:

Habría que montar un clúster con dos servidores para conseguir alta disponibilidad con HyperV, el sistema de almacenamiento se podría montar sobre una NAS, pero habría que tener también un Firewall para controlar el tráfico ya que sería necesario que la aplicación de evaluación manejase la BBDD de la empresa.

En términos de precios de este sistema habría que incluir al presupuesto anterior lo siguiente:

- 2 Licencias de Windows Server Standard : que aproximadamente saldrían sobre 1800€/año (2178 IVA incluido)
- El almacenamiento NAS [5]costaría sobre 178€ (215.38 IVA incluido)
- Por ejemplo un Cisco ASA 5505 Next Generation Firewalls[6] costaría sobre: 770€ (931,7 IVA incluido)

5

http://www.amazon.es/gp/product/B00CDG2XHC/ref=as_li_qf_sp_asin_il_tl?ie=UTF8&camp=3626&creative=24790&creativeASIN=B00CDG2XHC&linkCode=as2&tag=gouforit-21

- En cuanto al precio de los servidores físicos hay gran variedad de precios, un ejemplo sería de un servidor Supermicro cuyo coste IVA incluido rondaría los 2500€

El presupuesto se incrementaría en unos 5825€

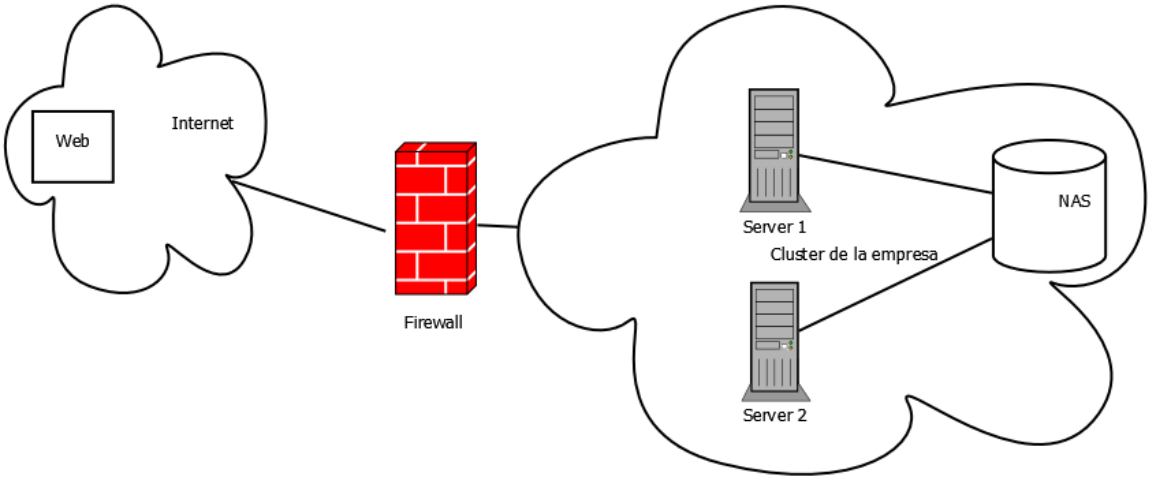


Ilustración 32 Cluster con dos servidores en la empresa

Por otra parte, también sería interesante ver la opción de cuánto costaría esta infraestructura si estuviese alojada en la nube.

Empresas como Amazon o Microsoft ofrecen este tipo de servicios:

- Amazon web Services ofrece espacios de trabajo por 55€/mes cada uno con un máximo de 100GB de almacenamiento (660 € al año)
- Microsoft Azure ofrece también multitud de opciones configurables dependiendo de las necesidades de la empresa. 2 máquinas virtuales en Windows de una categoría tirando a alta, con Soporte técnico y un ancho de banda razonable, costaría sobre 460€/Mes.(5520€ al año).

Otro de los puntos que es necesario mirar, es la estimación de costes para los resúmenes de los evaluadores, ¿cuánto costaría que un evaluador escribiese un tuitresumen?

Para realizar esa estimación, habría que fijarse en otras aplicaciones que ofrecen un servicio similar como Fiverr[7], donde sus usuarios ofrecen varios servicios de resumen a 5 dólares con gran variedad en las ofertas.

Por ejemplo salen los siguientes casos de las palabras que la gente resumiría por 5 dólares:

- 250 words (3 personas)
- 300 words (6 personas)
- 350 words (1 persona)
- 400 words (7 personas)

[6] http://www.senetic.es/product/ASA5505-SEC-BUN-K9?gclid=CPSii9yQs78CFWKWtAodMW8AvQ
[7] http://www.fiverr.com/categories/writing-translation/quality-translation-services/#layout=lists&page=1

- 500 words (11 personas)
- 550 words (1 persona)

(0.02 * 3 + 0.0167 * 6 + 0.0143 * 1 + 0.0125 * 7 + 0.01 * 11 + 0.009 * 1) / 29

De lo que se puede extraer que aproximadamente se gastará lo siguiente por palabra:

(0.06 + 0.1002 + 0.0143 + 0.0875 + 0.11 + 0.009) / 29 = 0.0131 $ por palabra

Dándose por supuesto que se trata de palabras en inglés (5.1 caracteres de media) y sabiendo que un tuit promedio tendría: 23 palabras (117.3 caracteres + 22 blancos) significaría que escribir un tuit costaría un máximo: de 0.30 $

Para la lectura[8] se ha estimado en base a costes de traducción (y dividiendo por 2 pues requiere escritura).

Ejemplo :

- 250 words (1 persona)
- 300 words (2 personas)
- 350 words (1 persona)
- 400 words (1 persona)
- 500 words (1 persona)
- 600 words (4 personas)
- 750 words (1 persona)
- 1000 words (5 personas)

Se calcularía el precio de la lectura de cada palabra:

(0.02 + 0.0167 * 2 + 0.0143 + 0.0125 + 0.01 + 0.0083 * 4 + 0.0067 + 0.005 * 5) / 16 =

(0.02 + 0.0334 + 0.0143 + 0.0125 + 0.01 + 0.0332 + 0.0067 + 0.025) / 16 =0.0097 $ / word

Obteniendo que la lectura de una palabra costaría 0.0097 $.

Suponiendo 23 palabras por tuit y dividiendo por 2 (sólo lectura): 0.112 $/tuit

Sugeriría enfocar la tarea evaluación de la siguiente forma:

Leer entre 50 y 100 tuits y escribir un máximo de 5 tuits. El precio por tarea sería entonces de: 5.90$ (leer 50 y escribir 1) a 12.7$ (leer 100 y escribir 5),

- El caso peor, suponiendo 10 trending topics diferentes cada 15 minutos, los costes de mantenimiento serían de 12192 $/día. (sí, doce mil)
- El caso mejor serían 10 trending topics de los más baratos y que durarían 24 horas: 59 $/día.

[8] http://www.fiverr.com/categories/writing-translation/quality-translation-services/#layout=lists&page=1

La estimación más realista sería realizar la media de ambos que serían unos 6125,5 $/día.

En euros unos 4.502 €/día.

Por todo esto se tendría que estipular dentro de la empresa cual va a ser la cantidad de evaluaciones que está dispuesta a financiar y si realmente se prefiere disponer de la infraestructura en la empresa o contratada como un servicio en la nube.