ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

Titulación :

INGENIERO EN INFORMÁTICA

Título del proyecto:

NUEVAS HERRAMIENTAS PARA EL ANÁLISIS DE OPINIÓN EN FLUJOS DE TEXTO

Alumno:

Vicente Hidalgo Santamaría

Tutor:

José Ramón González de Mendívil

Pamplona, 19 de Febrero de 2015

Este proyecto se ha llevado a cabo mientras el alumno realiza unas practicas a través de la Fundación Universidad-Sociedad, de la Universidad Pública de Navarra, en la empresa TweetBinder.

Presentación

La minería de opinión se ha convertido en uno de los objetivos de las grandes plataformas de Internet, analistas y entes gubernamentales que desean conocer las opiniones de los usuarios que hacen uso de ellas.

En este proyecto se ha implementado un sistema de procesamiento de flujos de textos procedentes de Twitter, basado en el uso de algoritmos de análisis de sentimiento. Se pretende analizar las diferentes opciones existentes hasta ahora. Previamente a ésta implementación se han estudiado algunos de los diferentes métodos y sistemas que existen actualmente: Storm, Trident, Spark. Además se han estudiado algunos de los métodos más utilizados en análisis de sentimiento. Finalmente se ha decidido implementar un entorno de análisis de sentimiento de datos de Twitter. El desarrollo y la investigación de esto proyecto han seguido el siguiente curso:

- **Análisis del estado del arte en minería de sentimientos.**

El primer paso a la hora de comenzar este proyecto ha sido recopilar la documentación existente sobre:

  - Métodos y algoritmos de detección de eventos.
  - Métodos y algoritmos de análisis de Sentimiento en texto.
  - Implementaciones existentes de análisis de sentimiento.

Posteriormente, la investigación y principalmente el desarrollo se han llevado a cabo sobre los siguientes entornos y sistemas.

- **Entornos de procesamiento de Streaming.**

  - Apache Storm y su entorno de ejecución mediante topologías.
  - Trident y el procesamiento mediante micro-batching
  - Apache Spark Streaming como la alternativa a Hadoop.
  - Nodejs como posible entorno de desarrollo y ejecución para la base de un sistema de procesamiento.

- **Desarrollo e implementación**

Se escogido el entorno Nodejs por las ventajas y rapidez que ofrece a la hora de desplegar un sistema y establecer la comunicación entre sus partes.
Éste no es un sistema definitivo, porque es posible implementar muchas funcionalidades. Sin embargo, se ha asentado una posible base o prototipo para los posibles nuevos sistemas de análisis de opinión en Streaming.

Contenido

En los siguientes puntos se desarrolla una explicación detallada con los diferentes pasos y métodos que se han seguido en este proyecto. Esta memoria esta organizada de la siguiente forma. En este primer punto se explica el contexto actual de las redes sociales, el análisis del sentimiento y los retos que supone aplicarlo. En el capítulo 2, se exponen algunas de las técnicas y algoritmos de análisis de sentimiento más utilizados actualmente. El capítulo 3 explica el estado del arte en técnicas de procesamiento de flujos de texto (Streams), así como las que se utilizan en este proyecto. El capítulo 4 expone las características más importantes de nodejs para este proyecto, ya que es lo que se va utilizar para el desarrollo final. El capítulo 5 presenta la arquitectura del desarrollo que se ha realizado y su funcionamiento.

Capítulo 1 - INTRODUCCIÓN

A continuación se explica detenidamente los diferentes aspectos en los que se basa este proyecto. Principalmente, las redes sociales y el uso que se hace actualmente de ellas, la importancia del análisis de sentimientos y los retos que supone su implementación.

1.1 Las redes sociales

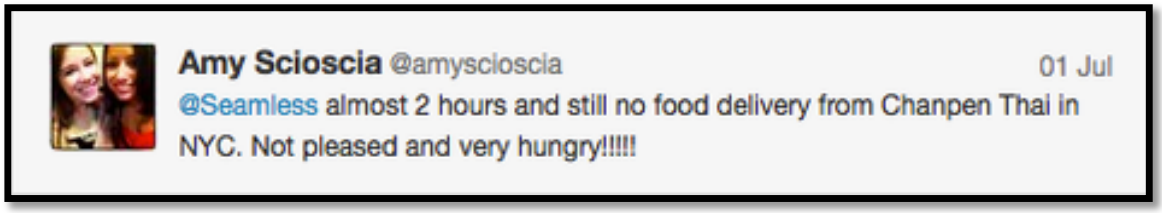
Las redes sociales han llegado a convertirse en la principal plataforma de publicación para los usuarios de internet. Se han convertido en el canal de publicación, no sólo de noticias y sucesos, sino en el principal medio para expresar las opiniones sobre la mayoría de los temas que atañen a las personas. De todas las plataformas existentes actualmente, es Twitter la que más ha crecido en menos tiempo. Actualmente el número de usuarios que participan activamente es de 284 millones, pero el alcance y potencial de esta red no se queda ahí. Ha quedado demostrado que es la forma de difusión mas rápida, permitiendo obtener información, a quién la consulta, acerca de lo que está sucediendo en cualquier parte del mundo en cualquier momento.

Las ventajas y posibilidades que las plataformas de micro-blogging ofrecen son múltiples. También lo son los posibles usos que se pueden hacer de ellas en diferentes áreas, por ejemplo, comercio, análisis, gobierno y finanzas. Además casi todas las compañías con presencia online utilizan las redes sociales como un canal directo de comunicación con sus clientes y usuarios. En el mundo del marketing online, la red se ha convertido en un nicho de mercado interesante y fructífero que permite ampliar los frentes de acción. En el ámbito de los gobiernos cada vez tiene más peso el conocer los sucesos y opiniones con la mayor rapidez posible.

A continuación se presentan algunos ejemplos de mensajes extraídos de Twitter:

Ejemplo de publicación en Twitter. Acción publicitaria de una compañía de agua con azúcar. [21]

Ejemplo de publicación en Twitter. Una empresa que comercializa libros electrónicos anuncia un concurso a través de la red. [22]



Ejemplo de publicación en Twitter. Una clienta se queja del mal servicio recibido por parte de una cadena de comida. [23]

Ejemplo de publicación en Twitter. Un ciudadano egipcio manifiesta su enfado y promueve un levantamiento contra el gobierno de su país. [24]

Llegados a este punto cabe preguntarse si hay más formas de explotar esos datos y obtener información útil a partir de ellos. Esta cuestión es la clave para las estrategias que se están desarrollando hoy en día.

1.2 Sentiment

El análisis de polaridad de la opinión es uno de los campos con más potencial de desarrollo. Gracias a los avances en temas de web semántica se ha llegado a ver la importancia y valor de las opiniones que se publican. Muchas de las campañas que se lanzan actualmente al publico en internet, tienen como objetivo a los usuarios de plataformas como Twitter. Los métodos más utilizados de interacción suelen ser la creación de etiquetas concretas o la publicación de mensajes originales con el objetivo de atraer la atención de los usuarios e de invitarles a publicar sobre dicho tema.

Sirva de ejemplo el lanzamiento de un nuevo producto por parte de un empresa. Posteriormente a su colocación en el mercado, resulta más fácil que nunca recoger las opiniones de los directamente datos públicos que ofrecen las redes. Esto se conoce con el termino minería de opiniones.

Dada la gran cantidad de publicaciones y la forma desordenada en que se producen, conviene desarrollar métodos para extraer métricas que puedan

resultar útiles dependiendo de cada caso concreto. El contenido más útil para este tipo de desarrollo se encuentra en forma de textos. En este trabajo, el estudio se centra en la minería de opiniones sobre la plataforma que ofrece Twitter.

1.2.1 ¿Qué es?

Sentiment analysis [1] (análisis de sentimientos) es la tarea de extracción de opiniones y emociones en general. La principal función del Sentiment analysis es la clasificación de emociones.

Dentro de los posibles resultados que puede ofrecer un análisis de sentimiento, el primero y más básico aporta información acerca del carácter objetivo o subjetivo de un fragmento de información. Posteriormente, el análisis debe proporcionar datos acerca de si el fragmento expresa una opinión positiva o negativa, en el caso de ser de carácter subjetivo.

Este proyecto se centra en las características que aportan información acerca de la opinión subjetiva. Existen otros proyectos que intentan profundizar más en las emociones intentando clasificarlas, miedo, angustia, satisfacción, etc.

1.2.2 ¿Para qué sirve?

La principal utilidad de la información que se pretende estudiar, consiste en una serie de valores que ilustran el sentimiento común a un grupo de fragmentos o presente individualmente en uno. La idea es que esta diferenciación aporte nueva información.

Conocer la opinión del publico, clientes o ciudadanos acerca de un aspecto o aspectos de forma inmediata es la mejor forma de saber qué es lo que está sucediendo en cualquier momento y lugar, y qué es lo que las personas opinan. Este conocimiento permite actuar de forma inmediata para llevar a cabo la acción que fuera necesaria. En este aspecto, el análisis automático puede ser de gran utilidad a varios tipos de entidades, por ejemplo:

- Tiendas online que desean conocer las opiniones que los usuarios publican después de comprar un producto.

- Fabricantes y empresas de servicios que quieren conocer el grado de aceptación de sus productos o servicios.

- Gobiernos y entes sociales que desean saber acerca de la situación social y opinión de la población.

- Cadenas de televisión, para conocer la opinión de los espectadores acerca de una determinada emisión.

Por ejemplo, En 2013 la multinacional Unilever lanzó una campaña [2] a través de la red social Twitter para conocer el grado de aceptación de un nuevo producto que había salido al mercado. Según el informe de procesamiento de redes sociales, que presenta la compañía, obtuvieron una precisión del 90% en los resultados de su clasificación.

1.3 Retos del Sentiment Analysis

El análisis de la información se ha convertido en una de las disciplinas más importantes dentro del uso comercial que se les da a las redes sociales. Recientemente han aparecido nuevas metas para la investigación y el desarrollo, permitiendo ampliar las métricas que existen actualmente. Cada una de estas líneas nuevas de investigación aportan a su vez nuevos retos.

1.3.1 Léxico y semántica

Léxico es el conjunto de palabras que conforman un determinado lenguaje y, por extensión, también se denomina así a los diccionarios que los recogen.

En el caso de Twitter, el uso de lenguajes diferentes es clave ya que se encuentra disponible en la mayoría de países. Dentro de cada uno de estos lenguajes la cantidad de variaciones posibles es muy grande, por lo que es necesario llegar a un consenso respecto a lo que aporta significado y lo que no.

El vocabulario de un idioma refleja el medio físico y social de sus hablantes porque es un inventario de las ideas, los intereses y las ocupaciones de la comunidad. Las lenguas se adaptan a las preocupaciones, los intereses y las vivencias de los hablantes. Conocer una lengua es conocer el léxico, saber usarlo, y también conocer y saber utilizar las reglas que controlan la combinación correcta de los elementos.

El término semántica (del griego semantikos, "lo que tiene significado") se refiere a los aspectos del significado, sentido o interpretación de signos lingüísticos como símbolos, palabras, expresiones o representaciones formales. En principio cualquier medio de expresión (lenguaje formal o natural) admite una correspondencia entre expresiones de símbolos o palabras y situaciones o conjuntos de cosas que se encuentran en el mundo físico o abstracto que puede ser descrito por dicho medio de expresión.

El punto de partida es pues, el conjunto de palabras de un texto. Las palabras correspondientes a un idioma son fácilmente recopilables en un diccionario. El problema surge al combinarlas para formar frases y con éstas, expresar opiniones. Estas frases tienen un significado que depende primeramente del significado de las palabras, y posteriormente, de la forma en la que se combinan. El significado puede verse afectado en cualquier momento de manera radical al introducir una nueva palabra. Este es el reto de la extracción de significados que plantea el partir de un contexto léxico para extraer una semántica.

1.3.2 Escalabilidad

El problema de la escalabilidad no es trivial y resulta sumamente importante.

El sistema debe poder admitir y procesar un número variable de mensajes. Esto significa que en cualquier momento, el rendimiento, de cara al usuario y los resultados, debe ser el mismo.
El gran reto en esta dimensión es uno de las características más interesantes de las redes sociales, en concreto y sobre todo, de Twitter. Cualquier desarrollo que pretenda tratar la información obteniéndola de una fuente de Streaming, debe estar preparada para un crecimiento repentino de la velocidad de llegada de información.

De forma paralela al crecimiento del flujo de los datos de entrada, debe estar planteado un posible crecimiento, de forma que el trabajo de procesamiento se realice a la vez sobre uno o más mensajes simultáneamente. Esta característica es necesaria si se quiere mantener la escalabilidad vertical, ya que se el sistema debe admitir la capacidad que requiera un Streaming de flujo variable para ser procesado.

1.3.3 Vulnerabilidad a ataques

Cualquier sistema de las características del que se platea en este proyecto es susceptible de recoger datos maliciosos que hayan sido generados con el fin de alterar los resultados. Esta vulnerabilidad es inherente a cualquier sistema que haga uso de los datos que los usuarios hacen públicos a través de una plataforma de las características de Twitter. Esto se debe a la capacidad de publicar cualquier contenido, que no viole la política de la aplicación, tendrá la misma acogida en el sistema que los demás.

Para hacer frente a este tipo de contratiempos sería necesario implementar una política de contenidos de manera que cada publicación sea revisada previamente a su entrada en la fase de análisis. Cualquier tipo de mensaje publicado con afán de deformar los resultados resulta un ataque a este tipo de análisis.

1.3.4 Forma y métricas de evaluación del sistema

A la hora de comprobar el rendimiento y veracidad de los datos que ofrece un sistema existen varias medidas diferente. Estas medidas de comparación funcionan de la siguiente manera que se ilustra con un ejemplo.

		Observación real Positivo	negativo
Predicción Esperada	Positivo	**a**	**b**
	Negativo	**c**	**d**

Sea **a** el número de mensajes correctamente clasificados como positivos, **b** el número de mensajes negativos clasificados como positivos, **c** el número de mensajes positivos clasificados como negativos y **d** el número de mensajes negativos clasificados correctamente.

A la hora de comparar y evaluar un método consideramos las siguientes métricas:

- Ratio de positivos acertados: $R = \frac{a}{a+c}$
- Ratio de positivos fallados: $P = \frac{c}{a+c}$
- Precisión: $A = \frac{a+d}{a+b+c+d}$
- Medida-F: $F = 2 \cdot \frac{P \cdot R}{P+R}$

Esta medida-F es la medida de precisión que tiene un test. Se emplea en la determinación de un valor único ponderado de la precisión y la exhaustividad. Se suele emplear en la fase de pruebas de algoritmos de búsqueda, recuperación de información y clasificación de documentos.

1.3.5 Interdisciplinaridad

El análisis de texto no es una tarea que pueda ser realizada desde un punto de vista únicamente computacional. Existen muchos y muy diferentes aspectos dentro de este campo de estudio que requieren de opiniones y desarrollos expertos:

- **Lingüística**

El objetivo de un estudio lingüístico es describir las lenguas caracterizando el conocimiento tácito que de las mismas tienen los hablantes y determinar cómo éstos las adquieren. En el análisis de sentimiento, es de vital importancia el significado de los fragmentos de información que se analizan. Por ello, parte del trabajo requiere de un estudio lingüístico conocedor de las características de la lengua que se esta estudiando en cada caso.

- **Sociología**

A la par que el conocimiento lingüístico necesario, se encuentra la necesidad de conocer el entorno, formas y circunstancias en las que se crea el contenido que se va a analizar. La sociología resulta de vital importancia para conocer y tener en cuenta los aspectos sociales que rodean al emisor.

- **Matemáticas (Estadística)**

Cuando existen muchas y muy diferentes maneras de evaluar y cuantificar datos como los que emite un análisis de sentimiento, es necesario regular la forma en que estos son tenidos en cuenta para que los resultados finales aporten información y sean útiles.

- **Sistemas distribuidos**

Cualquier sistema que tenga como objetivo el análisis de algo tan volátil como el contenido en las redes sociales, requiere de un planteamiento paralelo y distribuido en su arquitectura. La escalabilidad es un aspecto fundamental y necesario.

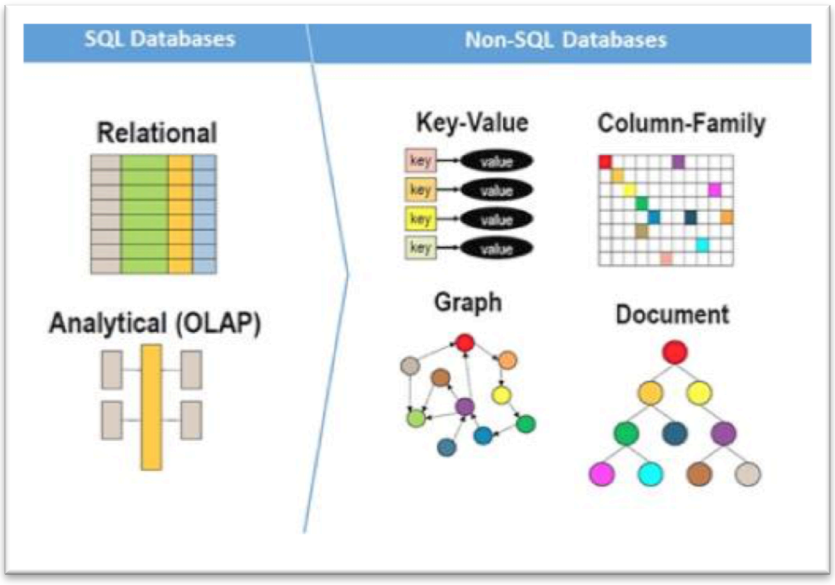
1.3.6 Almacenamiento

Para el tratamiento de los datos es necesaria una solución escalable, resistente y operacionalmente eficiente. El crecimiento web cada vez avanza más rápido y esto hace necesario eliminar las restricciones que supone el uso de los sistemas de bases de datos más antiguos.

Este tipo de problemas suele presentarse a menudo en las soluciones de procesamiento web. En seguida surgen interrogantes sobre qué almacenar y qué descartar. Debido a que los datos son obtenidos de un API multipropósito (sirve información de la forma más general posible) podemos encontrar por cada mensaje un sinfín de datos útiles que sin embargo no están destinados a ser almacenados.

Para aquellos datos que sí requieren ser almacenados la solución más común en este tipo de esquemas de funcionamiento es el uso de bases de datos. Los datos, tratándose de un red social, suelen corresponder a perfiles de usuarios y mensajes que éstos han publicado junto con los propios de los mensajes: número de Retweets y favoritos, numero de usuarios alcanzados por el mensaje, hora de emisión, etc.

El mayor problema en esta fase se encuentra en la elección del sistema de base de datos más apropiado. No-SQL suele ser el empleado para aplicaciones que hacen uso de Twitter debido a que el esquema de los documentos no es fijo. El uso de una base de datos relacional tiene sentido cuando los elementos de las colecciones siguen un esquema relacional fijo.



Diferencias entre sistemas de BBDD

1.3.7 Generación de informes

La información que un sistema de análisis de sentimiento puede proporcionar puede llegar a ser confusa y contraproducente si no se define bien cada conjunto de datos. Por ello, es importante que los resultados, gráficas e indicadores aporten una idea más abstracta que los propios datos, ya que es extremadamente fácil caer en una mala interpretación.

A la hora re presentar los datos, el método elegido influye en la calidad de los resultados. El mayor problema dentro de este aspecto, es la posible confusión y la llegada a conclusiones erróneas. Véase el ejemplo de un mensaje clasificado como negativo. Resulta de poca utilidad dentro del posible análisis que se puede hacer de él con los métodos basados en diccionarios o redes neuronales. Los resultados pueden ser menos precisos, pero si más acertados, si de ese mensaje se extraen conceptos y éstos son catalogados según su polaridad, resulta mucho más fácil trabajar ya que se crea un contexto para el análisis. Este tipo de análisis es el que realiza, por ejemplo, el método SenticNet que se explica más adelante.

Capítulo 2 - TÉCNICAS DE PROCESAMIENTO DE SENTIMIENTOS

Existen múltiples métodos de minería de sentimientos, entre éstos se incluyen los basados en el léxico y los que se basan en el aprendizaje. A pesar de la popularidad de algunos de ellos, no esta claro cuál es el mejor para identificar la polaridad de un mensaje de texto. A continuación se exponen algunas de las técnicas más utilizadas. Los tres métodos más comunes de aproximación al tema del análisis de sentimientos son:

- Machine Learning (ML)
- Natural Language Processing (NLP)
- Information Retrieval (IR)

Estas tres formas no son mutuamente excluyentes. Es más, lo normal y óptimo es la aplicación de varias de ellas para obtener unos resultados más sólidos que los que se obtendrían usando sólo una de ellas.

2.1 Emoticonos

El uso de estos caracteres se ha extendido hasta estar presente en la mayoría de dispositivos y medios. Son la manera más rápida y fácil de expresar sentimientos en cualquier contexto. Actualmente la forma más popular son los de la familia llamada *emoji.* Estos ideogramas provenientes de Japón ya están incluidos en el estándar *Unicode.*

A la hora de evaluar el contenido de sentimiento de un mensaje, esta analiza el tipo de emoticono. Sin embargo, el uso de éstos solamente se encuentra en el 10% de los mensajes. Además, está el problema de que la codificación varía según el dispositivo y la aplicación. Sin embargo, en un fututo muy cercano, gracias a la estandarización, será posible su clasificación.
[3]

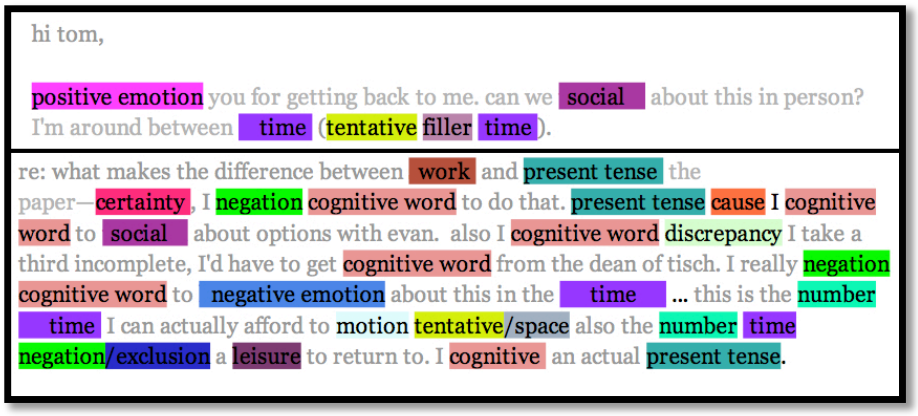
2.2 Linguistic Inquiry and Word Count (LIWC)

Es una herramienta de análisis emocional para textos. Se basa en el estudio emocional y estructural de un texto mediante diccionarios de palabras positivas y negativas. Actualmente, es posible aplicar este método a mensajes en varios idiomas, por ejemplo, Árabe, Chino, Inglés y Español.

Este método, propuesto por investigadores de la universidad de Austin (USA), está diseñado para poder utilizar colecciones de palabras y categorías tanto aportadas por el usuario como por el propio método, permitiendo una mejor adaptabilidad a los casos que se puedan plantear. Pretende calcular el grado en el que las personas usan diferentes categorías de palabras en textos como: emails, discursos, poemas o prensa escrita.

LIWC [4] evalúa los componentes emocional, cognitivo y estructural de un texto basándose en diccionarios que contienen palabras clasificadas según categorías. Además de detectar sentimientos positivos y negativos, LIWC ofrece otros tipos de categorías, por ejemplo, “acuerdo” pertenece a las siguientes categorías: asentimiento, afecto, emoción positiva, proceso cognitivo.

En la siguiente imagen se muestra el análisis que sea ha realizado, mediante LIWC, del texto de un email. Se han sustituido las palabras clasificadas según una categoría, o estructura, por el nombre de esta.



Análisis de correo electrónico (LIWC)

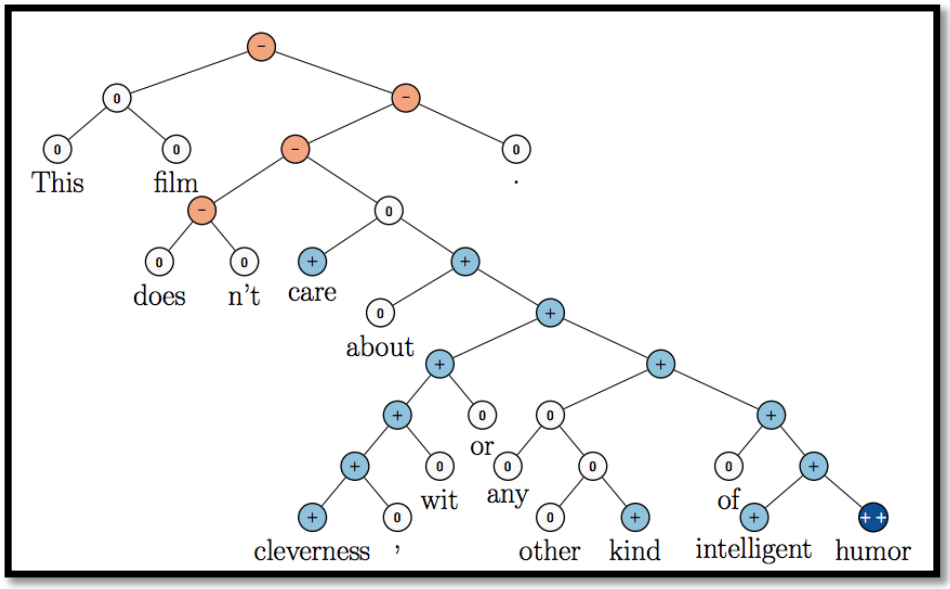
2.3 Staford Core-NLP

Core-NLP (Natural Language Processing) [5] es un conjunto de herramientas para el procesamiento de texto desarrolladas en la universidad de Stanford. Dada una entrada de texto, ofrece funcionalidades como: extracción de palabras y conjuntos clave, extracción de partes del discurso, normalización de fechas, horas y cantidades numéricas, y extracción de sentimientos. Esta última funcionalidad es la que interesa y se ha estudiado.

A diferencia de métodos anteriores basados en grupos semánticos de palabras, este método aporta el concepto de Sentiment Treebank. Incluye 215,154 frases, etiquetadas según su clasificación, de un total de 11,855 documentos.

Gracias al uso de una red neuronal recursiva, es capaz de clasificar, con una precisión del 85%, frases en varias categorías: muy negativa, negativa, positiva, muy positiva o neutra. Aunque en este proyecto se utiliza solamente la clasificación básica, positiva, negativa y neutra.

A continuación se incluye el árbol, correspondiente a una clasificación negativa mediante Core-NLP. Se puede apreciar la predicción que se hace en cada nodo según los colores (rojo para negativo y azul para positivo) y el momento en que se detecta una negación.



Ejemplo de clasificación mediante la red neuronal recursiva.

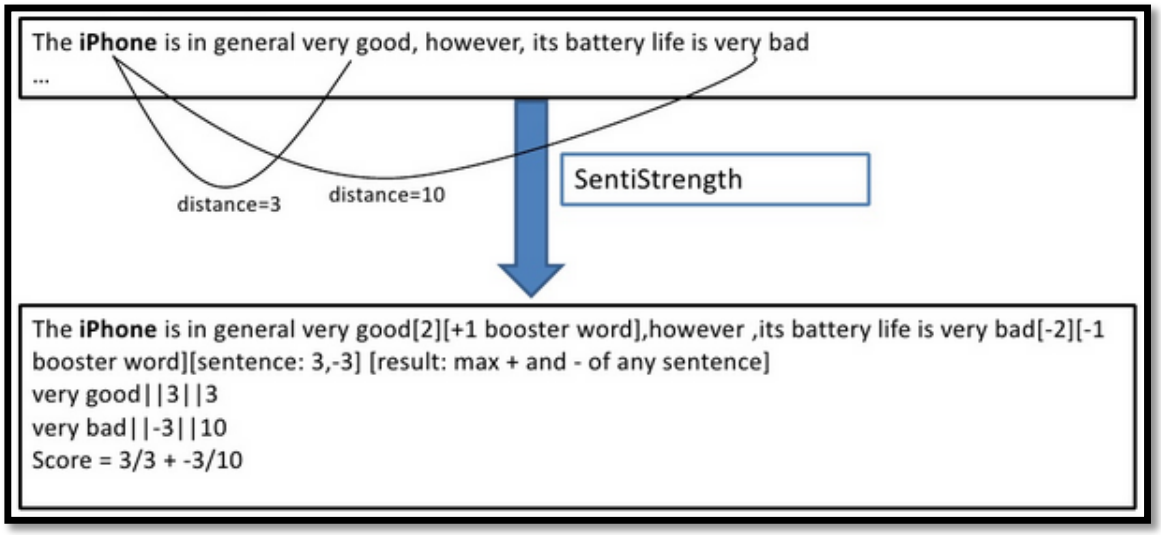
2.4 SentiStrength

Es una herramienta escrita en Java para el análisis de sentimiento. Ha sido desarrollada por el consorcio CyberEmotions, que agrupa investigadores de varias universidades europeas. Se encuentra disponible para el uso académico y comercial, con uso de licencia. SentiStrength [6] estima el grado de sentimiento en pequeños textos y es capaz de analizar el lenguaje coloquial. El grado de sentimiento que produce esta herramienta se expresa en una escala del 1 al 5 para sentimientos positivos, y de -1 a -5 para sentimientos negativos. Además, es capaz de detectar la neutralidad de un texto.

Este método combina el anteriormente expuesto y el aprendizaje mediante *Machine Learning*. SentiStrength amplia la funcionalidad de LIWC mediante el uso de palabras potenciadoras para clasificar mensajes con contenido muy positivo o muy negativo.

Para su desarrollo y aprendizaje se han utilizado mensajes de seis medios diferentes como Twitter, Digg, MySpace, el foro online de la cadena BBC, Runners Wordl Forum y comentarios de YouTube.

A continuación, se incluye una imagen en la que se puede ver como este método separa las partes del discurso (POS) y evalúa cada una. Además, se aprecia como tiene en cuenta las palabras potenciadoras.



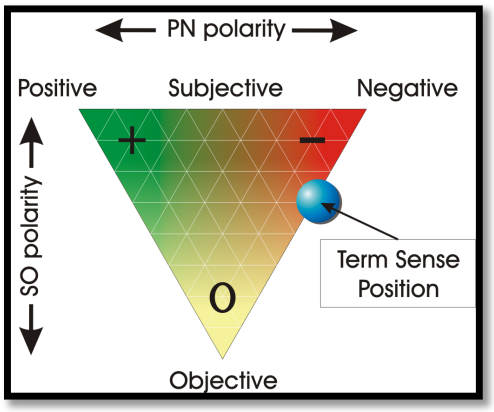
Desarrollo en árbol de análisis con SentiStrength

2.5 SentiWordNet

SentiWordNet [7] es una de las herramientas más empleadas en el campo de la minería de opiniones. Está basada en el uso del diccionario léxico en inglés llamado WordNet, el cual agrupa, adjetivos, nombres, verbos y otras clases gramaticales en sinónimos, llamadas *Synsets.*

El método está basado en un análisis cuantitativo de fragmentos que tiene asociados Synsets y en la representación vectorial de una clasificación semi-supervisada de estos conjuntos. Se obtienen tres calificaciones a partir de combinar los resultados de ocho clasificadores ternarios que tienen en común niveles similares de precisión aunque diferentes comportamientos.

A continuación se muestran las dimensiones de la clasificación de un termino mediante SentiWordNet. La polaridad de cada termino se expresa con tres valores: Objetividad, negatividad y positividad.



Dimensiones de la clasificación SWN

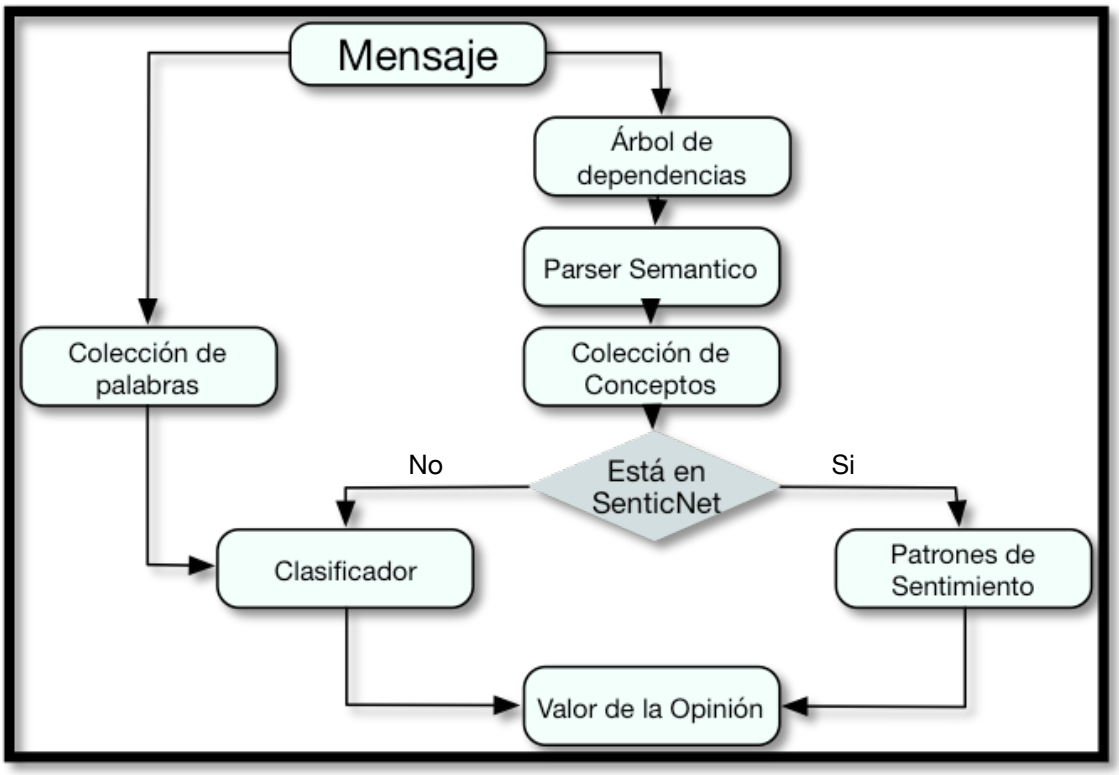
2.6 SenticNet

SenticNet [8] es un método de minería de opinión y análisis de sentimiento principalmente enfocado a la web semántica. El objetivo de esta herramienta es inferir la polaridad a través de conceptos del lenguaje natural escrito. Es decir, extraer el sentimiento de las ideas o conceptos en los que se centra el mensaje.

SenticNet consta de un extractor de conceptos y de una librería con hasta 14000 conceptos clasificados según su positividad o negatividad.

El funcionamiento del algoritmo SenticNet es diferente a los demás analizados en cuanto a que realiza una abstracción previa de los mensajes antes de clasificarlos. Resumidamente, la clasificación sigue los siguientes pasos:

1. Desarrollar un árbol de dependencias a partir del mensaje de entrada.
2. Separar las componentes semánticas del mensaje.
3. Extraer conceptos mediante la búsqueda de las componentes semánticas en una colección de conceptos previamente clasificados según su polaridad.
4. Aplicar agrupación sobre todas las componentes semánticas analizadas y conceptualizadas correctamente.



Algoritmo SenticNet

Capítulo 3 - ANALISIS DE STREAMING

A partir del año 2011 se empieza a aplicar la minera de datos y se crea el termino BigData. Al final de la primera década de este siglo se llega a la conclusión de que todo lo almacenado hasta la fecha son datos en crudo, que aportan poco pero pueden ser de más valor si son analizados. Estos documentos, bases de datos, registros, perfiles, etc. Pueden tener algún fin a parte de mantenerse almacenados. Se desarrolla entonces el concepto de Minería de Datos.

La minería de datos (*Data Mining)* ha sido hasta ahora un tema a tener en cuenta en la mayoría de plataformas con masas grandes de usuarios. Sin embargo, cuando ya se han procesado todas las grandes bases de datos, se han analizado todos los ficheros y extraído toda la información útil. Lo más probable es que el proceso vuelva a empezar.

Se ha hecho mucho énfasis en la importancia de los grandes volúmenes de datos estáticos y se ha infravalorado el potencial de los flujos como son el Streaming y las publicaciones online.

Ahora mismo, y cada vez más, es posible el conocimiento de los hechos en tiempo real. Ya sea para un evento de presentación de una marca, para una campaña de publicidad o para conocer la última hora acerca de un atentado terrorista. La acción inmediata salva vidas y genera ganancias.

3.1 Diferentes herramientas

Para afrontar la tarea del análisis de un flujo de mensajes de entrada. Lo primero y más necesario es especificar los resultados que queremos a la salida y el tratamiento para los datos de entrada. Esto se debe a que si, por ejemplo, el sistema no garantiza el procesamiento de todos los mensajes, se deberá tener esto en cuenta al entregar los resultados. Por ello, existen diferentes sistemas y técnicas de los cuales se explican los más significativos en las siguientes secciones.

3.2 Storm

Storm [9] es un sistema de procesamiento en tiempo real que fue desarrollado por BackType para Twitter cuando esta red comenzaba a crecer. Más tarde el proyecto se liberó y ahora se distribuye y desarrolla bajo licencia Apache. Storm es escalable, tolerante a fallos y garantiza que todos los datos de entrada serán procesados.

Aunque según las especificaciones, Storm puede ser utilizado a través de cualquier lenguaje de programación, esto sólo se aplica a los módulos de procesamiento, es decir, a topología.

Storm se ejecuta en topologías. Estas topologías o distribuciones de procesamiento pueden distribuirse a lo largo de una o varias máquinas en una red (Cluster). Dichas topologías tienen opciones de rebalanceo para dedicar más recursos y aumentar el paralelismo en aquellas áreas de procesamiento que soporten más carga. Cabe destacar que este rebalanceo se puede realizar en tiempo de ejecución.

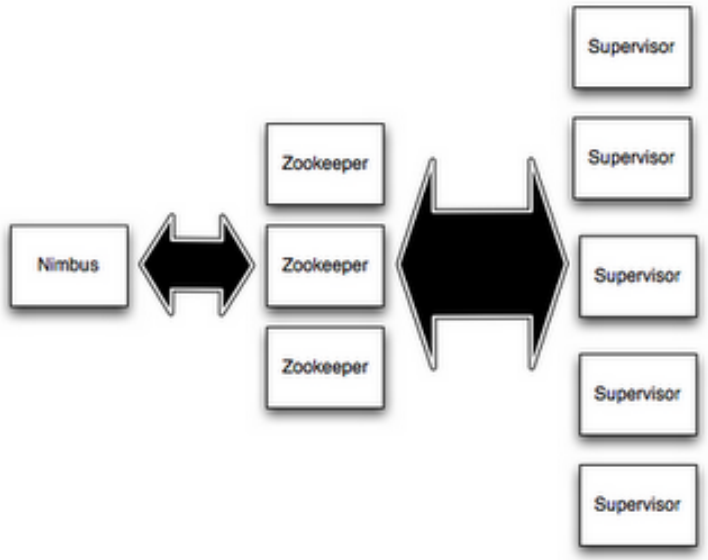
3.2.1 Componentes de un cluster Storm

Un sistema cluster Storm es, a un alto nivel, similar a un cluster Hadoop, con la diferencia de que en vez de trabajos map-reduce, realiza procesamiento en topologías. La principal diferencia entre una tarea map-reduce y una topología es que en esta ultima no hay estados. En map-reduce, cuando una tarea se acaba, el hilo de ejecución termina. En Storm, el flujo permanece constantemente abierto, por eso se denomina a este sistema *state-less*. Existen dos tipos de nodos en Storm:

Master: es el nodo maestro que mantiene en ejecución un programa llamado Nimbus, que en Hadoop sería *JobTracker.* Nimbus es el encargado de distribuir el código a lo largo del cluster, asignar tareas a máquinas y monitorizar errores.

Worker: Cada uno de los nodos Worker de un cluster Storm ejecuta un programa llamado *Supervisor*. Este programa esta continuamente escuchando la llegada de tareas nuevas para el nodo. Su misión principal es recibir las tareas y ejecutarlas en el nodo.

En resumen, cada nodo del cluster se encarga de la ejecución de una parte de la topología. Una topología puede estar repartida en tantos nodos como se desee. Finalmente está el programa que se encarga de la coordinación de la comunicación entre Nimbus y el *Supervisor*.



Procesos de Apache Storm

Una topología es un grafo de computación en el que cada nodo contiene una parte de la lógica de procesamiento y una configuración para dirigir el flujo de los datos de salida hacia otros nodos.

Dentro de una topología de Storm podemos encontrar dos tipos de nodos diferentes. Estos se ejecutan cuando se emite la topología a producción.

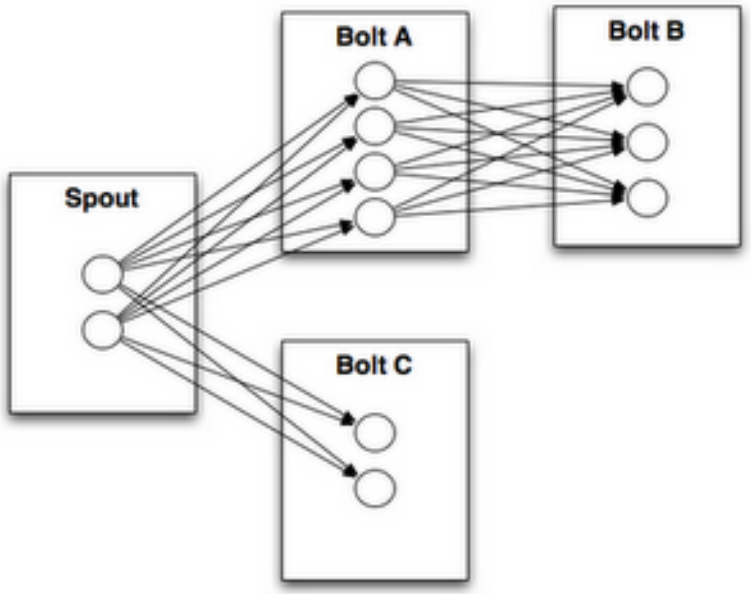
Spout: Son aquellos nodos de entrada para los datos. Estos pueden recoger información de casi cualquier fuente. Su principal ventaja es la adaptabilidad.

A partir de las últimas entregas, Storm incluye Apache Kafka. Un cluster distribuido de encolamiento que se distribuye bajo licencia Apache y que puede ser alimentado a través de una red o desde otras aplicaciones.

Kafka puede soportar una gran cantidad de datos de entrada y gracias a su escalabilidad puede mantener su funcionamiento de forma eficiente. Dentro de Kafka el flujo de entrada puede ser particionado y repartido alrededor de las máquinas que componen el cluster.

Una de las opciones para uno de estos nodos es la de implementar una entrada para los datos de Twitter. Mediante este nodo serían distribuidos entre los diferentes nodos internos de la topología los mensajes que van a ser procesados.

Bolt: Son aquellos nodos que realizan el procesamiento. Estos pueden estar programados en cualquier lenguaje. Sin embargo, es necesario que dentro de la topología se realicen las llamadas necesarias para que las entradas y salidas de cada nodo de procesamiento sean recogidas en el sistema.



Distribución de una topología

3.2.2 Análisis de Storm

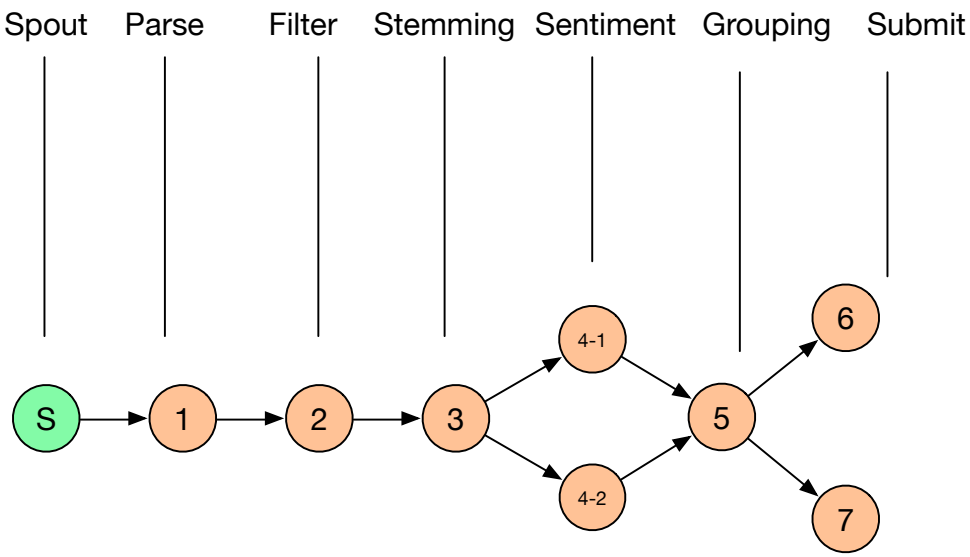
Storm fue considerado la alternativa a Hadoop para realizar tareas de procesamiento en tiempo real durante un tiempo. Sin embargo, finalmente se descartó para la implementación de este proyecto debido a su inestabilidad, dificultad de implantación y constantes cambios críticos de una versión a otra.

Puede llegar a ser, una vez su desarrollo esté completo, el Framework más apropiado para un proyecto de estas características. Sin embargo, tras realizar varias pruebas con él, queda claro que su estado de desarrollo es más bien el de una herramienta inmadura.

La documentación que se puede encontrar acerca de Storm es más bien poca y en la mayoría de los casos solamente se centra en algún aspecto muy concreto. De ahí que en la mayoría de entornos Storm que se encuentran en producción se cuenta con ingenieros que previamente han trabajado en Twitter.

Storm resulta, para concluir, un concepto de herramienta muy interesante y que sin embargo se ha quedado atrás respecto a otras por falta de intensidad en su desarrollo. La comunidad activa no es muy grande y los arreglos que requiere una herramienta como ésta, en fase *Beta,* suelen tardar varios meses.

A continuación se muestra un ejemplo del código principal que realiza un análisis mediante diccionario de los mensajes que van llegando al sistema. Finalmente, los mensajes junto con sus valoraciones se envían a un almacenamiento HDFS a través de una conexión con nodejs. El siguiente mapa de distribución muestra las fases que sigue la topología. La numeración de los nodos es la misma que en el código.



Topología de análisis de sentimiento

```
//Topología
private static StormTopology createTopology()
{   //entrada de los tweets mediante un cluster kafka
    SpoutConfig kafkaConf = new SpoutConfig(
        new ZkHosts(Properties.getString("rts.storm.zkhosts")),
        KAFKA_TOPIC,
        "/kafka",
        "KafkaSpout");
    kafkaConf.scheme = new SchemeAsMultiScheme(new StringScheme());
    TopologyBuilder topology = new TopologyBuilder();
    //Declaración del spout
    topology.setSpout("kafka_spout", new KafkaSpout(kafkaConf), 4);
    //Primer Bolt: Parseo de los tweets y creación de tuplas (key,value)
    topology.setBolt("twitter_filter", new TwitterFilterBolt(), 4)
            .shuffleGrouping("kafka_spout");
    //Segundo Bolt: Filtrado del texto del tweet
    topology.setBolt("text_filter", new TextFilterBolt(), 4)
            .shuffleGrouping("twitter_filter");
    //Tercer Bolt: Eliminación de palabras inútiles
    topology.setBolt("stemming", new StemmingBolt(), 4)
            .shuffleGrouping("text_filter");
    //Cuarto Bolt-Rama-1: Recuento de palabras positivas
    topology.setBolt("positive", new PositiveSentimentBolt(), 4)
            .shuffleGrouping("stemming");
    //Cuarto Bolt-Rama2: Recuento de palabras negativas
    topology.setBolt("negative", new NegativeSentimentBolt(), 4)
            .shuffleGrouping("stemming");
    //Quinto Bolt: Agrupación de resultados
    topology.setBolt("join", new JoinSentimentsBolt(), 4)
            .fieldsGrouping("positive", new Fields("tweet_id"))
            .fieldsGrouping("negative", new Fields("tweet_id"));
    //Sexto Bolt: Asignación del resultado final al tweet
    topology.setBolt("score", new SentimentScoringBolt(), 4)
            .shuffleGrouping("join");
    //Septimo Bolt: Envío del tweet evaluado a almacenamiento HDFS
    topology.setBolt("hdfs", new HDFSBolt(), 4)
            .shuffleGrouping("score");
    //Octavo Bolt: Envío del tweet evaluado a una conexión con node
    topology.setBolt("nodejs", new NodeNotifierBolt(), 4)
            .shuffleGrouping("score");
    return topology.createTopology();
}
```

Topología de análisis de sentimiento

3.3 Storm Trident

Trident es una capa de abstracción que realiza operaciones con Storm como base. Se trata de utilizar Storm a un nivel más alto donde la distribución del código entre nodos de entrada o de procesamiento no son tareas que deba hacer el desarrollador, sino que Trident las lleva a cabo de la forma más optima posible. Esta capa de abstracción resulta especialmente útil cuando se sabe que el procesamiento se va a llevar a cabo en tiempo real, pero la distribución de las funciones no.

Véase el siguiente ejemplo de código en el que se realiza una cuenta de las palabras que van apareciendo en las frases que entran en el sistema.

```
//Nueva topología Trident
TopologyBuilder builder = new TopologyBuilder();
//Asignación del Spout: Emite frases aleatorias
builder.setSpout("spout", new RandomSentenceSpout(), 5);
//Primer Bolt (función): Separa las frases en palabras
builder.setBolt("split", new SplitSentence(), 8).shuffleGrouping("spout");
//Segundo Bolt (función): Cuenta las palabras agrupando
builder.setBolt("count", new WordCount(), 12).fieldsGrouping("split", new Fields("word"));

Config conf = new Config();
conf.setDebug(true);
```

Implementación wordCount en Apache Spark

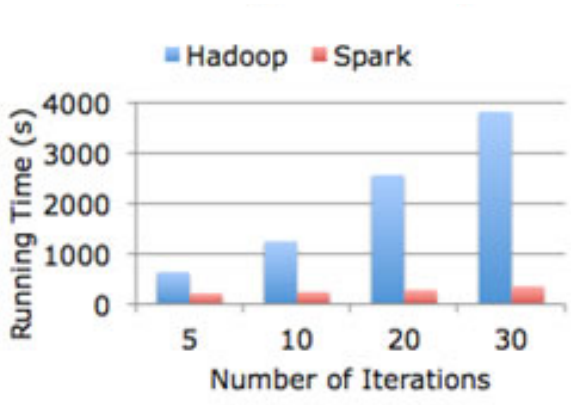
3.4 Spark

Apache Spark [10] es un sistema abierto de procesamiento cluster, desarrollado y mantenido principalmente en la universidad de Berkeley. Se trata de un sistema enfocado a trabajos Map-Reduce con la principal diferencia de que se hace todo en memoria. Spark supera a su antecesor, ya que las operaciones que antes suponían un cuello de botella, por los tiempos de acceso al disco en almacenamiento HDFS, pasan a ser mínimos.

Spark está desarrollado principalmente en Scala ya que permite mantener datasets distribuidos y manipulados como si fuesen colecciones almacenadas en forma local.

Una de las características interesantes de Apache Spark es que ofrece interactividad durante la minería de datos, resultando especialmente útil para la implementación de algoritmos de aprendizaje, análisis y cálculo.

Las diferencias [11] de rendimiento que se van al utilizar Spark en tareas en las que antes se usaba Hadoop son destacables. En la siguiente gráfica se puede ver las proporciones de el cambio que supone Spark.



Comparativa de rendimiento en una implementación de regresión logística Hadoop-Spark

3.4.1 Spark Streaming

Una de las extensiones más interesantes de Spark es Spark Streaming. Esta añade el procesamiento continuo a las funcionalidades que se pueden lograr.

El procesamiento continuo parece el siguiente paso evidente que el *Big Data* debe tomar. Spark Streaming es la respuesta. Manteniendo la tolerancia a fallos realiza *Batch-Processing* en memoria.

La principal diferencia entre Spark y Spark Streaming es que mientras en la primera se realiza *Batch-Processing* en la segunda las operaciones son *Micro-Batching*. Esto puede resultar contradictorio ya que realmente no se trata de Streaming de datos, sin procesamiento de porciones mucho más pequeñas.

El siguiente fragmento de código corresponde a una implementación de un sistema de análisis de sentimiento sencillo en Apache Spark. Su funcionamiento es muy similar la visto con Storm.

```
//Filtrado del texto de los Tweets
JavaDStream<Tuple2<Long, String>> tweetsFiltered = filtered.map(
    new TextFilterFunction());
//Separación de los componentes y eliminación de palabras inútiles
tweetsFiltered = tweetsFiltered.map(
    new StemmingFunction());
//Análisis de palabras positivas
JavaPairDStream<Tuple2<Long, String>, Float> positiveTweets =
    tweetsFiltered.mapToPair(new PositiveScoreFunction());
//Análisis de palabras negativas
JavaPairDStream<Tuple2<Long, String>, Float> negativeTweets =
    tweetsFiltered.mapToPair(new NegativeScoreFunction());
//Agrupación de tweets analizados
JavaPairDStream<Tuple2<Long, String>, Tuple2<Float, Float>> joined =
    positiveTweets.join(negativeTweets);
JavaDStream<Tuple4<Long, String, Float, Float>> scoredTweets =
    joined.map(new Function<Tuple2<Tuple2<Long, String>,
                                Tuple2<Float, Float>>,
                            Tuple4<Long, String, Float, Float>>() {
    private static final long serialVersionUID = 421;
    @Override
    public Tuple4<Long, String, Float, Float> call(
        Tuple2<Tuple2<Long, String>, Tuple2<Float, Float>> tweet)
    {
        return new Tuple4<Long, String, Float, Float>(
            tweet._1()._1(),
            tweet._1()._2(),
            tweet._2()._1(),
            tweet._2()._2());
    }
});

JavaDStream<Tuple5<Long, String, Float, Float, String>> result =
    scoredTweets.map(new ScoreTweetsFunction());
//Envío a almacenamiento HDFS
result.foreachRDD(new FileWriter());
//Envío mediante conexión nodejs
result.foreachRDD(new HTTPNotifierFunction());
```

Análisis de sentimiento. Implementación en Apache Spark

3.5 Nodejs

Nodejs es un entorno de programación en la capa del servidor basado en el lenguaje de programación ECMAScript, asíncrono, con entrada y salida de datos en una arquitectura orientada a eventos y basado en el motor V8 de Google. Fue creado con el enfoque de ser útil en la creación de programas de red altamente escalables, como por ejemplo, servidores web.

La arquitectura dirigida por eventos, Event-driven Architecture, es un patrón de arquitectura software que promueve la producción, detección, consumo de, y reacción a eventos.

Este patrón arquitectónico puede ser aplicado para el diseño e implementación de aplicaciones y sistemas que transmitan eventos entre componentes software que estén emparejados libremente y servicios. Un sistema dirigido por eventos está compuesto típicamente de emisores de eventos y consumidores de eventos. Los consumidores tienen la responsabilidad de llevar a cabo una reacción tan pronto como el evento esté presente. La reacción puede o no puede ser completamente proporcionada por el consumidor en sí mismo. Por ejemplo, el consumidor debe tener solamente la responsabilidad de filtrar, transformar y reenviar el evento a otro componente o debe proporcionar una reacción propia a algún evento.

Al contrario que la mayoría del código JavaScript, no se ejecuta en un navegador, sino en el servidor. Nodejs implementa algunas especificaciones de CommonJS.5. Nodejs incluye un entorno REPL para depuración interactiva.

A pesar de que inicialmente se pensaba implementar un sistema de análisis de flujos de texto (Streams) basado en el leguaje de programación Java sobre la plataforma Storm. Finalmente se escogió Nodejs dada la velocidad de ejecución, facilidad de escalado y el gran numero de librerías que están actualmente en desarrollo y mantenidas por la comunidad.

Hasta ahora la mayoría de desarrollos basados en nodejs estaban destinados a servicios web. De hecho, ha crecido enormemente la popularidad de este arquitectura en los últimos años gracias a la rapidez con que se producen estos servicios.

Sin embargo, es poca la cantidad de desarrollos que utilizan las ventajas del paralelismo en Nodejs. Llegados a este punto parece contradictorio que un motor que computa según una cola de procesamiento vaya a realizar computación paralela. Esto es posible gracias que en las últimas versiones se ha añadido la funcionalidad para poder crear procesos hijos que realizan tareas de forma paralela. Este punto se explicará con más detalle en el siguiente capitulo.

3.6 Comparativa

La mayoría de las comparativas en términos de eficiencia a la hora de analizar sistemas de procesamiento se centran en aspectos como el rendimiento o el flujo alcanzable. Sin embargo, en este caso se realiza una comparativa con el punto de vista en la escalabilidad y la fiabilidad de los sistemas. Esto se debe a que se ha llegado a tal capacidad computacional hoy en día, que la eficiencia viene marcada, más que por la velocidad de procesamiento, por la capacidad de adaptación de un sistema respecto a un flujo variable en la entrada de los datos.

Storm:

- Se ha convertido en una de las opciones más seguras en lo que se refiere a Stream Processing y cuenta con una mayor cuota de adopción que el resto de las herramientas .
- Es la única que realiza procesamiento en tiempo real, propiamente dicho.
- Cuenta con la ventaja sobre las demás alternativas de que soporta topologías programadas en cualquier lenguaje, a diferencia que Spark, Trident o Nodejs que son más limitados en este sentido.
- Asegura que el procesamiento de cada mensaje se realiza por lo menos una vez, aunque no asegura que se vaya a hacer solamente una.

Trident:

- Como capa de abstracción para Storm, es una gran aportación al mundo del Stream Processing.
- Aporta una metodología para poder realizar micro-batching sobre Storm. De esta forma se pierde el procesamiento real time, pero se consigue la seguridad de procesamiento de Storm con una metodología de desarrollo mucho más sencilla.
- Se está convirtiendo en la forma más común de utilizar Storm, hasta el punto de que parece que el desarrollo futuro se va a centrar casi completamente sobre él.
- Asegura que realiza el procesamiento una y solo una vez de cada mensaje.

Spark:

- Es la herramienta definitiva para reemplazar a Hadoop.
- Cuenta con las ventajas del batch-processing y elimina los problemas de lectura y escritura de este.
- Sólo es posible implementar algoritmos en Java y Scala.
- No asegura el procesamiento del 100% de las entradas ya que algunos de los posibles errores en la programación de los workers que pueden dejar fuera parte de la información.

Nodejs:

- se ha convertido en la alternativa en programación y funcionamiento seria con grandes posibilidades y mucho desarrollo.
- No cuenta todavía con desarrollos computacionales al nivel de complejidad e intensidad.
- El diseño no bloqueante y orientado entradas y salidas resulta especialmente interesante para Real Time Processing.

Finalmente se ha decidido utilizar Nodejs para realizar la implementación que inicialmente se pensaba hacer con Apache Storm. Esto se debe a que este último se encuentra en un estado de desarrollo en el que todavía no es planteable para el entorno que se propone aquí. Nodejs en cambio plantea muchos aspectos que pueden hacer posible desarrollar un entorno previo en el que estén muchas de las características de Storm.

Capítulo 4 - NODEJS

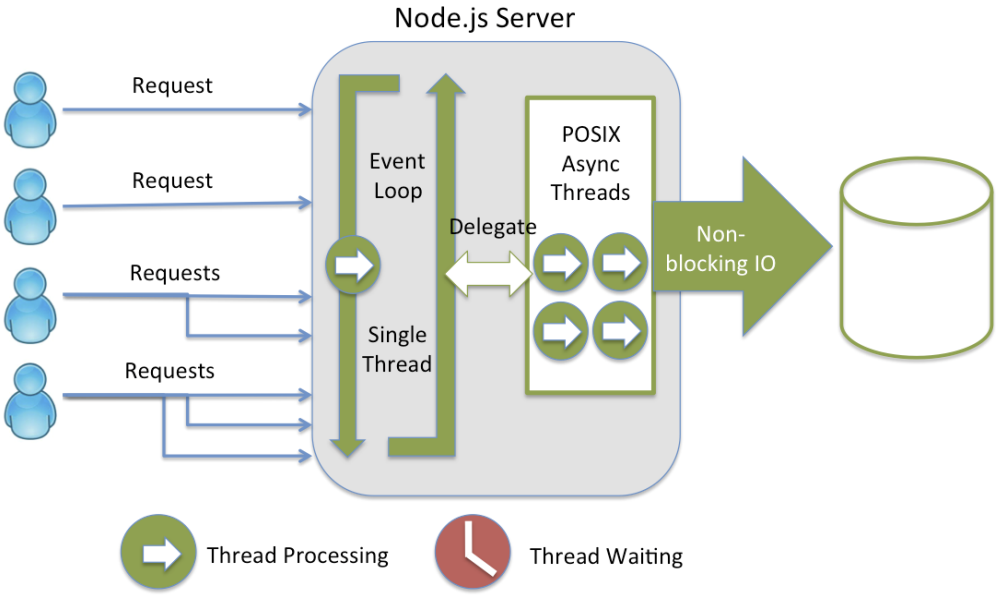
Puede parecer a primera vista que el utilizar algo tan relativamente nuevo como Nodejs es un movimiento erróneo. Sin embargo, dada la velocidad de crecimiento que ha tenido parece que seguramente llegue a asentarse como una de las herramientas interesantes de programación de entornos Cluster.
[12]

Resulta en cierto modo arriesgado este punto de vista, sin embargo, en la metodología que se propone en este desarrollo se implementa un ejemplo funcional de cómo la integración entre java, nodejs y otras plataformas puede resultar altamente beneficiosa.

Uno de los principales motivos por los que se ha escogido Nodejs como la base de la programación de este proyecto es la facilidad con la que los diferentes elementos de un cluster se pueden definir y desplegar, manteniendo siempre la comunicación entre ellos con un paso de mensajes sencillo y asequible para cualquier tipo de máquina o red que vaya soportar el sistema.

Indudablemente un sistema de análisis de sentimiento conlleva una carga computacional, grande, que no debe ser menospreciada y puede requerir gran parte de los recursos del sistema. Ésta parte se ha estimado suficientemente importante como para implementarla en otros lenguajes de programación más tradicionales.

En la siguiente imagen, se explica de forma esquemática el funcionamiento interno de Nodejs.



Arquitectura multiproceso (Nodejs)

4.1 Principales propiedades

La característica más importante del proceso y herramientas elegidas es el hecho de que la totalidad de las acciones que desencadenan procesos, son disparadas por entradas en el sistema.

Nodejs es un entorno orientado a eventos de entrada/salida. Esto, junto con el hecho de que todos los procesos son no-bloqueantes permite ejecutar el procesamiento según el flujo de entrada real que haya en cada momento.

A partir de la versión 0.10.36 se incluye dentro de las funcionalidades de nodejs la posibilidad de clustering multiproceso. Esto es, la posibilidad de crear procesos *hijos* desde un proceso padre y delegar en ellos el trabajo. Esta funcionalidad es clave en este trabajo y se explica con más detalle en los siguientes puntos.

4.2 Integración con sistemas, APIS

Una de las utilidades más importantes de los servicios web actuales es la posibilidad de servir herramientas accesibles de forma externa.

Este sistema de acceso a los datos se ha extendido muy rápidamente en el ámbito de las aplicaciones web hasta el punto de que hoy en día la mayoría cuentan con un sistema API para la consulta de datos aportados por los usuarios y la interacción con las funcionalidades de la aplicación.

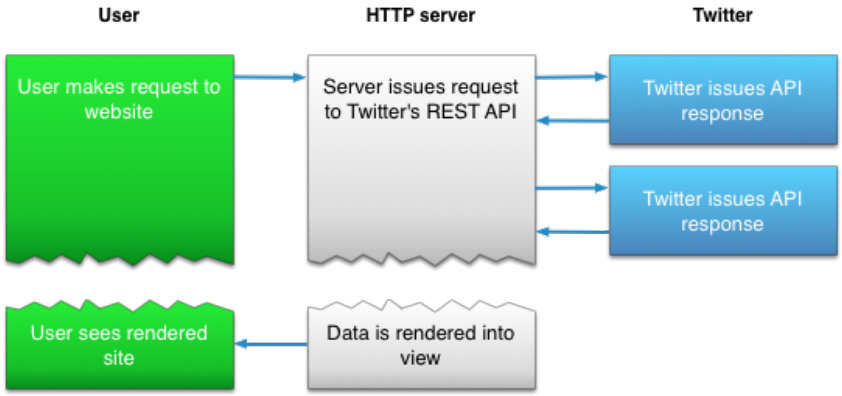
- **Twitter:**

En este caso la API con la que se va a interactuar, es la que ofrece Twitter gratuitamente. Existen limitaciones en esta API ya que su uso comercial esta dirigido a través de Gnip, empresa dedicada a comercializar los datos de Twitter.

Es importante destacar que las operaciones de lectura de un Streaming no son viables para desarrollar un producto comercial y destinado al gran público. Por ello este proyecto se ha implementado pensando en la escalabilidad a lago plazo y en la futura implementación del Streaming de Gnip, la plataforma de pago de Twitter.

- **REST**:

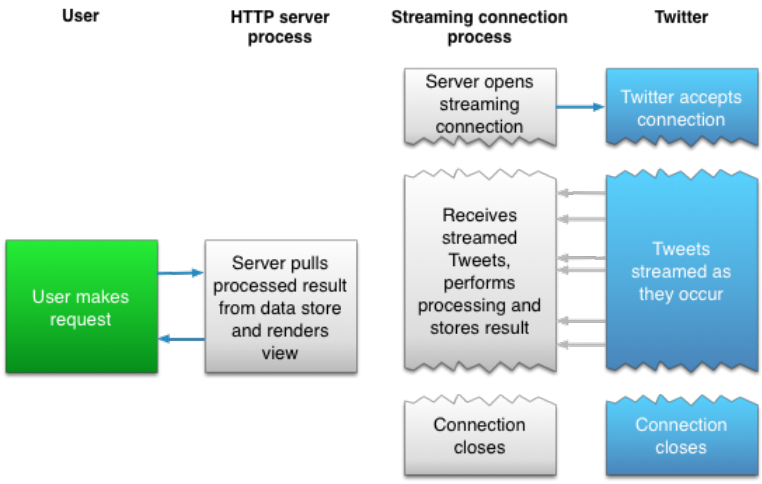
La API REST que ofrece gratuitamente Twitter consiste en una serie de métodos de llamada que proporcionan acceso limitado a los mensajes publicados hasta la fecha según unos de búsqueda. Cuenta con ciertas limitaciones que impiden su uso comercial, por ejemplo, solamente permite realizar doce llamadas por minuto.
[13]



Twitter REST API

- **Streaming:**

Este servicio está orientado a la monitorización de unos términos fijos o el seguimiento de la actividad de usuarios concretos.
La limitación de esta API consiste en que para cambiar las palabras que se están monitorizando es necesario cerrar la conexión desde el lado del cliente. Estas conexiones y reconexiones están penadas, y de manera arbitraria pueden ser cerradas ,desde el lado de Twitter, si superaran las 150 por minuto.
[14]



Twitter Streaming API

- **GNIP:**

Como se expone anteriormente, GNIP es la sección de Twitter que se dedica a la comercialización de datos. Ofrece las posibilidades de buscar mensajes con un mes de antelación y Streaming de términos ilimitados.
[15]

4.3 Comunicación entre nodos y modularidad

Uno de los puntos clave de este proyecto es la modularidad. Se ha tenido en cuenta en la mayoría de funcionalidades este concepto hasta el punto en que cada una de las partes funciona de forma independiente a las demás. Esto significa que: de suceder una caída en alguno de los módulos esto no afectaría a la información que está siendo procesada en el sistema en ese momento.

El punto de la comunicación se explica detenidamente más adelante ya que tiene mucho que ver con el uso de los módulos de Nodejs.

La modularidad de este proyecto parte de la separación de funcionalidades en procesos independientes que reaccionan a los mensajes que les llegan desde otras partes de la aplicación o del exterior.

El paso de mensajes es una de las partes más importantes en el desarrollo de una aplicación nodejs ya que el flujo de los datos se produce con el intercambio de objetos entre las partes de la aplicación.

Al tratarse éste de un desarrollo nodejs, el lenguaje que se utiliza en la parte principal es JSON (JavaScript Object Notation). Mediante JSON se pueden enviar objetos directamente, ya que no hace falta parsearlos antes o después de enviarlos. En la mayoría de partes de este desarrollo el paso de objetos se hace tal cual los objetos son generados.

4.4 Características útiles de Nodejs

Además de las posibilidades de desarrollo modular y escalable que ofrece nodejs hay que desatacar el gran numero de librerías (módulos) que actualmente existen y se encentran bajo supervisión de la comunidad open source. El uso de estás librerías, en la mayoría de los casos, es libre y se encuentran disponibles a través de la herramienta NPM (Node Package Manager) que incluye nodejs. Podría compararse a Apache Maven por sus funcionalidades de organización y gestión de paquetes utilizados dentro de un proyecto. A continuación se explica detenidamente los módulos que más funcionalidad han aportado a este desarrollo.

4.4.1 Socketio

En el ámbito de la comunicación entre procesos, el componente más útil en el proceso de desarrollo ha sido la librería Socket.io.
[16]

Socket.io es una librería en JavaScript para Nodejs que permite una comunicación bidireccional en tiempo real entre cliente y servidor. Para ello se basa principalmente en Websocket pero también puede usar otras alternativas como sockets de Adobe Flash, JSONP polling o long polling en AJAX, seleccionando la mejor alternativa para el cliente en tiempo de ejecución.

Casi todas las funciones de comunicación entre los módulos que componen este desarrollo utilizan la librería socket.io ya sea como servidores o como clientes.

Sirva de ejemplo la comunicación entre en servidor web, que mantiene la página para la interacción con el usuario, se conecta con el modulo que monitoriza la API de Streaming de Twitter.

```
var _TWTR_port = 8003;
var clientio = require('socket.io-client');
var rules = ["Madrid", "London"];
var _TWTR_server = clientio.connect('http://localhost:'+_TWTR_port);
_TWTR_server.emit('filter', rules);
```

En este ejemplo se ve como se realiza una conexión mediante socket.io a un servidor del que se conoce su dirección y puerto. A éste se le envía el objeto *rules* cuyo contenido es un *array* de palabras para iniciar una conexión al Streaming de Twitter.

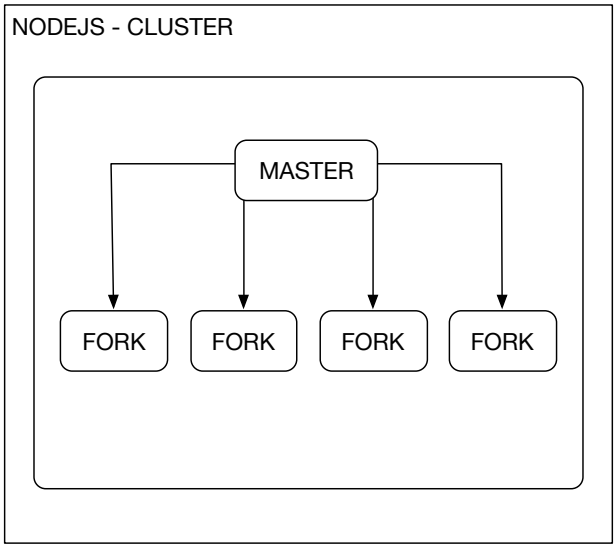
En el otro lado de la conexión, la gestión y respuesta al evento *rules* es igual de sencilla que en la llamada inicial:

```
var _TWTR_port = 8003;
var io = require('socket.io').listen(_TWTR_port);
//espera conexiones
io.sockets.on('connection', function (socket) {
//cuando le llega una nueva palabra para filtrar
    socket.on('filter', function (rules) {
        console.log("twitter_filter: " + rules);
//empieza una nueva conexion al stream de twitter con la palabra que ha llegado
    credentialsTwitter.stream('statuses/filter',
        {
            'track': rules
        },
        function (stream) {...});
    });
});
```

4.4.2 Cluster

Hasta hace relativamente poco tiempo. Una de las razones por las que se valoraba nodejs como una opción de menos rendimiento que las demás existentes para clustering, era la carencia de capacidades multiproceso. Esto se ha solucionado en las ultimas versiones, que ya incluyen la función Cluster.

Cluster [17] permite ejecutar varias instancias de nodejs funcionando de forma paralela y utilizando todos los procesadores de que dispone la maquina que lo ejecuta. Gracias a esta funcionalidad se ha conseguido un desarrollo multiproceso capaz de realizar procesamiento paralelo.



Multiprocesamiento en Nodejs

Los procesos hijos se comunican con el padre vía IPC (interprocess communication) y a través de esta conexión reciben los mensajes que deben procesar.

El módulo Cluster admite dos tipos diferentes de distribución de los mensajes. Round-Robin, en el que el proceso padre envía mensajes a los hijos de forma secuencial y otro en el que el nodo padre crea un socket de escucha y se lo envía a cada uno de los hijos. Aunque este segundo método parezca el más eficiente, se ha escogido el primero porque en este último la carga se acaba repartiendo de forma muy poco balanceada después de un periodo largo de funcionamiento.

Algunos de los métodos y acciones con cluster que se han utilizado son:

- Cluster.isMaster: permite comprobar en tiempo de ejecución si el proceso es el proceso principal.
- Cluster.isWorker: permite comprobar en tiempo de ejecución si el proceso es un proceso hijo.

- Worker.sendd: Permite mandar mensajes a lis procesos hijos. De esta forma es como se les envía la información que deben procesar.
- Cluster.on('exit'): permite ejecutar una acción cuando uno de los hijos finaliza su ejecución inesperadamente (muere).

Este conjunto de utilidades abre muchas puertas de cara a desarrollos que en un principio parecían imposibles en un entorno como nodejs.

4.4.3 Node-twitter

Este proyecto se basa en el análisis de los datos provenientes de Twitter. En concreto de la API de Stream. Por tanto, la parte dedicada a recoger esos datos es de vital importancia. El funcionamiento de la API se explica en puntos anteriores.
[18]

Gracias al módulo node-Twitter es posible mantener una escucha continua a de la llegada de los mensajes a la aplicación de forma muy sencilla.

```
var client = new nTwitter({
    consumer_key: process.env.TWITTER_CONSUMER_KEY,
    consumer_secret: process.env.TWITTER_CONSUMER_SECRET,
    access_token_key: process.env.TWITTER_ACCESS_TOKEN_KEY,
    access_token_secret: process.env.TWITTER_ACCESS_TOKEN_SECRET,
});

client.stream('statuses/filter', {track: 'twitter'},  function(stream){
    stream.on('data', function(tweet) {
        console.log(tweet.text);
    });

    stream.on('error', function(error) {
        console.log(error);
    });
});
```

Uso de la API de Streaming de Twitter

4.4.4 Express

Express es el módulo que realiza las funciones de servidor web. Se ha convertido en una alternativa a Apache bastante fiable. La facilidad con que permite mantener un servidor web resulta especialmente útil en un contexto en el que la mayoría de las comunicaciones dentro de la aplicación se realizan mediante sockets.
[19]

Al no ser la parte de Front-end el punto fuerte de este proyecto, se ha desarrollado un portal con las funcionalidades justas para probar el entorno y permitir al usuario realizar consultas.

Capítulo 5 - DESARROLLO E IMPLEMENTACIÓN

El desarrollo que se ha llevado a cabo en este proyecto consiste en un sistema distribuido para analizar el sentimiento de los mensajes provenientes de la API de Streaming de Twitter. Como se ha explicado anteriormente, se han estudiado muchas opciones. Finalmente se ha decidido utilizar el entorno Nodejs para crear un sistema similar en funcionamiento a los vistos: Storm y Spark. El diseño modular en el que se basa este desarrollo consta de varias partes diferenciadas, aunque se comuniquen entre ellas.

Este desarrollo tiene como objetivo servir de referencia (prototipo) para los futuros sistemas que se podrán desarrollar siguiendo la metodología que se describe aquí. La funcionalidad principal es proveer de un sistema de procesamiento del Stream de Twitter, así como demostrar las posibilidades de Nodejs como alternativa a los sistemas más utilizados. Además se ha implementado un pequeño portal web para probar las funcionalidades del sistema a través una interfaz clara y sencilla.

5.1 Funcionalidad

El sistema puede explicarse en dos fases claramente diferenciadas. Estas son Back-End y Front-End. El desarrollo principal de este proyecto y la mayor parte del tiempo se ha centrado en el desarrollo Back-End.

En resumen, el funcionamiento general del sistema ofrece el análisis de sentimiento de mensajes que llegan según unas palabras de filtrado.

5.1.1 Back-End

La idea general de este entorno es la de proveer de un sistema cluster para el análisis de mensajes. Este sistema cluster está compuesto de varias fases:

5.1.1.1 Entrada de mensajes en el sistema

Esta entrada se ha implementado en este caso utilizando la API de Twitter. Según las reglas de filtrado que llegan, se abre un flujo de datos y se mantiene a la espera de la llegada de mensajes. Cuando llega un mensajes al sistema, se realiza un parseado de sus partes y se desechan las que no van se necesarias para el análisis. En concreto, de cada mensaje, el sistema guarda los valores de:

- ID: EL identificador único del mensaje.
- Autor: El nombre completo del emisor del mensaje.
- Avatar: Url de la imagen del perfil de Twitter correspondiente al emisor.
- Text: El contenido del mensaje.
- Fecha: Fecha en la que ha sido emitido el mensaje.
- Username: Nombre de usuario del perfil emisor en Twitter.

Después de este tratamiento previo, el siguiente paso es la emisión del objeto, que contiene el mensaje, a la siguiente fase del procesamiento: El Cluster.

5.1.1.2 Procesamiento del mensaje.

Esta fase es uno de los aspectos claves del desarrollo de este proyecto. Se ha creado un entorno Cluster en el que se aprovechan la mayor parte de los recursos de que dispone la máquina en la que se ejecuta. Esto tiene sentido porque el proceso de asignación de los mensajes se realiza de forma óptima según una planificación round-robin. De esta forma se realiza un procesamiento paralelo de los mensajes que van llegando.

Este entorno está compuesto de dos partes: un proceso padre y un grupo de procesos hijos que son desplegados y mantenidos por el padre. La misión del proceso padre es:

- Desplegar un numero determinado de procesos hijos.
- Mantener activos estos procesos. Si uno termina inesperadamente, se crea uno nuevo para sustituirle.
- Recibir información del proceso que monitoriza Twitter.
- Distribuir esa información entre los hijos para que la procesen.

Por otra parte, los procesos hijos son los que realizan la mayor parte del trabajo en esta fase. Estos realizan las siguientes acciones de forma paralela e independientes unos de otros:

- Recibir los mensajes que envía el proceso padre.
- Enviar los mensajes a los diferentes servicios de análisis del sentimiento y mantenerse a la espera de una respuesta.
- Calcular el grado de sentimiento según el resultado obtenido de la consulta.
- Enviar los mensajes ya analizados al usuario que ha realizado la consulta.

5.1.1.3 Análisis de sentimiento

Podría decirse que esta es una de las fases intermedias en el procesamiento que se describe en el punto anterior, sin embargo, esta parte se lleva a cabo en un proceso independiente de Nodejs. Su implementación ha supuesto un gran reto y tiene ciertas características que la hacen especial. Las fases del funciones de esta parte del procesamiento son las siguientes:

- Crear una instancia Java del entorno Core-NPL inicializándola con las opciones escogidas.
- Poner en funcionamiento un servidor http para la llegada de los mensajes con peticiones de los procesos hijos descritos anteriormente.
- Procesar los mensajes mediante los métodos que se han especificado.
- Responder a las peticiones con las puntuaciones de polaridad.

Una de las características clave de esta fase es el hecho de que puede ser realizada en cualquier entorno, plataforma o servidor. En este caso se ha decidido utilizar un adaptador de Java a Nodejs para que todo funcione de la forma más nativa posible. Esto se debe a que la mayoría de implementaciones de algoritmos de Sentiment análisis se encuentran en entonos muy diferentes a Nodejs. Esto no supone un problema ya que el sistema está diseñado para realizar peticiones http a servidores que ofrecen las funcionalidades que no sería óptimo implementar de forma totalmente nativa.

5.1.1.4 Servicio Web

En esta fase del procesamiento se encuentra todo lo relacionado con las entradas iniciales y las salidas finales de la implementación. Este servicio es, al igual que los demás, independiente. Esto quiere decir que el sistema podría servir para realizar análisis de sentimiento sin necesidad de comunicarse con un usuario, es decir, integrado dentro del funcionamiento de otro sistema.

En este entorno en concreto, el sistema realiza las siguientes acciones:

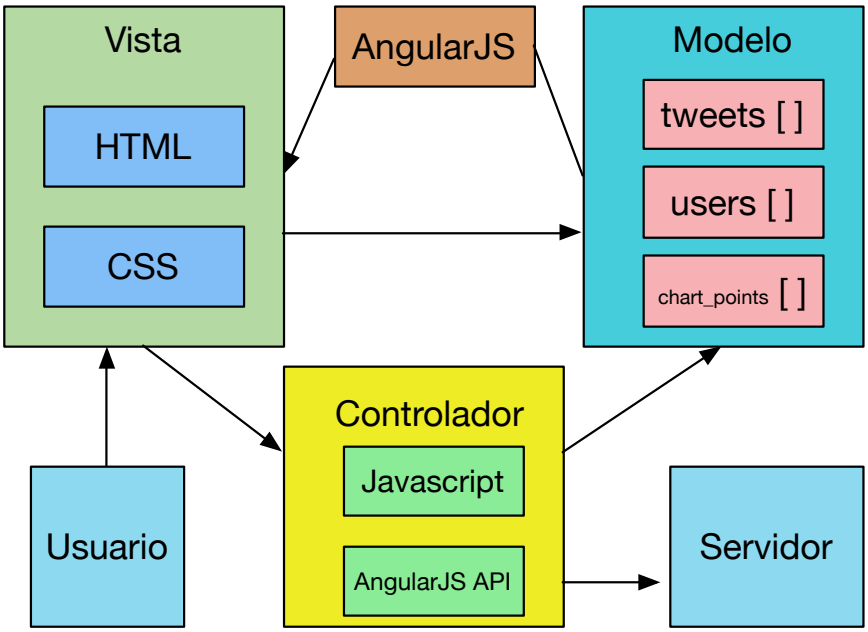
- Desplegar un servidor web con la página de gestión del sistema.
- Mantenerse a la espera de palaras de filtrado introducidas por el usuario.
- Enviar las palabras de filtrado al servidor que gestiona la conexión con la API de Streaming de Twitter.
- Recibir los mensajes analizados y enviarlos a la interfaz de usuario.

La interfaz que se ha desarrollado es una implementación simple y sencilla que permite comprobar el correcto funcionamiento del sistema, pero sobre todo, permite poner el sistema en funcionamiento con unas palabras de filtrado.

5.1.2 Front-End

Aunque esta parte del proyecto sea la que menos dificultad ha supuesto implementar, es necesario que esté presente para poder interactuar con él y mostrar sus salidas.

Se ha utilizado el modelo-vista-controlador para desplegar un entorno en el que se muestra continuamente lo que está sucediendo de la forma más inmediata y eficiente posible. A continuación se muestra un esquema del funcionamiento del patrón Modelo-Vista-Controlador utilizado en la implementación de la interfaz de usuario.



Modelo-Vista-Controlador en la web [20]

Esta parte de la implementación se ha diseñado para funcionar de la forma más rápida posible. Sus principales funciones son:

- Enviar las palabras de filtrado al servidor web.
- Recibir los mensajes ya analizados.
- Mostrar y actualizar una gráfica con la progresión de los valores de sentimiento de los mensajes.
- Mostrar y actualizar un listado con los mensajes recibidos y sus puntuaciones de sentimiento.

5.2 Estructura y funcionamiento

Esta implementación se divide en cuatro partes entre las que se reparte el funcionamiento del sistema. Estos cuatro módulos se encuentran separados y se ejecutan de forma independiente unos de otros, aunque exista comunicación entre constante entre ellos, esta tiene lugar una vez se han desplegado todos.

5.3 Funcionamiento

A continuación se exponen los diagramas de secuencia que corresponden a los procesos que tienen lugar en el sistema cuando éste está funcionando. Se ha utilizado nombres esquemáticos para los componentes.

USER: Interfaz web con la que el usuario interactúa.

WEB: Servidor Web.

TWITTER: Servidor que atiende a la API de Stream de Twitter

SENT: Servidor de análisis de sentimiento. Web Service.

CLUSTER [Master]: CLUSTEReso padre en el servidor cluster.

CLUSTER [Worker]: CLUSTEReso hijo en el servidor cluster.

USER - WEB - TWITTER - CLUSTER

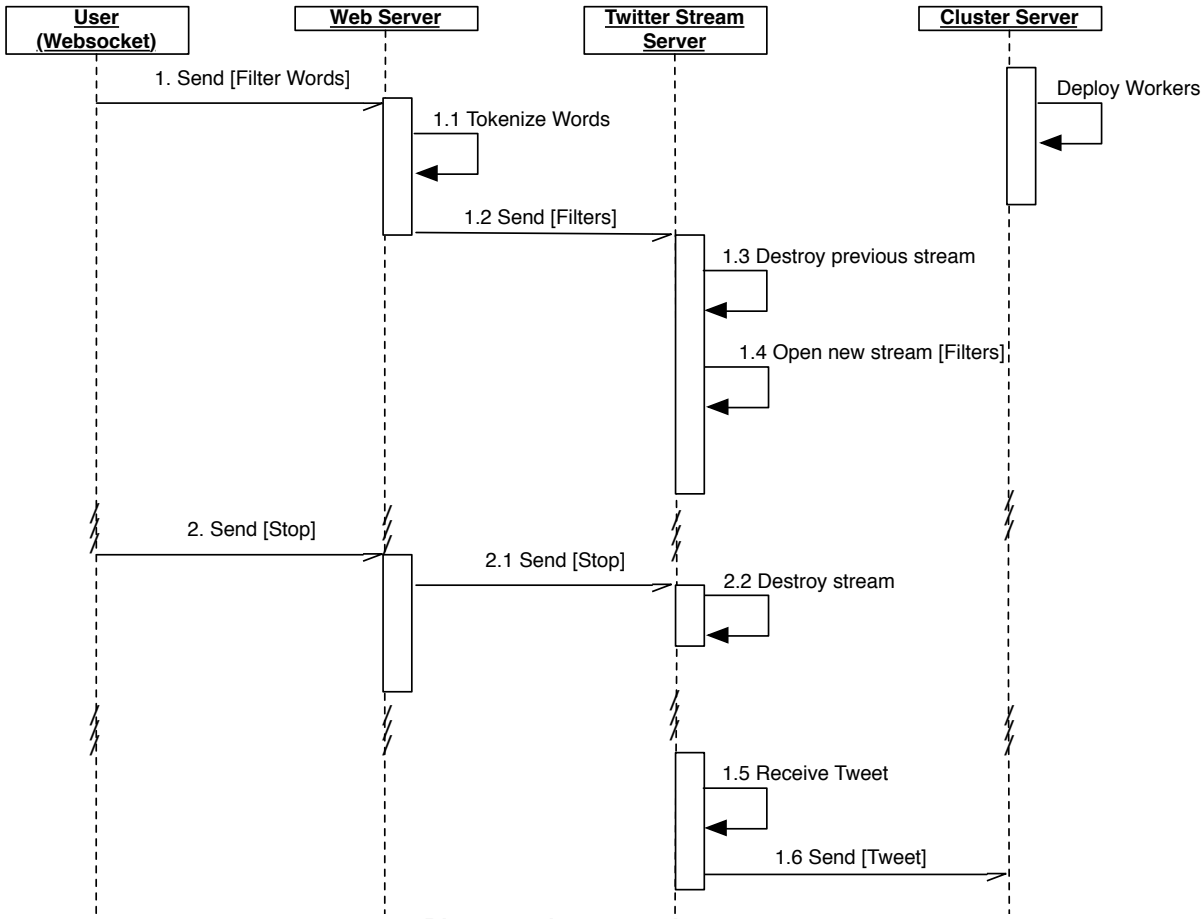


Diagrama de secuencia 1

1- Envío mediante WebSocket de las palabras de filtrado al servidor web.
1.1- Chequeo y normalización los filtros.
1.2- Envío de los filtros al servidor de flujo de Twitter.
1.3- Eliminación del anterior flujo de mensajes.
1.4- Creación de un nuevo flujo de Twitter.
1.5- Recibir mensaje desde el flujo de Twitter.
1.6- Envío del mensaje al servidor Cluster.

2- Envío de la señal de parada del flujo al servidor Web.
2.1- Envío del mensaje de parada al servidor de flujo.
2.2- Destrucción del flujo actual.

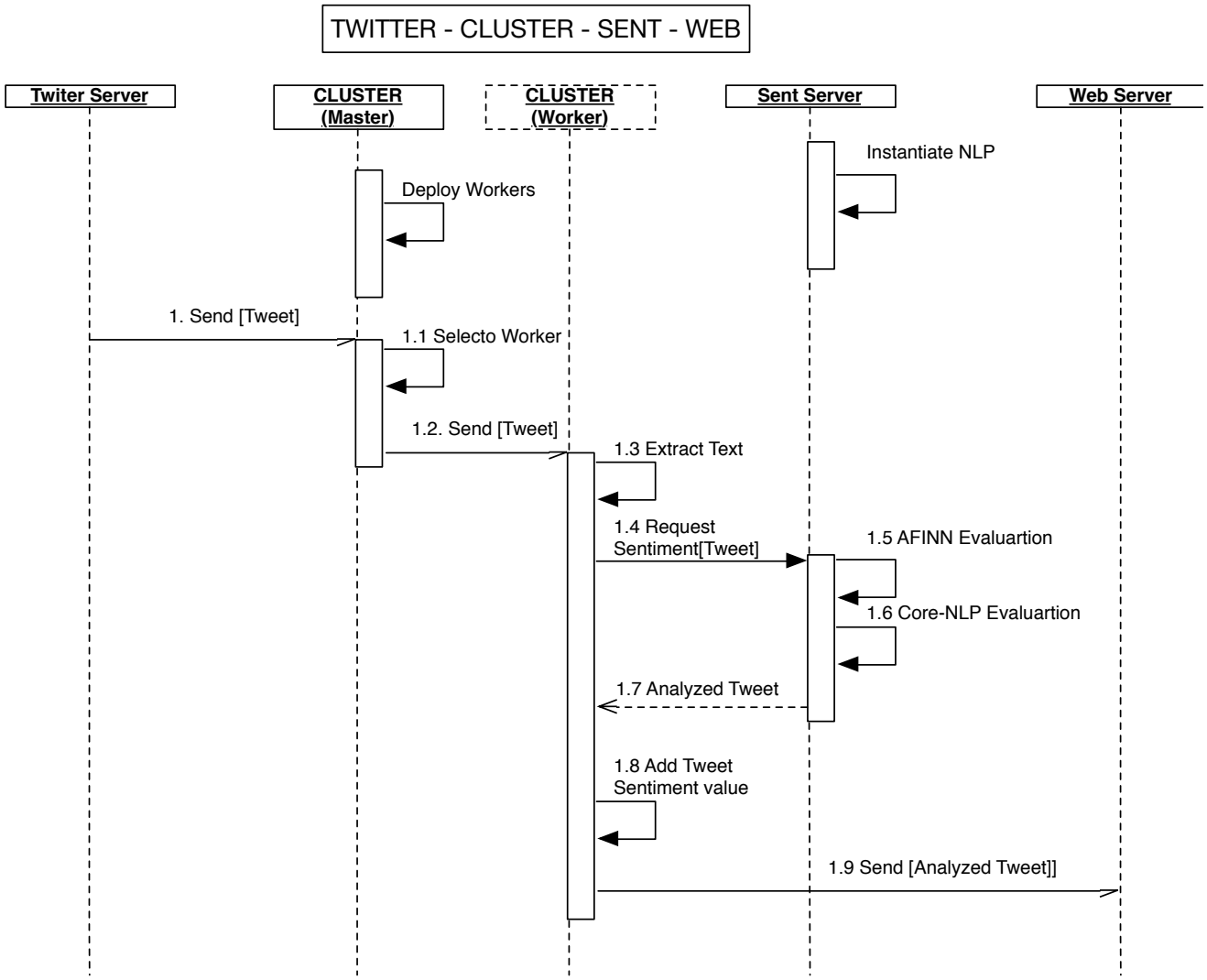


Diagrama de secuencia 2

1- Envío del mensaje al servidor Cluster.
1.1- Selección del proceso Worker destino.
1.2- Envío de mensaje al proceso Worker.
1.3- Extracción del texto del mensaje.
1.4- Petición de procesamiento del mensaje al servidor de análisis de sentimiento.
1.5- Evaluación mediante el método AFINN.
1.6- Evaluación mediante el método Core-NLP.
1.7- Envío del mensaje ya analizado.
1.8- Agregación de los datos de sentimiento al cuerpo del mensaje.
1.9- Envío del mensaje analizado al servidor web.

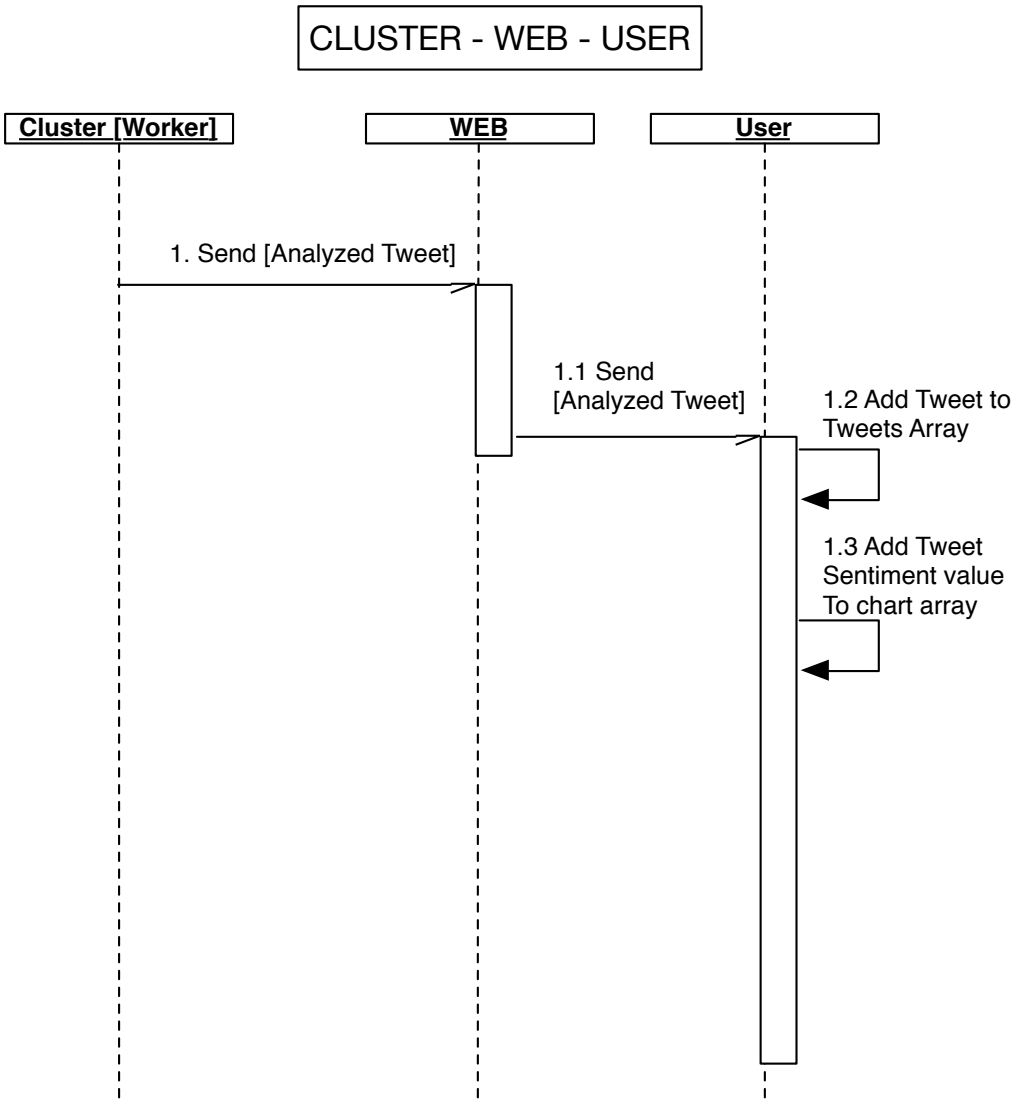


Diagrama de secuencia 3

1- Envío del mensaje analizado al servidor web.
1.1- Envío del mensaje analizado a través del WebSocket del usuario.
1.2- Inclusión del mensaje analizado en la lista de mensajes.
1.3- Inclusión de la puntuación del mensaje en la lista de puntos de la gráfica.

5.4 Estructura

Como se ha descrito anteriormente. Este proyecto se encuentra dividido en cuatro partes o fases de funcionamiento. Estas se encuentran representadas de forma esquemática en la imagen que se muestra a continuación.

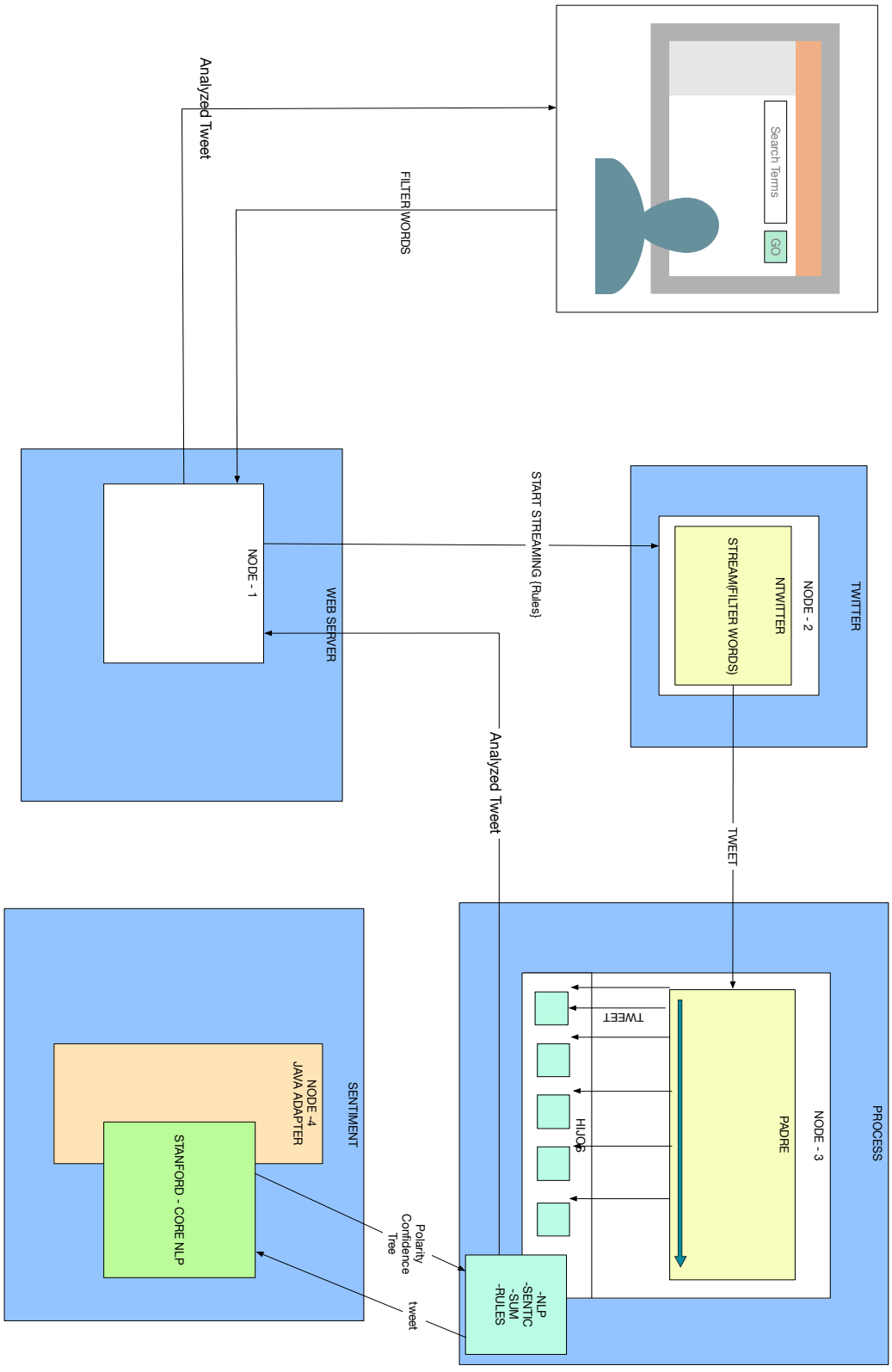


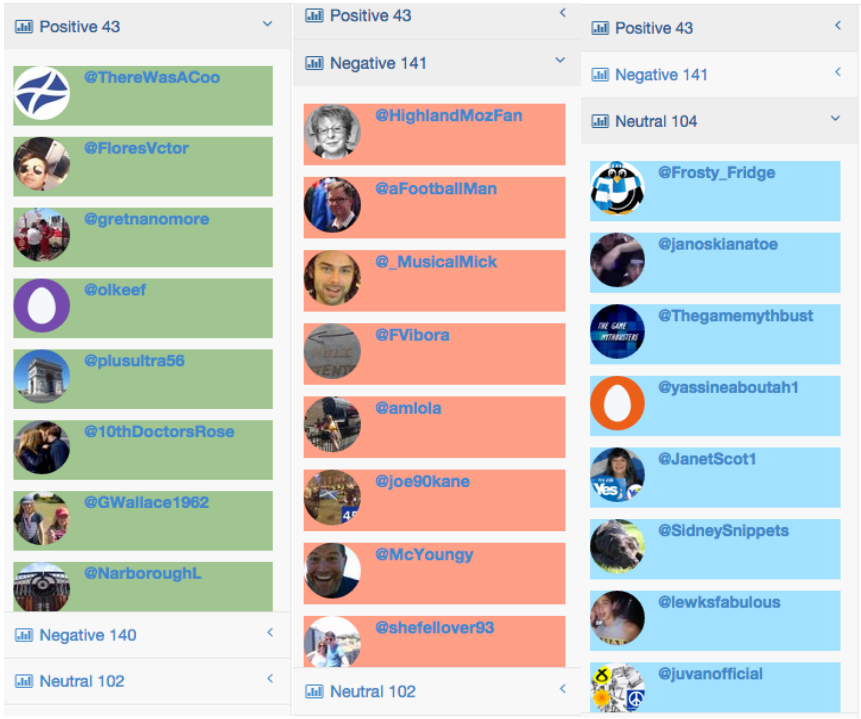
Diagrama de distribución de la implementación

5.5 Métricas

La principal y más asequible de las formas de hacer uso de una herramienta como la que se ha descrito aquí es la utilización de una interfaz de usuario fácil de entender y usar. Sin embargo, esto a veces no basta y es necesario que esta contenga ciertos valores o muestre algunos detalles que puedan dar una impresión, de la forma más rápida posible, acerca del estado de lo que el usuario está consultando. Estos datos reciben el nombre de métricas.

A través de la interfaz que se ha desarrollado se pretende ofrecer varias métricas nuevas relacionadas con el análisis de sentimiento que se ha hecho de los mensajes:

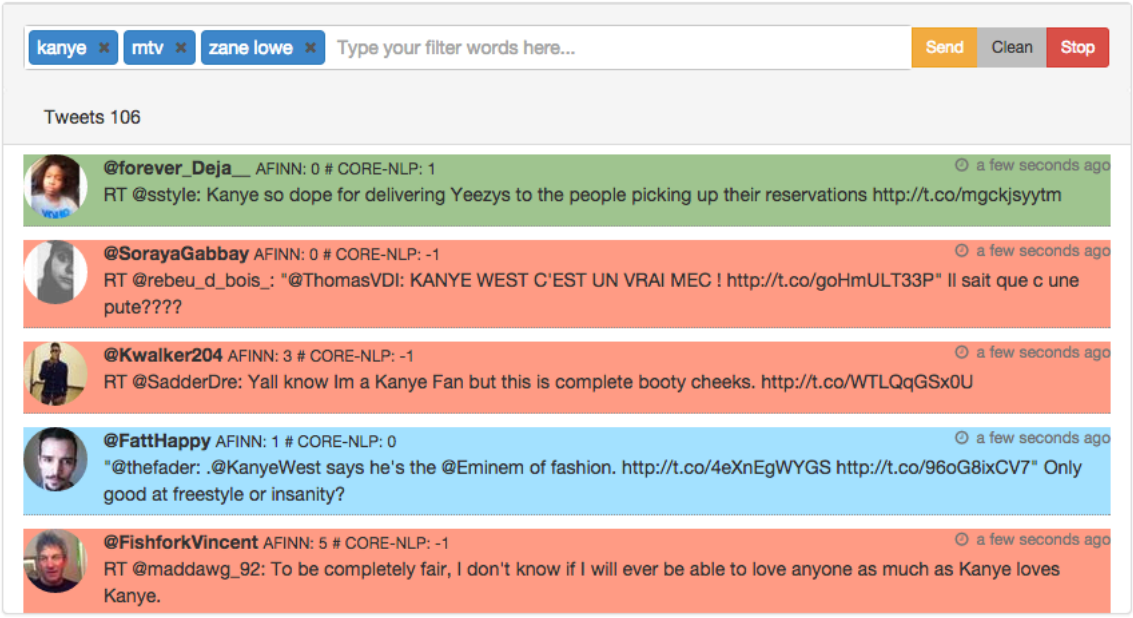
1. Listado de usuarios contentos, descontentos y neutrales respecto a unos temas:



El principal valor de este tipo de listados es que permite conocer rápidamente y a primera vista, quiénes son los usuarios de Twitter que se han manifestado en desacuerdo o de acuerdo con los temas analizados. Además, al mantener el cursor encima de la entrada de un usuario en concreto, es posible leer cual ha sido su publicación, de cara a poder responderle o realizar un seguimiento más activo del mismo en la red. Esto es de especial utilidad por ejemplo, en agencias de publicidad o comunicación que tienen interés por conocer el grado de acogida de un producto o una acción televisiva de forma rápida.

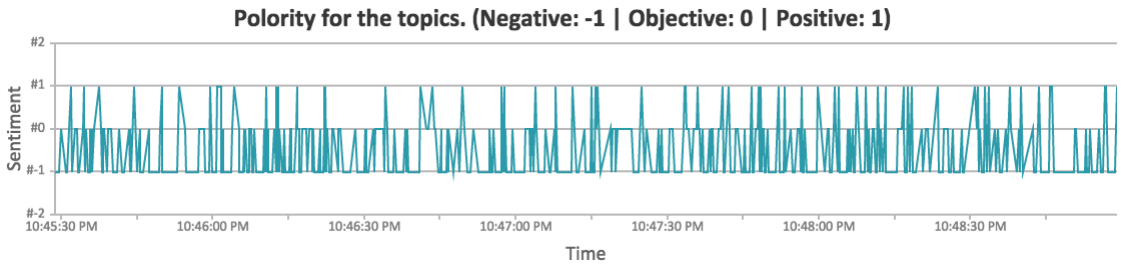
2. Listado con todas las publicaciones sobre un tema.

Gracias a un listado con todas las publicaciones ordenadas de forma cronológica sobre uno o varios temas, es posible comprobar el progreso que ha tenido, o está teniendo, la audiencia y los usuarios. Esto es de gran utilidad por ejemplo, en temas como la medición de la audiencia en televisión.



3. Gráfica temporal

Finalmente, se presenta esta gráfica temporal que muestra de una forma visual el progreso que ha seguido el sentimiento de las publicaciones de los usuarios en Twitter.



Gracias a este tipo de visualización es posible apreciar los momentos en los que se concentran grandes cantidades de mensajes con una polaridad concreta.

5.6 Conclusión del desarrollo y posibles mejoras

Inicialmente, el propósito de este proyecto era realizar una implementación de los algoritmos expuestos anteriormente utilizando Storm. Sin embargo, a medida que avanzaba el desarrollo se llegó a la conclusión de que resulta un entorno complicado de implementar y los resultados no son suficientes para demostrar su valor. Posteriormente se analizaron otras tecnologías, como Apache Spark, pero debido a la falta de documentación y la inmadurez del desarrollo, se descartó.

Como se puede comprobar, ha sido necesario realizar una investigación previa que ha supuesto el 60% del tiempo de este proyecto. Durante este tiempo han ido apareciendo diferentes opciones, que finalmente se han decidido implementar.

Sobre los algoritmos implementados, se ha decidido utilizar dos de todos los estudiados ya que incluyen los dos métodos más comunes en el entorno del análisis de sentimiento. Estos son: Core-NLP de Stanford y AFINN. Como se puede comprobar en la implementación, no suponen un avance cuantitativo en cuanto al resultado de las clasificaciones, ya que en muchos casos difieren en sus tomas de decisión, pero en más casos todavía los dos fallan. Sí que se considera un avance en este aspecto a la aplicación de ambos en tiempo de ejecución mientras se mantiene la entrada del flujo de mensajes de Twitter.

Respecto al rendimiento alcanzado, es importante destacar que está muy limitado por la máquina en la que se ha llevado a cabo el desarrollo de este proyecto. Se ha tenido en cuenta las posibles aplicaciones a gran escala y por lo tanto es posible desplegar el entorno en un conjunto de máquinas distribuidas sin que afecte esta separación a la comunicación.

En cuanto a las posibles mejoras que se podrían llevar a cabo en el futuro, se ha tenido en cuenta las limitaciones que conlleva un desarrollo como este en el periodo de tiempo en el que se ha hecho. Los principales aspectos sobre los que se podría mejorar son los siguientes:

- Realizar el análisis de sentimiento en máquinas separadas y de forma paralela.
- Ofrecer una API más completa para poder desarrollar servicios externos.
- Realizar una comparación de los resultados de la aplicación de todos los métodos vistos.
- Implementar el sistema en Apache Spark.
- Implementar un sistema de almacenamiento para poder guardar informes.
- Desarrollar una aplicación Front-End más completa y orientada a usuarios y sesiones.
- Implementar gráficas más descriptivas.

Referencias

- Manning, Christopher D., Surdeanu, Mihai, Bauer, John, Finkel, Jenny, Bethard, Steven J., and McClosky, David. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations.

- Pollyanna Gonçalves , Matheus Araújo , Fabrício Benevenuto , Meeyoung Cha, Comparing and combining sentiment analysis methods, Proceedings of the first ACM conference on Online social networks, October 07-08, 2013, Boston, Massachusetts, USA

- Gabriel Pui Cheong Fung , Jeffrey Xu Yu , Philip S. Yu , Hongjun Lu, Parameter free bursty events detection in text streams, Proceedings of the 31st international conference on Very large data bases, August 30-September 02, 2005, Trondheim, Norway.

- Soujanya Poria, Erik Cambria, Grégoire Winterstein, Guang-Bin Huang, Sentic patterns: Dependency-based rules for concept-level sentiment analysis, Knowledge-Based Systems, Volume 69, October 2014, ISSN 0950-7051.

- CAMBRIA, E.; OLSHER, D.; RAJAGOPAL, D.. SenticNet 3: A Common and Common-Sense Knowledge Base for Cognition-Driven Sentiment Analysis. AAAI Conference on Artificial Intelligence, North America, jun. 2014.

- P. Taylor Goetz, Brian O'Neill. Storm Blueprints: Patterns for Distributed Real-time Computation. ISBN 139781782168294.

- Jonathan Leibiusky, Gabriel Eisbruch, Dario Simonassi. Getting Started with Storm. Continuous streaming computation with Twitter's cluster technology. ISBN 10:1-4493-2401-0.

- Ankit Jain, Anand Nalya. Learning Storm. Create real-time stream processing applications with Apache Storm. ISBN 139781783981328

- Finn Årup Nielsen. A new ANEW: Evaluation of a word list for sentiment analysis in microblogs

Enlaces

[1]
- https://semantria.com/support/resources/technology#sentiment-analysis
- http://homepages.dcc.ufmg.br/~fabricio/download/de03-araujo.pdf
- http://www.sentiment140.com/
-https://gnip.com/pages/ihs-next-generation-sentiment-intelligence-in-action-whitepaper/

[2]
-https://success.crowdflower.com/hc/en-us/articles/203510355-Sentiment-Analysis-Case-Study-Unilever

[3]
- http://homepages.dcc.ufmg.br/~fabricio/download/cosn127-goncalves.pdf

[4]
- http://www.hope-you-are-well.com/docs/assets/img/examples/tom-liwc.jpg
- http://www.liwc.net/

[5]
- http://nlp.stanford.edu/sentiment/code.html
- http://nlp.stanford.edu/~socherr/EMNLP2013_RNTN.pdf

[6]
- http://sentistrength.wlv.ac.uk/
- http://www.slideshare.net/humanzz/entity-aspect-analysis

[7]
- http://ontotext.fbk.eu/sentiwn.html
- http://sentiwordnet.isti.cnr.it/

[8]
- http://sentic.net/about/
- http://www.aaai.org/ocs/index.php/AAAI/AAAI14/paper/viewFile/8479/8602

[9]
- https://storm.apache.org/
- http://hortonworks.com/hadoop/storm/

[10]
- https://spark.apache.org/docs/latest/
-http://www.zdatainc.com/2014/08/real-time-streaming-apache-spark-streaming/

[11]
- https://www.xplenty.com/blog/2014/11/apache-spark-vs-hadoop-mapreduce/
- http://www.cs.duke.edu/~kmoses/cps516/dstream.html

[12]
- http://nodejs.org/
- http://nodejs.org/about/

[13]
- https://dev.twitter.com/rest/public

[14]
- https://dev.twitter.com/streaming/overview

[15]
- https://gnip.com/products/realtime/firehose/

[16]
- http://socket.io/

[17]
- http://nodejs.org/api/cluster.html

[18]
- https://www.npmjs.com/package/node-twitter-api

[19]
- http://expressjs.com/

[20]
- https://angularjs.org/

[21]
- https://twitter.com/cocacola/status/511424163492659200

[22]
- https://twitter.com/24symbols_es/status/564389314331951104

[23]
- https://twitter.com/amyscioscia

[24]
- https://twitter.com/Sarahngb/status/354327094300672000

NUEVAS HERRAMIENTAS PARA EL ANÁLISIS DE OPINIÓN EN FLUJOS DE TEXTO

Vicente Hidalgo Santamaría

INDICE

1- Introducción

1- Introducción

- La minería de opinión se ha convertido en uno de los objetivos de las grandes plataformas de Internet, analistas y entes gubernamentales que desean conocer las opiniones de los usuarios que hacen uso de ellas.
- En este proyecto se ha implementado un sistema que usa unos algoritmos de análisis de sentimiento para analizar flujos de texto de redes sociales.

2- Las Redes Sociales

2- Redes sociales

- Durante los últimos años han cobrado un protagonismo trascendental en el entorno de la comunicación.
- Twitter
  - La segunda red por contenido/usuarios.
  - Se ha convertido en la plataforma de comunicación más rápida y ubicua.
  - Usado por 248 Millones de usuarios. 8,118 Tweets/s.

2- Redes sociales - Sociedad

- Facebook tiene 1 de cada 7 minutos de la audiencia online.
- Twitter es la tercera red más utilizada después de Facebook y YouTube.
- Muchos usuarios acuden antes a Twitter que a los medios o páginas web para conocer la actualidad.
- Ha quedado más que demostrada su eficacia en levantamientos sociales.

Redes sociales - Marketing

- Mayor reconocimiento de la marca.
- Mayor lealtad de los consumidores.
- Alta tasa de conversión.
- Incremento del tráfico.
- Menores costes de marketing.
- Mejor experiencia de usuario.

Redes sociales - Marketing

3- Análisis de Sentimiento

3- Análisis de Sentimiento

- ¿Qué es?

  Es el proceso de determinar la polaridad contextual de un contenido. En el caso de un texto, determinar si es positivo o negativo.

3- Análisis de Sentimiento

- Diferentes métodos
  - Búsqueda de palabras clave
  - Afinidad léxica
  - Técnicas basadas en conceptos
  - Métodos basados en Redes Neuronales

3- Análisis de Sentimiento

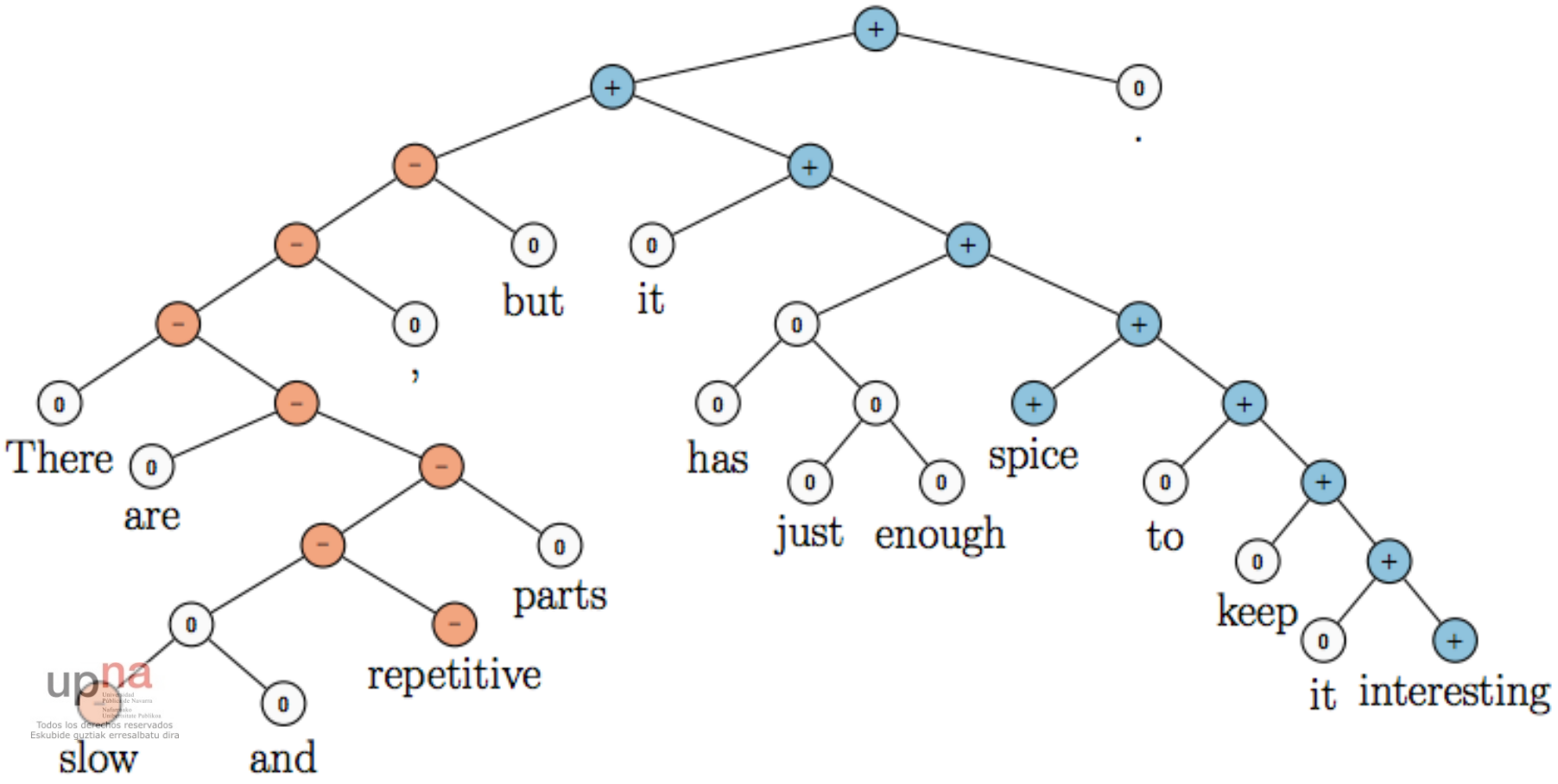
- Afinidad Léxica (Lexicon enhaced methods)
  - AFINN, LIWC
  - Colección de palabras catalogadas según su polaridad.
  - No solo detecta palabras con una polaridad de sentimiento evidente, sino que además atribuye a palabras arbitrarias valores de polaridad.

3- Análisis de Sentimiento

- Métodos basados en Redes Neuronales
  - Stanford Core-NLP
  - Basado en el uso de Sentiment Treebanks
    - 215,154 frases, etiquetadas según su clasificación, de un total de 11,855 documentos (Reviews de películas).
  - Detecta las negaciones.
  - Red neuronal recursiva.
  - Implementación en Java.
  - Incluye herramientas para parseado y extracción del sujeto o conceptos.

3- Análisis de Sentimiento

- Stanford Core-NLP



4- Sistemas de análisis de Streaming

4- Sistemas de análisis de Streaming

- Sistemas evaluados previamente
  - Apache Storm
  - Storm Trident
  - Apache Kafka
  - Apache Spark

4- Sistemas de análisis de Streaming

- Apache Storm
  - Procesamiento en tiempo real.
    - Asegura la entrega de resultados en un plazo de tiempo configurable.
  - Tolerante a Fallos en todas las fases del procesamiento.
  - Configuración en topologías.
  - Despliegue en docker.
  - Desarrollo muy lento y despliegue difícil.

4- Sistemas de análisis de Streaming

- Trident
  - Capa de abstracción para Apache Storm.
  - Tolerante a Fallos en todas las fases del procesamiento.
  - Sistema más sencillo. Auto-distribución.
- Apache Kafka
  - Cluster de colas de datos.

4- Sistemas de análisis de Streaming

- Apache Spark
  - Map-Reduce en memoria.
  - Micro Batching.
  - Sucesor de Hadoop.
    - 10 veces más rápido.
  - Integración con otros entornos.
  - Java, Scala y Python.
  - Permite el despliegue en Hadoop.
  - Documentación escasa hasta ahora.

5- Nodejs

5- Nodejs

- Plataforma construida para ejecutar aplicaciones JavaScript en entorno de servidor sobre el motor V8 de Google Chrome.
  - Funciona a través de una cola de procesamiento.
  - Facilidad para construir entornos escalables.
  - Orientado a eventos de entrada y salida.
  - Existen muchos módulos que aportan funcionalidades muy interesantes:
    - Socket.io
    - Express
    - etc

5- Nodejs

- Cluster
  - Provee de la funcionalidad multiproceso.
  - Facilidad en la comunicación entre procesos padres e hijos.
  - El padre es capaz de levantar a los hijos si estos mueren.
  - Aplicable en muchas funciones del servidor.

5- Nodejs

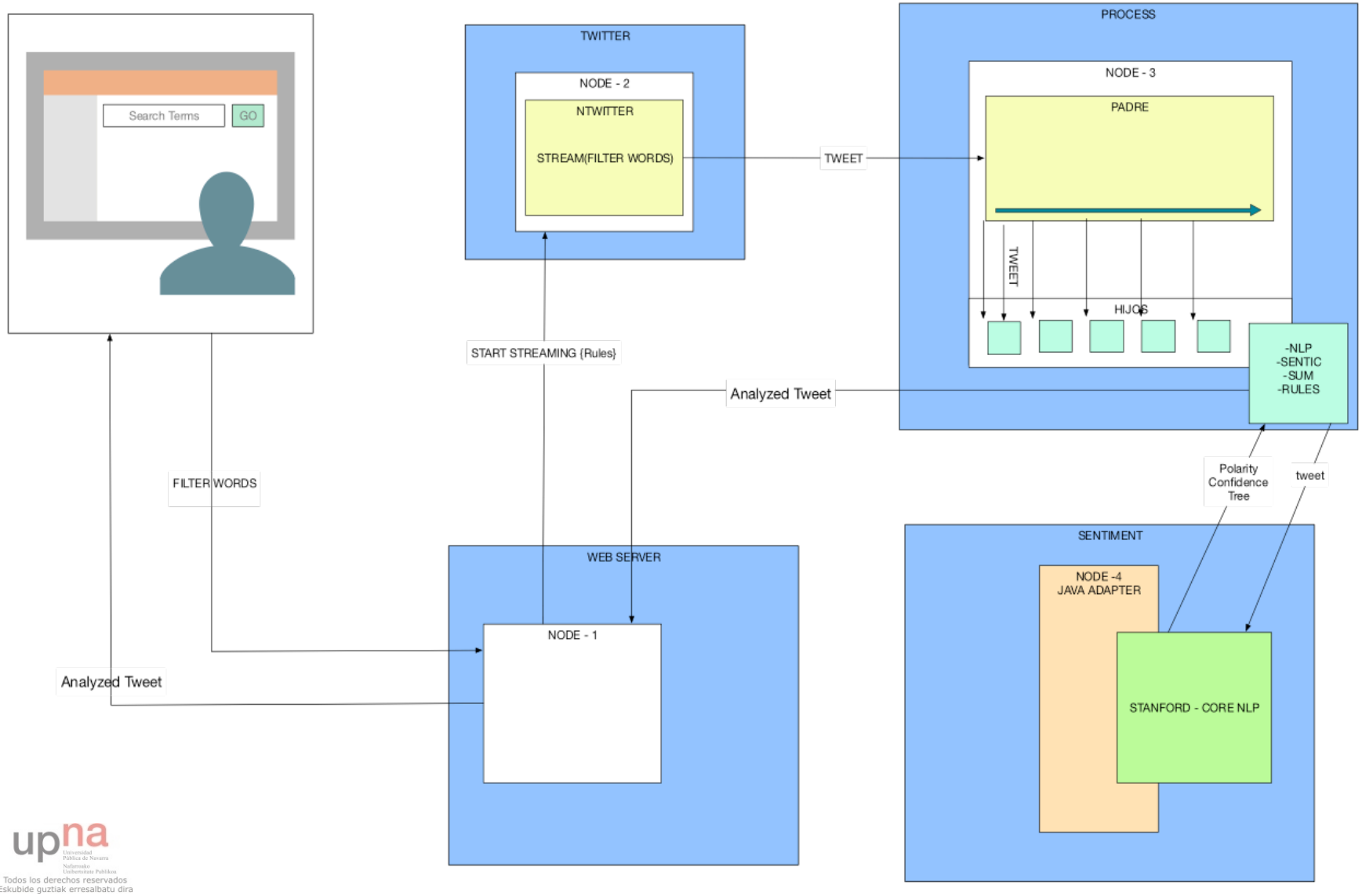
- Socket.io
  - Es un módulo (librería) de Nodejs.
  - Módulo para el paso de mensajes mediante sockets a través de conexiones http.
  - Se puede utilizar tanto en el lado de cliente como en el servidor.
  - Recupera automáticamente la conexión en el caso de que haya una caída en uno de los módulos.

6- Desarrollo e implementación

6- Implementación

- Sistema de análisis de sentimiento del Stream de mensajes de Twitter.
- Obtiene los mensajes a través de la API de Streaming.
- Realiza procesamiento paralelo.
- Se ha utilizado un adaptador de Java a Nodejs para las partes con más coste computacional.

6- Implementación



7- DEMO