

SENTBALL

Análisis de sentimientos utilizando Node.js y Twitter

Marcos J. Peña Pollastri Daniel Fuentes

24 junio del 2019

Resumen

En el presente trabajo se investiga el desarrollo de una webapp que permita la captura de datos no estructurados en Twitter y su correspondiente análisis de sentimientos. Los alumnos aplicaron dicho software en una prueba de análisis de la serie Game Of Thrones.

SENTBALL

UNIVERSIDAD NACIONAL DE LA RIOJA

SEDE CAPITAL

LIC. EN SISTEMAS DE INFORMACIÓN
CÁTEDRA DE SEMINARIO DE ACTUALIZACIÓN II

MARCOS J. PEÑA POLLASTRI
DANIEL FUENTES

2019

Índice

I	Introducción	5
1.	Sobre este trabajo	6
1.1.	Motivación	6
1.2.	Objetivos	7
1.3.	Alcances	7
1.4.	Calendario de trabajo	7
II	Marco Teórico	8
2.	Procesamiento del lenguaje natural (PLN o NLP)	9
2.1.	Ventaja	9
2.2.	Desventaja	10
2.3.	Aplicaciones del Procesamiento de Lenguaje Natural	10
2.4.	Arquitectura de un sistema PLN	10
2.5.	Recursos abiertos para aplicar el Procesamiento de Lenguaje Natural	11
2.6.	Problema del procesamiento de lenguaje natural	12
2.7.	Historia de PLN	12
2.7.1.	La lingüística teórica y el PLN en los años 40 y 50	12
2.7.2.	Los años 60	12
2.8.	Ambigüedad	14
2.9.	Adquisición de conocimiento léxico	15
2.10.	La comprensión del lenguaje en Inteligencia Artificial	17
2.11.	Las expresiones regulares en PLN y la búsqueda de patrones	18
3.	Análisis de sentimientos	20
3.1.	Tipos de análisis de sentimiento	21
3.1.1.	Análisis a nivel de documento	21
4.	Aprendizaje Computacional	23
4.1.	Tipos de Aprendizaje	23
4.1.1.	Aprendizaje supervisado	23
4.1.2.	Aprendizaje no supervisado	24
4.1.3.	Aprendizaje semi-supervisado	24
5.	La Red Social Twitter	26
III	Software utilizado en este proyecto	27
6.	Node.Js	28
6.1.	Descripción general	28
6.1.1.	Arquitectura de la plataforma	29
6.1.2.	Soporte a la industria	29
6.2.	Historia	30
6.3.	Versiones	31
6.4.	Aspectos técnicos	32

6.4.1.	Hilos	32
6.4.2.	V8	32
6.4.3.	Gestor de paquetes NPM	32
6.4.4.	Ciclo de eventos	32
7.	Express	32
7.1.	¿Qué popularidad tiene Node/Express?	33
7.2.	¿Es Express dogmático?	33
7.3.	¿Cómo es el código para Express?	34
8.	MongoDB	35
8.1.	Historia de MongoDB	35
8.2.	Características	35
8.2.1.	Consultas Ad hoc	35
8.2.2.	Indexación	35
8.2.3.	Replicación	35
8.2.4.	Equilibrio de carga	36
8.2.5.	Agregación	36
8.2.6.	Ejecución de JavaScript	36
8.2.7.	Colecciones limitadas	36
8.3.	¿Por qué MongoDB?	36
8.4.	Esquema de datos de este proyecto	37
9.	API de Twitter	38
9.1.	Objeto Tweet	39
9.1.1.	Diccionario de datos del objeto Tweet	39
10.	Sentiment	40
10.1.	Funcionamiento	40
IV	Prueba de Sentball	41
11.	#GameOfThrones	42
11.1.	Resultados	44
V	Conclusión	46
12.	Conclusión	47
13.	Evolución a futuro	48
VI	Bibliografía	49
VII	ANEXO	50

Parte I. Introducción

1. Sobre este trabajo

El presente trabajo es realizado por pedido expreso de la cátedra de Seminario de Actualización II, correspondiente al quinto año de la carrera de Lic. en Sistemas de Información de la Universidad Nacional de La Rioja.

Siendo la consigna el realizar un desarrollo e investigación relacionada al software en una temática que sea de interés en la actualidad de la disciplina de los Sistemas de Información. Correspondiendo a eso, el Análisis de los Sentimientos es la temática seleccionada.

En esta sección introductoria se expondrá:

- Motivación que dio pie inicial a este trabajo.
- Objetivos principales del mismo.
- Campo de aplicación de la temática seleccionada.
- Alcances.
- Calendario de trabajo.

1.1. Motivación

Con el auge de las redes sociales y el internet, las formas de recolectar información por medios tradicionales (encuestas telefónicas, rating de tv, cartas, etc) comenzaron a mostrar signos de obsolescencia, debido a las propias limitaciones con las que fueron concebidas. Por ejemplo, podemos saber con el rating televisivo que un programa no es atractivo para los televidentes, sin embargo carecemos de información sobre el porqué. Esto nos lleva a que se debe encarar entonces algún otro método de medición, como una encuesta telefónica. Sin embargo, en la misma se cuentan factores problemáticos como:

- El engorroso proceso de seleccionar una muestra útil sin “ruido”, por ejemplo, si decidimos llamar a 1000 personas en la Ciudad de Buenos Aires, tendremos seguramente un considerable porcentaje de personas que no estén interesadas en responder, o que para evitar hacer una encuesta demasiado extensa mientan o no den razones extendidas de porqué.
- Mientras se ha encargado este estudio de marketing para delimitar las razones, nuestro programa de tv sigue en el aire, y sigue con problemas. El tiempo en el que recibimos la información es un factor primordial, por lo que quizás tardamos semanas en recibir los resultados del estudio, pero para entonces ya será demasiado tarde y nuestro programa será cancelado.

Esto nos hace preguntar, y si existiera una forma de que las personas mismas nos digan extensamente qué es lo que les molesta de nuestro producto, sin intermediarios, sin complicaciones, solo como si fuera una charla amistosa. Esta es justamente la facilidad que nos proporciona las redes sociales: datos no estructurados pero igualmente valiosos.

1.2. Objetivos

Los objetivos planteados son los siguientes:

1. Desarrollar un software que permita la captura de datos no estructurados que se encuentren en redes sociales.
2. Que dicho software sea capaz de analizar los sentimientos de los datos capturados por medio de una formula matemática.
3. Persistencia de los datos capturados y sus correspondientes resultados.

1.3. Alcances

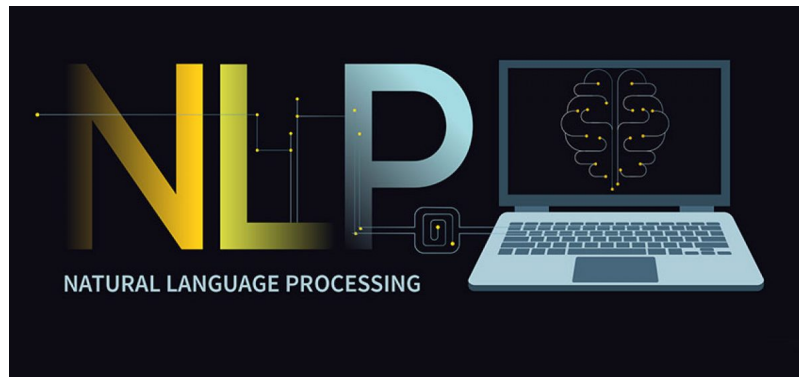
1. Desarrollo de una web app que permita el procesamiento en backend de sentimientos con respecto a un tema en la red social Twitter.
2. La creación de un banco de datos de lo procesado.

1.4. Calendario de trabajo

Nº	Actividad	Fecha de inicio	Fecha de fin
1	Elección de tema	18/03/2019	25/03/2019
2	Investigación de antecedentes	26/03/2019	02/04/2019
3	Investigación de herramientas y componentes de software a utilizar	03/04/2019	09/04/2019
4	Programación del proyecto	10/04/2019	19/04/2019
5	Elaboración de presentación de diapositivas	20/04/2019	21/04/2019
6	Prueba con el hashtag “#gameofthrones”	21/04/2019	21/04/2019
7	Presentación en la clase de Seminario de actualización II	22/04/2019	22/04/2019
8	Elaboración de video de presentación	20/05/2019	24/05/2019
9	Redacción final de este informe	25/05/2019	16/06/2019

Parte II. Marco Teórico

2. Procesamiento del lenguaje natural (PLN o NLP)



El procesamiento del lenguaje natural PLN, o NLP del idioma inglés Natural Language Processing, es un campo de las ciencias de las computación, inteligencia artificial y lingüística que estudia las interacciones entre las computadoras y el lenguaje humano. El PLN se ocupa de la formulación e investigación de mecanismos eficaces computacionalmente para la comunicación entre personas y máquinas por medio del lenguaje natural, es decir, de las lenguas del mundo. Por medio de estos algoritmos o procesos matemáticos traducen datos (lenguaje natural) en el cerebro de la máquina para que éste determine patrones y pueda generar una respuesta. Mucho dependerá del tipo, calidad y cantidad de datos de entrenamiento para determinar el éxito de respuesta de la máquina. Por ejemplo, para entrenar a Google Home, el tipo de datos que se utilizó fue la voz humana y seguramente con una extensa variedad de tonalidades y registros de voz (calidad y cantidad) provenientes de hombres, mujeres, niños, adultos, extranjeros y locales.

El PLN no trata de la comunicación por medio de lenguas naturales de una forma abstracta, sino de diseñar mecanismos para comunicarse que sean eficaces computacionalmente, que se puedan realizar por medio de programas que ejecuten o simulen la comunicación. Los modelos aplicados se enfocan no solo a la comprensión del lenguaje de por sí, sino a aspectos generales cognitivos humanos y a la organización de la memoria. El lenguaje natural sirve solo de medio para estudiar estos fenómenos. Hasta la década de 1980, la mayoría de los sistemas de PLN se basaban en un complejo conjunto de reglas diseñadas a mano. A partir de finales de 1980, sin embargo, hubo una revolución en PLN con la introducción de algoritmos de aprendizaje automático para el procesamiento del lenguaje.

2.1. Ventaja

Por un lado es una ventaja, en la medida en que el locutor no tiene que esforzarse para aprender el medio de comunicación a diferencia de otros medios de interacción como lo son los lenguajes de comando o las interfaces gráficas.

2.2. Desventaja

Su uso también presenta limitaciones porque la computadora tiene una limitada comprensión del lenguaje. Por ejemplo, el usuario no puede hablar sobrentendidos, ni introducir nuevas palabras, ni construir sentidos derivados, tareas que se realizan espontáneamente cuando se utiliza el lenguaje natural. Realmente, lo que constituye en ventaja para la comunicación humana se convierte en problema a la hora de un tratamiento computacional, ya que implican conocimiento y procesos de razonamiento que aún no sabemos ni cómo caracterizarlos ni cómo formalizarlos.

2.3. Aplicaciones del Procesamiento de Lenguaje Natural

Las aplicaciones del PLN son muy variadas, ya que su alcance es muy grande, algunas de las aplicaciones son:

- Traducción automática.
- Recuperación de la información.
- Extracción de Información y resúmenes.
- Resolución cooperativa de problemas.
- Tutores inteligentes.
- Reconocimiento de Voz.

2.4. Arquitectura de un sistema PLN

La arquitectura de un sistema de PLN se sustenta en una definición del LN por niveles: estos son : fonológico, morfológico, sintáctico, semántico, y pragmático.

- A)- Nivel Fonológico: trata de cómo las palabras se relacionan con los sonidos que representan.
- B)- Nivel Morfológico: trata de cómo las palabras se construyen a partir de unas unidades de significado más pequeñas llamadas morfemas.
- C)- Nivel Sintáctico: trata de cómo las palabras pueden unirse para formar oraciones, fijando el papel estructural que cada palabra juega en la oración y que sintagmas son parte de otros sintagmas.
- D)- Nivel Semántico: trata del significado de las palabras y de cómo los significados se unen para dar significado a una oración, también se refiere al significado independiente del contexto, es decir de la oración aislada.

- E)- Nivel Pragmático: trata de cómo las oraciones se usan en distintas situaciones y de cómo el uso afecta al significado de las oraciones. Se reconoce un subnivel recursivo: discursivo, que trata de cómo el significado de una oración se ve afectado por las oraciones inmediatamente anteriores.

La arquitectura del sistema de procesamiento del lenguaje natural muestra como la computadora interpreta y analizar las oraciones que le sean proporcionadas. La explicación de este sistema, es sencilla:

- El usuario le expresa a la computadora que es lo que desea hacer.
- La computadora analiza las oraciones proporcionadas, en el sentido morfológico y sintáctico, es decir, si las frases contienen palabras compuestas por morfemas y si la estructura de las oraciones es correcta. En esta etapa juegan un papel importante el analizador lexicográfico y el analizador sintáctico. El primero denominado scanner se encarga de identificar los componentes léxicos definidos a priori, el segundo denominado parser se encarga de verificar si se cumple un orden gramatical entre los elementos identificados por el scanner.
- El siguiente paso es analizar las oraciones semánticamente, es decir saber cual es el significado de cada oración, y asignar el significado de estas a expresiones lógicas (cierto o falso).
- Una vez realizado el paso anterior, ahora podemos hacer el análisis pragmático de la instrucción, es decir una vez analizadas las oraciones, ahora se analizan todas juntas, tomando en cuenta la situación de cada oración, analizando las oraciones anteriores, una vez realizado este paso, la computadora ya sabe que es lo que va a hacer, es decir, ya tiene la expresión final.
- Una vez obtenida la expresión final, el siguiente paso es la ejecución de esta, para obtener así el resultado y poder proporcionárselo al usuario.

2.5. Recursos abiertos para aplicar el Procesamiento de Lenguaje Natural

Hoy en día, existe una infinidad de recursos abiertos para implementar las técnicas de Procesamiento de Lenguaje Natural. Por ejemplo, el Natural Language Toolkit de Python es una plataforma para construir programas que permiten manipular lenguaje natural, además de proporcionar más de 50 recursos que pueden ser utilizados en la fase de entrenamiento. En este mismo sentido, se encuentra disponible de manera gratuita el set de herramientas para el análisis de lenguaje natural de Stanford, CoreNLP. Adicionalmente, se han puesto a disposición otros modelos más complejos como TextSum de TensorFlow, que a través del uso de redes neuronales genera un titular tomando como insumo el primer párrafo de una noticia.

Finalmente, para las personas que quieran ahondar en este tema, existen cursos abiertos de Procesamiento de Lenguaje Natural. Uno de los cursos más famosos para un público con conocimiento avanzado en matemáticas, es el curso en inglés de la Universidad de Stanford sobre Aprendizaje Profundo aplicado al Procesamiento de Lenguaje Natural.

2.6. Problema del procesamiento de lenguaje natural

La principal dificultad en los procesos de recuperación de información mediante lenguajes formales no es de índole técnica sino psicológica: entender cuál es la necesidad real del usuario, cual es la correcta formulación de su pregunta o necesidad. La dirección más prometedora de resolver este problema es el uso de lenguaje natural. Sin embargo, uno de los grandes problemas del PLN se produce cuando una expresión en LN posee más de una interpretación, es decir, cuando en el lenguaje de destino se le pueden asignar dos o más expresiones distintas.

Este problema de la ambigüedad se presenta en todos los niveles del lenguaje, sin excepción. Ejemplo: “Hay alguien en la puerta, que te quiere hablar” “Hay alguien, en la puerta que te quiere hablar” No está claro, si el predicado “te quiere hablar” se adjudica a “alguien” o a “la puerta”, sabemos que la puertas no hablan, por tanto deducimos que es a alguien. Pero esto no lo puede deducir la máquina, a no ser que esté enterada de lo que hacen o no hacen las puertas. En apariencia este problema es demasiado sencillo, pero en realidad, es uno de los más complicados y que más complicaciones ha dado para que el PLN pueda desarrollarse por completo, ya que al presentarse en todos los niveles del lenguaje, se tienen que desarrollar programas (lenguaje formal) para solucionarlos en cada caso.

2.7. Historia de PLN

2.7.1. La lingüística teórica y el PLN en los años 40 y 50

La investigación en PLN se remonta a los años 40, siendo la traducción automática una de sus primeras aplicaciones. en 1949, el interés por la traducción automática despertó gracias a un famoso memorándum del matemático Warren Weaver, en el cual se propone la aplicación de las técnicas del desciframiento criptográfico, los métodos estadísticos y la teoría de la información, para la traducción automática de textos con el fin de “solucionar los problemas mundiales de traducción”. en realidad, esta idea no era tan original como parecía, ya que una de las tareas habituales de un ordenador ha sido siempre la traducción del código escrito en un lenguaje informático de alto nivel al lenguaje máquina, el cual se limita a una secuencia de ceros y unos.

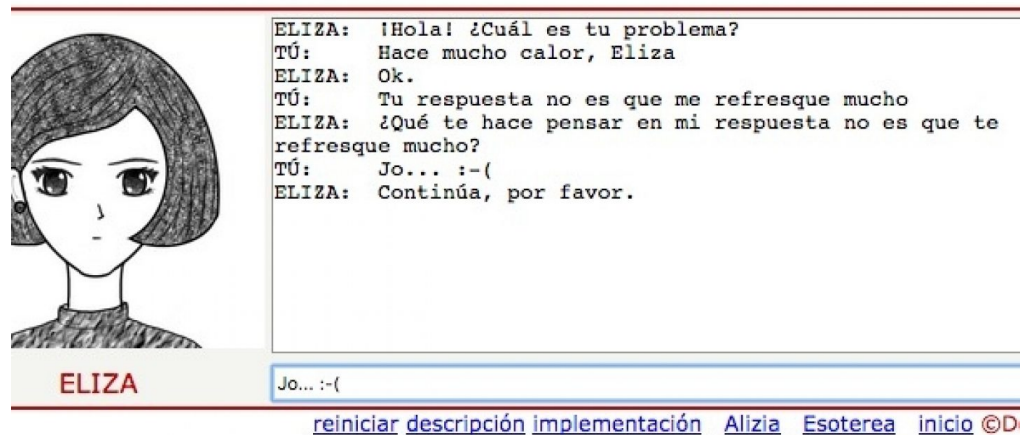
2.7.2. Los años 60

El nacimiento de la Inteligencia Artificial uno de los hitos en las investigaciones en PLN en la década de los 60 fue el nacimiento de la inteligencia

artificial, la cual tuvo lugar en un seminario de dos meses organizado por John mccarthy y celebrado en el verano de 1956 en el Dartmouth college en Hanover, new Hampshire:



Las discusiones intelectuales que se mantuvieron en este seminario, en el que participaron los investigadores pioneros en inteligencia automática, tales como marvin minsky, nathaniel Rochester y claud shannon, entre otros muchos, sirvieron para colocar los pilares de la nueva disciplina de la Inteligencia Artificial. uno de los aspectos del problema de la inteligencia artificial que se trató fue la posibilidad de que un ordenador pudiera ser programado para utilizar una lengua, en concreto el inglés. Especulaba con la idea de que la mayor parte del pensamiento humano consistía en la manipulación de palabras según un conjunto de reglas de razonamiento. Por tanto, y debido a la marcada formación matemática de los organizadores, el centro de interés de estas actividades radicó principalmente en el desarrollo de sistemas basados en el razonamiento lógico. con respecto a la comprensión automática del lenguaje, uno de los sistemas de diálogo más representativos en la década de los 60 fue ELIZA (Weizenbaum, 1966).



El cual simulaba ser un psicoterapeuta que mantenía una conversación con el usuario. el algoritmo que simulaba la inteligencia de Eliza consistía básicamente en leer una oración de entrada, buscar la presencia de un patrón a modo de plantilla predefinida (i.e. constantes y variables), el cual se activaba a través de una palabra clave, y finalmente transformar la entrada en una respuesta. Por tanto, el sistema elegía una respuesta al azar de las respuestas asociadas al patrón, incorporando igualmente expresiones utilizadas en el propio texto de entrada.

2.8. Ambigüedad



Las lenguas naturales son inherentemente ambiguas en diferentes niveles:

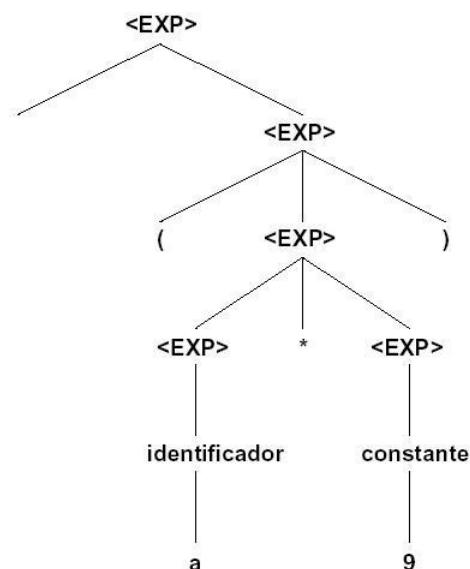
- En el nivel léxico, una misma palabra puede tener varios significados, y la

selección del apropiado se debe deducir a partir del contexto oracional o conocimiento básico. Muchas investigaciones en el campo del procesamiento de lenguajes naturales han estudiado métodos de resolver las ambigüedades léxicas mediante diccionarios, gramáticas, bases de conocimiento y correlaciones estadísticas.

- A nivel referencial, la resolución de anáforas y catáforas implica determinar la entidad lingüística previa o posterior a que hacen referencia.
- En el nivel estructural, se requiere de la semántica para desambiguar la dependencia de los sintagmas preposicionales que conducen a la construcción de distintos árboles sintácticos. Por ejemplo, en la frase Rompió el dibujo de un ataque de nervios.
- En el nivel pragmático, una oración, a menudo, no significa lo que realmente se está diciendo. Elementos tales como la ironía tienen un papel importante en la interpretación del mensaje.

Para resolver estos tipos de ambigüedades y otros, el problema central en el PLN es la traducción de entradas en lenguas naturales a una representación interna sin ambigüedad, como árboles de análisis.

1. $\langle \text{EXP} \rangle ::= \langle \text{EXP} \rangle + \langle \text{EXP} \rangle$
2. $\langle \text{EXP} \rangle ::= \langle \text{EXP} \rangle * \langle \text{EXP} \rangle$
3. $\langle \text{EXP} \rangle ::= \langle \text{EXP} \rangle - \langle \text{EXP} \rangle$
4. $\langle \text{EXP} \rangle ::= \langle \text{EXP} \rangle / \langle \text{EXP} \rangle$
5. $\langle \text{EXP} \rangle ::= \langle \text{EXP} \rangle ^ \langle \text{EXP} \rangle$
6. $\langle \text{EXP} \rangle ::= - \langle \text{EXP} \rangle$
7. $\langle \text{EXP} \rangle ::= (\langle \text{EXP} \rangle)$
8. $\langle \text{EXP} \rangle ::= \text{identificador}$
9. $\langle \text{EXP} \rangle ::= \text{constante}$



2.9. Adquisición de conocimiento léxico

El gran problema al que se enfrentan en el diseño de sistemas de lenguaje natural a gran escala, es el gran número de unidades léxicas de las lenguas naturales, así como a la constante incursión de palabras nuevas o nuevas acepciones

de palabras existentes. La adquisición de la información léxica necesaria para lexicones computacionales plantea serios problemas, tanto en lo que se refiere a la efectividad de los diferentes métodos que se han empleado como a la inversión de tiempo, dinero y recursos humanos y computacionales que estos métodos requieren. Se puede considerar que existen tres métodos o fuentes principales para la adquisición de conocimiento léxico:

1. Adquisición manual de información léxica.
2. Diccionarios en formato magnético (MRDs).
3. Los córpora textuales informatizados.

Los tres métodos plantean ventajas y desventajas, tanto en lo que se refiere a los recursos que requieren como a la efectividad que han demostrado hasta ahora. Aunque en principio las fuentes electrónicas pueden aportar una gran cantidad de información lingüística muy valiosa, que puede servir como punto de partida para la creación de una base de datos léxica, en la práctica es difícil aprovechar toda la información que esas fuentes electrónicas contienen. Una de las dificultades, y quizás la principal, es que los diccionarios están diseñados por humanos (y no máquinas) para ser usados por humanos. Los usuarios (humanos) son hablantes nativos de una lengua, que conocen el contexto de lo que se está hablando, y saben implícitamente, cómo está estructurado el lexicon de su lengua. Los MRDs, en muchas ocasiones, son elaborados por lexicógrafos, quienes explotan el conocimiento lingüístico de sus usuarios potenciales, de modo que las entradas de un diccionario contienen solo la información necesaria para que un hablante de una lengua sea capaz de conectarla con su conocimiento lingüístico general.

Karen Sparck-Jones demostró en un estudio realizado que los diccionarios deben contener un componente de circularidad, ya que cada palabra usada en las definiciones ha de ser, a su vez, definida en el diccionario. Algunas de estas circularidades mantienen una distancia semántica reducida, como por ejemplo las definiciones mutuas de “bueno” y “excelente”, y son por tanto fáciles de observar y asimilar por un lector humano, pero son muy difíciles de localizar a nivel formal lo cual dificulta la labor de extracción de información de las definiciones.

El lexicon se considera como un “diccionario mental” en el que se registran las palabras que conoce un hablante. Este “diccionario” especifica los rasgos característicos de los componentes léxicos (palabras y morfemas), como irregularidades morfológicas, requerimientos sobre alomorfos, información pragmática, etc. Un símbolo alomorfo se refiere a cada uno de las diferentes formas fonológicas que puede tener un morfema abstracto. Estrictamente la realización fonológica concreta de un morfema se llama morfo, si existe más de un morfo para el mismo morfema entonces usamos el término alomorfo. Algunos modelos gramaticales formales basan la generación de oraciones en el procesamiento de los rasgos de las unidades del lexicon.

En estos modelos, el lexicon no es parte de la gramática, sino que proyecta sus rasgos a través de mecanismos inherentes a las gramáticas. La finalidad

fundamental del procesamiento de lenguaje natural es la automatización de los procesos lingüísticos, tales como la comprensión, producción o adquisición de una lengua, tareas que los usuarios de una lengua realizan fluida y naturalmente. Esto hace converger intereses de varias disciplinas como son lingüistas computacionales, psicolingüistas, informáticos e ingenieros de sistemas. Todos ellos, desde diferentes perspectivas teóricas y prácticas, intentan desarrollar una teoría que sea totalmente explícita (y por tanto automatizable) de los procesos lingüísticos.

La mayoría de los sistemas de procesamiento de lenguaje natural adoptan un enfoque denominados “basado en el conocimiento” (knowledge-based), ya que para llevar a cabo la tarea para la que están diseñados, necesitan incorporar conocimiento lingüístico explícito, junto con otros tipos de conocimiento de carácter más general. Por ejemplo, un sistema que convierta un texto en su correspondiente cadena hablada, necesita “conocimiento” sobre la pronunciación de las letras, así como de las palabras individuales que no siguen las reglas generales. También precisa conocimiento sobre los patrones rítmicos de acentuación y de cómo la organización sintáctica afecta la entonación y prosodia. Atendiendo estas consideraciones, con el objetivo de consensuar en la investigación sobre el PLN, se ha dividido su estudio en subsistemas, en relación con los niveles presentados en la arquitectura de un sistema de PLN, identificando cinco tipos de conocimiento:

1. Conocimiento Fonológico: Información sobre el sistema de sonidos y la estructura de las palabras y las expresiones, los patrones de acentuación, la entonación, etc.
2. Conocimiento morfológico: Información sobre la estructura de las palabras; por ejemplo, que los fonemas /s/ y /z/ se añaden en inglés a los nombres para formar el plural.
3. Conocimiento sintáctico: Información sobre las reglas sintácticas y/o gramaticales.
4. Conocimiento semántico: Información sobre el significado que se da a las diversas construcciones sintácticas y de cómo esos significados se combinan para formar el significado de las oraciones.
5. Conocimiento pragmático: Información central en muchas tareas específicas como por ejemplo, la recuperación de los referentes de los pronombres y las intenciones comunicativas que subyacen en una frase en particular, el análisis de las presuposiciones del hablante.

2.10. La comprensión del lenguaje en Inteligencia Artificial

La inteligencia Artificial de los años 70 se orientó principalmente hacia el desarrollo de sistemas de comprensión del lenguaje natural. La escuela de Yale, i.e. Roger Schank y sus colaboradores (Schank, 1972, 1975; Schank y Abelson, 1977), lideró las investigaciones en este campo, incorporando la teoría del guión

como base para un modelo dinámico de la memoria. este modelo de comprensión del lenguaje, el cual resultó ser una importante influencia en la semántica y la representación del conocimiento, se basó en el formalismo de la Dependencia conceptual (schank, 1972), i.e. grafos que permiten representar conceptualmente un texto de entrada a partir de la descomposición semántica de los verbos usando la Gramática de casos de Fillmore.

Un sistema de comprensión automática del inglés que podía manipular bloques de juguete sobre una mesa a partir de unas órdenes, además de poder ser interrogado sobre el escenario resultante incorporó la teoría funcional de Halliday. este sistema utilizaba información semántica y del contexto para comprender el discurso, ya que se basaba en la idea de que no es posible construir un sistema informático razonablemente inteligente a menos que pueda comprender el tema sobre el cual está trabajando, lo cual implicaba proporcionarle un modelo detallado del conocimiento que requería. no obstante, otras interfaces de diálogo persona-máquina en esta década no requirieron un complejo formalismo de representación del significado con el fin de simular la interacción comunicativa.

Éste es el caso de PARRY (colby, 1973), un sistema que permitía simular las respuestas de un paciente paranoico que sufría la psicosis de que lo estaba persiguiendo la mafia. Aunque inspirado en el trabajo de eliza, PARRY no repetía las palabras de sus entrevistadores, sino que contribuía a la conversación de forma fluida además de reaccionar tal y como lo haría un paranoico. A pesar de que SHREDI y PARRY coincidían en ser sistemas basados en el diálogo, las diferentes concepciones de sus autores sirven para representar dos enfoques muy diferentes de entender el PLN. el primero empleaba un análisis lingüístico basado en un modelo teórico gramatical incorporando igualmente conocimiento del mundo sobre el cual se aplicaba un razonamiento basado en la lógica. en cambio, el segundo apostaba por el simple reconocimiento de patrones en la superficie de la entrada textual y un módulo de interpretación-acción cuyas reglas de producción permitían recrear el modelo de paranoia.

2.11. Las expresiones regulares en PLN y la búsqueda de patrones

La tarea del reconocimiento de entidades tradicionalmente está compuesto por dos sub-tareas: la detección o identificación y la clasificación de las entidades. En la detección de las entidades es donde tiene lugar la utilización de expresiones regulares, pues permiten codificar las estructuras que marcan los patrones detectados en el texto. Para cada patrón o grupo de patrones que se intente detectar, existirá al menos una regla que lo define.

La extracción de información es el proceso de “localizar las porciones de un texto dado que contengan información relevante para las necesidades de un usuario y proporcionar dicha información de forma adecuada a su proceso (manual o automático)”. Típicamente un sistema de extracción de información extrae informaciones sobre entidades, relaciones y eventos a partir de documentos en un dominio restringido. En la fase de detección de la información relevante también

es necesaria la utilización de reglas que determinen los patrones que correspondan a la información que se intenta detectar. Por ejemplo, se desea detectar, etiquetar y extraer la información referente a las entidades de tipo persona y moneda. Para el ejemplo: Juan tiene \$5,00 y Pedro tiene \$4,00. Se necesitaría establecer reglas que deriven en expresiones regulares que modelen el patrón correspondiente a cada entidad buscada. En la Tabla 1 se muestran las expresiones implicadas para el ejemplo anterior y cómo tributa a cada una de las técnicas.

Para establecer las reglas se propone un método que a su vez se materializa en una herramienta informática capaz de conformar fácilmente autómatas finitos y obtener su correspondientes reglas como expresión regular.

Técnicas	Expresiones regulares		Resultados	
	Moneda	Nombre		
Detección	\d+[.,]\d\d	[A-Z][a-z] +	\$5,00 \$4,00	
Etiquetado			<nombre>Juan</nombre> tiene <moneda>\$5,00</moneda> y <nombre>Pedro</nombre> tiene <moneda>\$4,00</moneda>.	
Extracción de información				
			Nombre	Moneda
			Juan	\$5,00
	Pedro	\$4,00		

3. Análisis de sentimientos

El análisis de sentimientos es una tarea incluida en el ámbito del Procesamiento de Lenguaje Natural o NLP (Natural Language Processing). Su objetivo es encontrar contenido subjetivo en los textos de entrada. El análisis de sentimientos busca extraer opiniones y la polaridad de éstas, mediante la identificación de características que determinan cuán positivo o negativo es el texto.

El análisis de sentimientos ha sido considerado inicialmente como una subdisciplina de clasificación basada en la opinión. La técnica en general puede estar orientada a tres grandes tipos de tareas:

- Clasificación de sentimientos: realizar una clasificación de un conjunto de opiniones en tres categorías: positivas, negativas o neutrales. Presenta desafíos adicionales el hecho de que las opiniones se encuentran en múltiples idiomas o provienen de varios dominios, como biología, sociología, etc.
- Clasificación de subjetividad: determinar si una oración es subjetiva u objetiva. Una oración objetiva contiene información imparcial, mientras que una oración subjetiva contiene información de carácter personal como opiniones.
- Resumen de opinión: permitir extraer las características principales que son compartidas por uno o más documentos y el sentimiento acerca de estas características.

En el análisis de redes sociales, particularmente en análisis de sentimientos a partir del estudio de tweets para obtener su polaridad, existen dos métodos que son los más populares. El primero de ellos utiliza el enfoque del aprendizaje computacional (machine-learning approaches) y el segundo utiliza diccionarios léxicos.

El aprendizaje computacional analiza la información automáticamente de forma supervisada, basándose en conjuntos de entrenamiento que son utilizados para catalogar al resto de las opiniones encontradas en la web, realizando pruebas y luego validándose. Las principales técnicas de este método son: Support Vector Machines (SVM), Naive Bayes y Clasificadores de Máxima Entropía. En estas técnicas se utiliza la categoría gramatical de las palabras, la presencia y frecuencia de algunos términos y su composición semántica. Sin embargo, la mayoría de estos métodos son acompañados de algún diccionario que entrega información a priori de los términos para obtener las polaridades respectivas. En algunos casos, estos diccionarios son realizados por personas y en otros se ocupa un sistema semiautomático. El método de diccionarios léxicos se basa en una lista de palabras con un determinado peso y/o categoría emocional. Estos diccionarios presentan principalmente adjetivos, que son los que aportan mayor información al momento de analizar los sentimientos, aunque también incluye verbos, adverbios y sustantivos.

La mayor cantidad de versiones de diccionarios está en idioma inglés aunque existen algunas en español, pero en general en versión beta. Éstos permiten

determinar si una frase es negativa o positiva dependiendo de la cantidad de palabras presentes en el diccionario y de la fuerza de su sentimiento. Uno de los diccionarios más completos es LIWC, presente una versión en inglés bastante completa y una en español que está aún en etapa de desarrollo. En este caso, las palabras son etiquetadas en una categoría determinada y además se les asigna una ponderación.

3.1. Tipos de análisis de sentimiento



A la hora de extraer esta información, hay una gran variedad de métodos y algoritmos dependiendo del nivel de granularidad del análisis que queramos llevar a cabo. Se distinguen tres niveles: nivel de documento, de oración o de aspecto.

El análisis a nivel de documento determina el sentimiento general expresado en un texto, mientras que el análisis a nivel de frase lo especifica para cada una de las oraciones del texto. Sin embargo, estos dos tipos de análisis no profundizan en detalle el elemento que a las personas les gusta o no. No especifican sobre qué es la opinión, ya que considerando la opinión general de un objeto como positiva (o negativa) no significa que el autor tenga una opinión positiva (o negativa) de todos los aspectos de dicho objeto.

Para este trabajo nos enfocamos en realizar un análisis a nivel de documento, ya que, debido al límite en los mensajes, los autores suelen ser concisos y van directo al grano sin tener la posibilidad de incluir varios aspectos diferenciados en un solo tweet. Por esta razón, usar el tweet como unidad de análisis parece proveer un nivel de granularidad adecuado para hacer un análisis de sentimiento desglosado.

3.1.1. Análisis a nivel de documento

Considerada como una de las tareas más simples de la minería de opiniones, el análisis a nivel de documento apunta a clasificar la opinión de un documento, en este caso un tweet. Esta tarea no considera los detalles en cuanto a entidades o aspectos, sino que considera el documento como un todo, el cual será etiquetado como positivo o negativo. Esta puede ser considerada como una tarea tradicional de clasificación de texto, donde las clases son las diferentes orientaciones en cuanto a los sentimientos. No obstante, para asegurar que este tipo de análisis tenga sentido asumimos que cada documento expresa una única opinión sobre

una única entidad. Si bien esto puede parecer una limitación, porque en un tweet uno podría expresar más de una opinión hacia distintas entidades, en la práctica produce resultados positivos, ya que los usuarios suelen enfocarse en un único aspecto en cada tweet. Seguramente en otros contextos, o si no estuviera limitado el largo de los mensajes, sería una buena idea considerar sistemas de análisis más complejos que permitan realizar un análisis con mayor granularidad. Las palabras que conforman las opiniones son el factor determinante en el análisis de sentimientos también es una buena opción utilizar métodos de aprendizaje basados en el uso de lexicones. Estos son diccionarios que contienen listados de palabras etiquetadas con el sentimiento asociado correspondiente, en algunos casos por un valor que también indica la intensidad del mismo. Luego, dada una opinión, simplemente podemos buscar el valor de todas las palabras que la componen y sumar el valor asociado para finalmente clasificar la opinión como positiva o negativa según corresponda.

4. Aprendizaje Computacional

Cuando el ser humano adquiere conocimientos, habilidades, actitudes o valores a través del estudio, de la experiencia o la enseñanza, se dice que aprende. Este proceso es fácil para el humano, sin embargo, lograr que una máquina aprende cómo lo hacen las personas es una interrogante que existe desde los inicios de las computadoras. Actualmente, no existe una máquina capaz de aprender de la misma manera que lo hace el hombre, sin embargo, se han creado algoritmos eficaces para algunas tareas de aprendizaje. En términos muy generales, se puede decir, que un programa aprende, si el desempeño obtenido para realizar alguna tarea, mejora con la experiencia. Se puede decir entonces que el Aprendizaje Computacional estudia los procesos computacionales que hay detrás del aprendizaje en humanos y en las máquinas. Esta disciplina juega un papel importante en muchas áreas de la ciencia.

4.1. Tipos de Aprendizaje

Un aspecto importante que influye en el aprendizaje es el grado de supervisión. En algunos casos, un experto en el dominio proporciona al aprendiz retroalimentación acerca de lo que es apropiado para su aprendizaje. En otros casos, a diferencia del aprendizaje supervisado, ésta retroalimentación está ausente, dando lugar al aprendizaje no-supervisado.

Y en otros casos, se combinan el aprendizaje supervisado y el no supervisado, dando lugar al aprendizaje semi-supervisado.

4.1.1. Aprendizaje supervisado

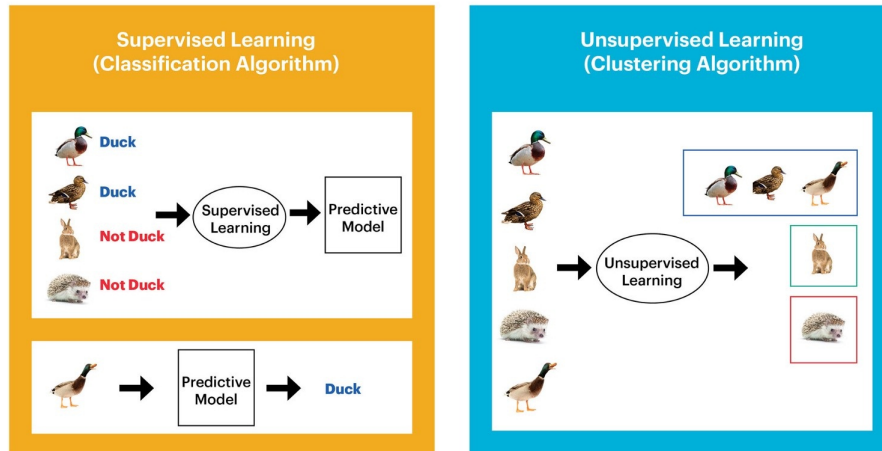
El aprendizaje supervisado es aquel en donde se intenta aprender de ejemplos como si estos fueran un maestro. Se asume que cada uno de estos ejemplos incluye características o atributos que especifican o definen a qué categoría o clase pertenece, de un conjunto de categorías o clases predefinidas, de esta manera, cada ejemplo se asocia con su clase. Este tipo de aprendizaje es llamado supervisado por la presencia de los ejemplos para guiar el proceso de aprendizaje. Al conjunto de ejemplos del cual se intenta aprender se le llama conjunto de entrenamiento. Usando estos datos se construye un modelo de predicción, o aprendiz, el cual permitirá predecir la clase para nuevos objetos no vistos por el aprendiz. La construcción del modelo se logra gracias a un algoritmo de aprendizaje.

Los algoritmos comúnmente utilizados son Naive Bayes, Máquinas de Soporte Vectorial, Vecinos más cercanos, J48 entre otros. En particular, en el área de Clasificación de Textos, los algoritmos típicamente utilizados son Naive Bayes y Máquinas de Soporte Vectorial. Este tipo de aprendizaje tiene la ventaja de que no es necesario que al aprendiz se le muestren todos los ejemplos existentes, es decir que puede clasificar un ejemplo sin haberlo visto nunca. La desventaja es que a pesar de lo anterior, sí es necesaria una gran cantidad de ejemplos para el entrenamiento.

4.1.2. Aprendizaje no supervisado

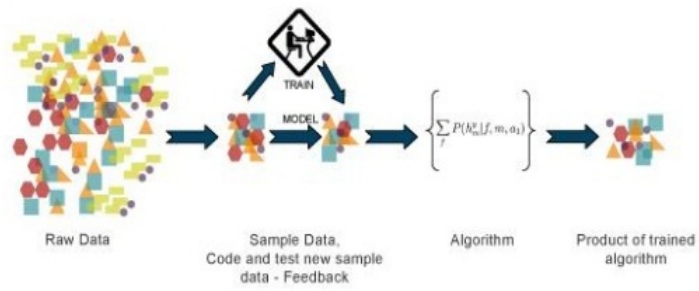
Este tipo de aprendizaje no presupone ningún conocimiento previo sobre lo que se quiere aprender. Tampoco existe un maestro que conozca los conceptos a aprender, por esta razón a este tipo de aprendizaje se le denomina Aprendizaje no-supervisado.

La diferencia es que, los ejemplos sólo incluyen los atributos, es decir, no se encuentran asociados a una clase. Para este caso la tarea se enfoca en descubrir patrones comunes entre los datos, que permitan separar los ejemplos en clases o jerarquías de clases. De éstas se podrán extraer caracterizaciones, o permitirán predecir características, o deducir relaciones útiles, a lo que se denomina como agrupación (clustering). Algunos de los algoritmos más comunes son: Cobweb, EM, y K Means. Siendo este último el más utilizado en el área de Clasificación de Textos. Este tipo de aprendizaje tiene la ventaja de que no es necesaria la presencia de un maestro para el aprendizaje o de un conjunto de entrenamiento.



4.1.3. Aprendizaje semi-supervisado

El aprendizaje semi-supervisado es la combinación del aprendizaje supervisado y el no-supervisado. En éste se aprende con la ayuda de dos conjuntos. Uno que contiene datos asociados a una clase, y el otro que contiene datos no asociados a una clase. La idea es aprender con los datos asociados a su clase y asociar una clase a los datos que no contienen asociada una clase. Algunos de los algoritmos más comunes son: Co-training, ASSEMBLE y self-training.



5. La Red Social Twitter

Twitter se creó en 2006 como una red social en base a contenido con forma de SMS. Desde entonces ha crecido rápidamente ganando mucha popularidad en los últimos años. Según los datos oficiales que figuran en su web son aproximadamente 326 millones de usuarios activos por mes que acceden diariamente para compartir experiencias y opiniones convirtiéndose así en una herramienta ideal para la realización de encuestas y sondeos.

Twitter permite a los usuarios enviar y leer mensajes de texto de hasta 140 caracteres, conocidos como tweets. Este sitio es una gran fuente de información subjetiva en tiempo real ya que estos millones de usuarios comparten opiniones sobre diferentes aspectos de su vida cotidiana. Twitter ha sido usada para una variedad de propósitos en diferentes industrias y situaciones. Por ejemplo, los usuarios pueden encontrar o emitir opiniones sobre un producto o servicio de su interés, las compañías y figuras públicas pueden controlar su reputación en línea, se puede conocer la opinión de cada usuario respecto a implementación de políticas públicas, campañas de difusión, entretenimiento, entre otras.

Twitter permite identificar estos temas a través de los denominados hashtag o etiquetas, que se caracterizan por comenzar con el carácter # y una cadena de caracteres a continuación formada por una o varias palabras concatenadas. Esta etiqueta funciona como metadato y permite que los tweets se añadan a una lista de mensajes con hashtag similares. Esto permite a los usuarios obtener resultados rápidamente sobre un mismo tema.

Parte III. Software utilizado en este proyecto

6. Node.Js



Node.Js o también conocido como Node, es un entorno de ejecución de Javascript en backend multiplataforma, código abierto. Esto permite la ejecución de código Javascript fuera del navegador web, y poder escribir programas de líneas de comando o scripting para servidores (como por ejemplo la creación de paginas web dinámicas).

Sumado a esto, Node.js representa el paradigma "Javascript en todos lados", unificando el desarrollo de las aplicaciones web en un solo lenguaje, en vez de utilizar diferentes lenguajes para el scripting en servidor o en el cliente.

Node.js posee una arquitectura orientada a los eventos, capaz de realizar entradas y salidas asíncronas. Esto permite optimizar el consumo y escalabilidad de aplicaciones web en muchas operaciones de E/S, como también aplicaciones en tiempo real.

El proyecto de desarrollo distribuido de Node.js, a manos de la Node.js Foundation, es facilitada por el programa de proyectos colaborativos de la Linux Foundation. Algunas de las grandes empresas que utilizan software construido en Node.js son: GoDaddy, Groupon, IBM, LinkedIn, Microsoft, Netflix, PayPal, Walmart, entre otros.

6.1. Descripción general

Node.js permite la creación de servidores web y herramientas de red utilizando JavaScript y una colección de "módulos" que manejan varias funciones centrales. Se proporcionan módulos para la E / S del sistema de archivos, redes (DNS, HTTP, TCP, TLS / SSL o UDP), datos binarios (buffers), funciones de criptografía, flujos de datos, y otras funciones básicas. Los módulos de Node.js utilizan una API diseñada para reducir la complejidad de escribir aplicaciones de servidor.

Aunque inicialmente el sistema de módulos se basaba en el patrón de módulos commonjs, la reciente introducción de módulos en la especificación ECMAScript ha cambiado la dirección de usar módulos ECMAScript en Node.js de forma predeterminada.

Node.js es oficialmente compatible con Linux, macOS y Microsoft Windows 7 y Server 2008 (y posteriores) por parte del equipo. También hay soporte de nivel 2 para SmartOS e IBM AIX. Y soporte experimental para FreeBSD; OpenBSD también funciona y, por ejemplo, las versiones LTS disponibles para IBM i (AS/400). El código fuente provisto también puede construirse en sistemas operativos similares a los que son oficialmente compatibles o ser modificado por terceros para que sean compatibles con otros, como los servidores NonStop y Unix. Alternativamente, se puede escribir con CoffeeScript (una alternativa de JavaScript), Dart o TypeScript (formas de JavaScript fuertemente tipadas), o cualquier otro lenguaje que se pueda compilar en JavaScript.

Node.js se utiliza principalmente para crear programas de red como los servidores web. La diferencia más significativa entre Node.js y PHP es que la mayoría de las funciones en el bloque PHP hasta su finalización (los comandos solo se ejecutan después de que terminan los comandos anteriores), mientras que las funciones de Node.js no se bloquean (los comandos se ejecutan simultáneamente o incluso en paralelo, y use devoluciones de llamada para indicar la finalización o el fallo).

6.1.1. Arquitectura de la plataforma

Node.js lleva la programación dirigida por eventos a los servidores web, lo que permite el desarrollo de servidores web rápidos en JavaScript. Los desarrolladores pueden crear servidores escalables sin utilizar subprocesos, mediante el uso de un modelo simplificado de programación dirigida por eventos que utiliza devoluciones de llamada para indicar la finalización de una tarea. Node.js conecta la facilidad de un lenguaje de scripting (JavaScript) con el poder de la programación en red de Unix.

Node.js fue construido en el motor de JavaScript Google V8 ya que fue de código abierto bajo la licencia BSD. Es competente con los fundamentos de Internet como HTTP, DNS, TCP. JavaScript también era un lenguaje muy conocido, lo que hacía que Node.js fuera accesible para la comunidad de desarrollo web.

6.1.2. Soporte a la industria

Hay miles de bibliotecas de código abierto para Node.js, la mayoría de ellas alojadas en el sitio web de npm. La comunidad de desarrolladores de Node.js tiene dos listas de correo principales y el canal IRC # node.js en freenode. Existen múltiples conferencias y eventos de desarrolladores que son compatibles con la comunidad Node.js, incluidos NodeConf, Node Interactive y Node Summit, así como varios eventos regionales.

La comunidad de código abierto ha desarrollado marcos web para acelerar el desarrollo de aplicaciones. Estos marcos incluyen Connect, Express.js, Socket.IO, Feathers.js, Koa.js, Hapi.js, Sails.js, Meteor, Derby y muchos otros. También se han creado varios paquetes para interactuar con otros idiomas o entornos de tiempo de ejecución como Microsoft .NET.

Los IDE de escritorio modernos proporcionan funciones de edición y depuración específicamente para las aplicaciones Node.js. Tales IDE incluyen Atom, corchetes, JetBrains WebStorm, Microsoft Visual Studio (con Node.js Tools para Visual Studio, o TypeScript con definiciones de nodo), NetBeans, Nodeclipse, Enide Studio (basado en Eclipse), y Visual Studio Code.

Ciertos IDE en línea basados en la web también son compatibles con Node.js, como Codeanywhere, Codenvy, Cloud9 IDE, Koding y el editor de flujo visual en Node-RED.

6.2. Historia

Node.js fue escrito inicialmente por Ryan Dahl en 2009, aproximadamente trece años después de la introducción del primer entorno de JavaScript del lado del servidor, LiveWire Pro Web de Netscape. La versión inicial solo era compatible con Linux y Mac OS X. Su desarrollo y mantenimiento fue dirigido por Dahl y, posteriormente, patrocinado por Joyent.

Dahl criticó las posibilidades limitadas del servidor web más popular en 2009, el Servidor HTTP Apache, para manejar muchas conexiones concurrentes (hasta 10,000 o más) y la forma más común de crear código (programación secuencial).

Dahl demostró el proyecto en la JSConf europea inaugural el 8 de noviembre de 2009. Para ello combinó el motor de JavaScript V8 de Google, un ciclo de eventos y una API de E/S de bajo nivel.

En enero de 2010, se introdujo un administrador de paquetes para el entorno Node.js llamado npm. El administrador de paquetes facilita a los programadores publicar y compartir el código fuente de las bibliotecas Node.js y está diseñado para simplificar la instalación, actualización y desinstalación de las bibliotecas.

En junio de 2011, Microsoft y Joyent implementaron una versión nativa para Windows de Node.js. La primera compilación de Node.js compatible con Windows se lanzó en julio de 2011.

En enero de 2012, Dahl se hizo a un lado, promoviendo al compañero de trabajo y creador de npm Isaac Schlueter para gestionar el proyecto. En enero de 2014, Schlueter anunció que Timothy J. Fontaine lideraría el proyecto.

En diciembre de 2014, Fedor Indutny comenzó io.js, una bifurcación de Node.js. Debido al conflicto interno sobre el liderazgo de Joyent, io.js se creó como una alternativa de gobierno abierto con un comité técnico independiente. A diferencia de Node.js, los autores planeaban mantener io.js actualizado con las últimas versiones del motor de JavaScript Google V8.

En febrero de 2015, se anunció la intención de formar una Fundación Node.js neutral. Para junio de 2015, las comunidades Node.js y io.js votaron para trabajar juntas bajo la Fundación Node.js.

En septiembre de 2015, Node.js v0.12 y io.js v3.3 se fusionaron de nuevo en Node v4.0. Esta combinación trajo las características del V8 ES6 a Node.js y un ciclo de lanzamiento de soporte a largo plazo. A partir de 2016, el sitio web io.js recomienda que los desarrolladores vuelvan a Node.js y que no se planifiquen más lanzamientos de io.js debido a la fusión.

6.3. Versiones

Las nuevas versiones principales de Node.js se eliminan de la rama maestra de GitHub cada seis meses. Las versiones con números pares se terminan en abril y las versiones con números impares en octubre. Cuando se lanza una nueva versión impar, la versión anterior experimenta la transición al Soporte a largo plazo (LTS), que le otorga a la versión 18 meses de soporte activo desde la fecha en que se designó LTS. Después de que expiren estos 18 meses, un lanzamiento de LTS recibe 12 meses adicionales de soporte de mantenimiento. Una versión activa recibe backports de cambios ininterrumpidos unas semanas después de su lanzamiento en la versión actual. Una versión de mantenimiento solo recibe correcciones críticas y actualizaciones de documentación. El Grupo de trabajo LTS gestiona la estrategia y la política en colaboración con el Comité Directivo Técnico de la Fundación Node.js.

Version	Nombre clave	Fecha de lanzamiento	Estado de LTS	Inicio de LTS activo	Inicio de mantenimiento	Fin de mantenimiento
v0.10.x		2013-03-11	Fin de ciclo	-	2015-10-01	2016-10-31
v0.12.x		2015-02-06	Fin de ciclo	-	2016-04-01	2016-12-31
4.x	Argon	2015-09-08	Fin de ciclo	2015-10-01	2017-04-01	2018-04-30
5.x		2015-10-29	No LTS	N/A		2016-06-30
6.x	Boron	2016-04-26	Fin de ciclo	2016-10-18	2018-04-30	2019-04-30
7.x		2016-10-25	No LTS	N/A		2017-06-30
8.x	Carbon	2017-05-30	Mantenimiento	2017-10-31	2019-01-01	2019-12-31
9.x		2017-10-01	No LTS	N/A		2018-06-30
10.x	Dubnium	2018-04-24	Activo	2018-10-30	2020-04-01	2021-04-01
11.x		2018-10-23	No LTS	N/A		2019-06-30
12.x	Erbium	2019-04-23	Pendiente	2019-10-22	2021-04-01	2022-04-01

6.4. Aspectos técnicos

6.4.1. Hilos

Node opera en un ciclo de eventos en un único hilo de ejecución, usando llamadas E/S sin bloqueo, lo que permite miles de conexiones concurrentes sin el costo de cambio de contexto en el hilo de procesamiento. El diseño de compartir un solo hilo para todas las solicitudes está orientado a la construcción de aplicaciones altamente concurrentes, donde cada tarea de E/S debe utilizar una función callback.

Una desventaja de esta arquitectura es que no se puede realizar escalamiento vertical incrementando el número de núcleos en el CPU del servidor sin utilizar un módulo adicional, como un cluster por ejemplo. Sin embargo, algunos desarrolladores pueden incrementar el número de hilos por defecto en la librería libuv.

6.4.2. V8

V8 es el motor de ejecución de JavaScript inicialmente desarrollado para el navegador Google Chrome. Escrito en C++, V8 compila código fuente JavaScript en código máquina en vez de interpretarlo en tiempo real.

6.4.3. Gestor de paquetes NPM

NPM es el gestor de paquetes por defecto para Node.js. Permite de manera sencilla y eficiente la instalación de software Node.js que se encuentren registrados en el registro de programas de NPM. En este trabajo, se utilizarán varios paquetes de npm tales como: sentiment, Express.js, twitter, y mongoose.

6.4.4. Ciclo de eventos

Node.js se registra con el sistema operativo así el mismo notificará de las conexiones o problemas a una función retorno. Dentro de la ejecución de Node.js, cada conexión es una asignación en pila. Tradicionalmente, los sistemas operativos utilizan procesos o hilos para cada conexión, sin embargo, Node.js utiliza el ciclo de eventos para la escalabilidad. En contraste con otros servidores orientados a eventos, el ciclo de eventos de Node no necesita ser llamado explícitamente, en cambio las funciones retorno son definidas y el servidor automáticamente entra al ciclo de eventos al finalizar una definición de retorno.

7. Express

Express.js, o simplemente Express, es un framework de desarrollo de aplicaciones web para Node.js. Fue liberado como software de código abierto con la Licencia MIT. Está diseñado para la construcción de aplicaciones web y APIs. Ha sido llamado como el framework de servidor estándar de facto para Node.js.

El autor original, TJ Holowaychuk, lo describió como un servidor inspirado en Sinatra. Esto significa que es minimalista, y algunas de las varias funciones se encuentran disponibles a través de plugins.

7.1. ¿Qué popularidad tiene Node/Express?

La popularidad de un framework web es importante porque es un indicador de se continuará manteniendo y qué recursos tienen más probabilidad de estar disponibles en términos de documentación, librerías de extensiones y soporte técnico.

No existe una medida disponible de inmediato y definitiva de la popularidad de los frameworks de lado servidor (aunque sitios como Hot Frameworks intentan asesorar sobre popularidad usando mecanismos como contar para cada plataforma el número de preguntas sobre proyectos en GitHub y StackOverflow). Una pregunta mejor es si Node y Express son lo "suficientemente populares" para evitar los problemas de las plataformas menos populares. ¿Continúan evolucionando? ¿Puedes conseguir la ayuda que necesitas? ¿Hay alguna posibilidad de que consigas un trabajo remunerado si aprendes Express?

De acuerdo con el número de compañías de perfil alto que usan Express, el número de gente que contribuye al código base, y el número de gente que proporciona soporte tanto libre como pagado, podemos entonces decir que Express es un framework popular.

Express proporciona mecanismos para:

- Escritura de manejadores de peticiones con diferentes verbos HTTP en diferentes caminos URL (rutas).
- Integración con motores de renderización de "vistas" para generar respuestas mediante la introducción de datos en plantillas.
- Establecer ajustes de aplicaciones web como qué puerto usar para conectar, y la localización de las plantillas que se utilizan para renderizar la respuesta.
- Añadir procesamiento de peticiones "middleware" adicional en cualquier punto dentro de la tubería de manejo de la petición.

A pesar de que Express es en sí mismo bastante minimalista, los desarrolladores han creado paquetes de middleware compatibles para abordar casi cualquier problema de desarrollo web. Hay librerías para trabajar con cookies, sesiones, inicios de sesión de usuario, parámetros URL, datos POST, cabeceras de seguridad y muchos más. Puedes encontrar una lista de paquetes middleware mantenida por el equipo de Express en Express Middleware (junto con una lista de algunos de los paquetes más populares de terceros).

7.2. ¿Es Express dogmático?

Los frameworks web frecuentemente se refieren a sí mismos como "dogmáticos" ("opinionated") o "no dogmáticos" ("unopinionated").

Los frameworks dogmáticos son aquellos que opinan acerca de la "manera correcta" de gestionar cualquier tarea en particular. Ofrecen soporte para el desarrollo rápido en un dominio en particular (resolver problemas de un tipo en particular) porque la manera correcta de hacer cualquier cosa está generalmente bien comprendida y bien documentada. Sin embargo pueden ser menos flexibles para resolver problemas fuera de su dominio principal, y tienden a ofrecer menos opciones para elegir qué componentes y enfoques pueden usarse.

Los frameworks no dogmáticos, en contraposición, tienen muchas menos restricciones sobre el modo mejor de unir componentes para alcanzar un objetivo, o incluso qué componentes deberían usarse. Hacen más fácil para los desarrolladores usar las herramientas más adecuadas para completar una tarea en particular, si bien al coste de que necesitas encontrar esos componentes por uno mismo.

Express es no dogmático, transigente. Puede insertar casi cualquier middleware compatible que se desee dentro de la cadena de manejo de la petición, en casi cualquier orden que se quiera. Permite estructurar una app en un fichero o múltiples ficheros y usar cualquier estructura de directorios.

7.3. ¿Cómo es el código para Express?

En sitios web o aplicaciones web dinámicas, que accedan a bases de datos, el servidor espera a recibir peticiones HTTP del navegador (o cliente). Cuando se recibe una petición, la aplicación determina cuál es la acción adecuada correspondiente, de acuerdo a la estructura de la URL y a la información (opcional) indicada en la petición con los métodos POST o GET. Dependiendo de la acción a realizar, puede que se necesite leer o escribir en la base de datos, o realizar otras acciones necesarias para atender la petición correctamente. La aplicación ha de responder al navegador, normalmente, creando una página HTML dinámicamente para él, en la que se muestre la información pedida, usualmente dentro de un elemento específico para este fin, en una plantilla HTML.

Express posee métodos para especificar que función ha de ser llamada dependiendo del verbo HTTP usado en la petición (GET, POST, SET, etc.) y la estructura de la URL ("ruta"). También tiene los métodos para especificar que plantilla ("view") o gestor de visualización utilizar, donde están guardadas las plantillas de HTML que han de usarse y como generar la visualización adecuada para cada caso. El middleware de Express, puede usarse también para añadir funcionalidades para la gestión de cookies, sesiones y usuarios, mediante el uso de parámetros, en los métodos POST/GET. Puede utilizarse además cualquier sistema de trabajo con bases de datos, que sea soportado por Node (Express no especifica ningún método preferido para trabajar con bases de datos).

8. MongoDB



En la capa de datos, se utiliza la base de datos NoSQL MongoDB. Utiliza esquemas de datos llamados colecciones, donde se almacenan en formato BSON (Binary Javascript Object Notation).

8.1. Historia de MongoDB

El desarrollo de MongoDB se inició a principios de 2007 cuando la compañía estaba desarrollando una plataforma similar a Microsoft Azure como un servicio. Esta era una empresa con sede en Nueva York nombre 10gen que ahora se cambió su nombre a MongoDB Inc . El desarrollo inicial se centró en la creación de una PaaS (Plataforma como servicio), pero más tarde, en 2009, MongoDB llegó al mercado como un servidor de base de datos de código abierto y fue mantenido por esta organización.

En marzo de 2010, lanzó su primera versión “product ready”, que era la versión 1.4. La última versión, así como la versión estable de MongoDB, es la versión 4.0.10 que se lanzó el 31 de Mayo de 2019.

8.2. Características

8.2.1. Consultas Ad hoc

MongoDB admite búsquedas de campos, consultas de rango y expresiones regulares. Las consultas pueden devolver campos específicos de documentos y también incluir funciones de JavaScript definidas por el usuario. Las consultas también se pueden configurar para devolver una muestra aleatoria de resultados de un tamaño determinado.

8.2.2. Indexación

Los campos de un documento MongoDB se pueden indexar con índices primarios y secundarios.

8.2.3. Replicación

MongoDB proporciona alta disponibilidad con conjuntos de réplicas. Un conjunto de réplicas consta de dos o más copias de los datos. Cada miembro del conjunto de réplicas puede actuar en el rol de réplica primaria o secundaria en cualquier momento. Todas las escrituras y lecturas se realizan en la réplica

principal o secundaria en cualquier momento. Todas las escrituras y lecturas se realizan en la réplica principal de forma predeterminada. Las réplicas secundarias mantienen una copia de los datos del primario mediante la replicación incorporada. Cuando una réplica primaria falla, el conjunto de réplicas realiza automáticamente un proceso de elección para determinar qué secundaria debe convertirse en la primaria. Los secundarios pueden opcionalmente realizar operaciones de lectura, pero esos datos solo son finalmente consistentes de forma predeterminada.

8.2.4. Equilibrio de carga

MongoDB escala horizontalmente utilizando fragmentación. El usuario elige una clave de fragmento, que determina cómo se distribuirán los datos en una colección.

Los datos se dividen en rangos (según la clave del fragmento) y se distribuyen en múltiples fragmentos. Alternativamente, la clave del fragmento puede ser codificada para asignar un fragmento, lo que permite una distribución uniforme de los datos.

MongoDB puede ejecutarse en varios servidores, equilibrando la carga o duplicando datos para mantener el sistema en funcionamiento en caso de falla del hardware.

8.2.5. Agregación

MongoDB proporciona tres formas de realizar la agregación: el canal de agregación, la función de reducción de mapas y los métodos de agregación de un solo propósito.

Map-reduce se puede utilizar para el procesamiento por lotes de datos y las operaciones de agregación. Pero según la documentación de MongoDB, el Canal de agregación proporciona un mejor rendimiento para la mayoría de las operaciones de agregación.

8.2.6. Ejecución de JavaScript

JavaScript se puede usar en consultas, funciones de agregación (como MapReduce) y se puede enviar directamente a la base de datos para que se ejecute.

8.2.7. Colecciones limitadas

MongoDB admite colecciones de tamaño fijo denominadas colecciones con límite. Este tipo de colección mantiene el orden de inserción y, una vez alcanzado el tamaño especificado, se comporta como una cola circular.

8.3. ¿Por qué MongoDB?

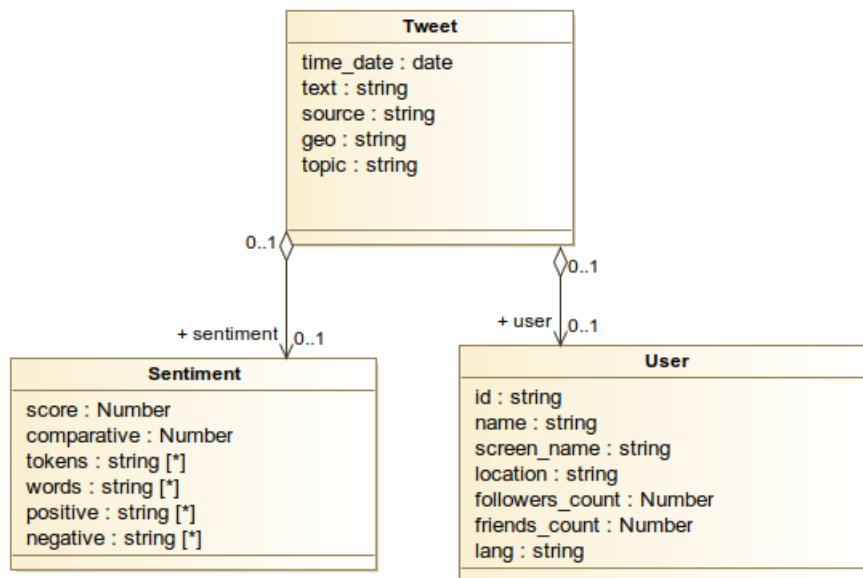
1. Los datos proporcionados por la API de Twitter son recibidos en formato JSON, mismo formato de almacenamiento de MongoDB. Esto permite que

la normalización de datos sea más sencilla.

2. La escalabilidad para manejar una alta cantidad de consultas en poco tiempo.
3. Flexibilidad y simplicidad en los esquemas de datos.

8.4. Esquema de datos de este proyecto

Cada tweet se encuentra almacenado como un documento en una colección llamada Tweet. El siguiente diagrama de clases UML se utiliza para representar la estructura correspondiente del documento JSON:



Siendo en la colección de Tweets:

- `time-date`: fecha y hora de emisión del tweet.
- `text`: texto del tweet. `source`: fuente del tweet (aplicación móvil, web, api, etc).
- `geo`: coordenadas de donde fue emitido el tweet si la tuviera.
- `topic`: tema de búsqueda, el mismo fue ingresado por el usuario al iniciar el análisis de sentimientos.

En la colección anidada Sentiment:

- `score`: suma de los valores de sentimiento de los tokens.
- `comparative`: score del tweet dividido en cantidad de tokens.

- tokens: array de todos los tokens del tweet.
- words: array de todas las palabras del tweet.
- positive: array de todas los tokens positivos.
- negative: array de todas los tokens negativos.

En la colección anidada User:

- id: id del usuario de twitter.
- name: nombre del usuario (identificador con @).
- screen_name: nombre del usuario a mostrar en su página de perfil.
- location: ubicación (ciudad y/o país) del usuario.
- followers_count: cantidad de seguidores.
- friends_count: cantidad de amigos.
- lang: lenguaje del usuario.

9. API de Twitter

Una interfaz de programación de aplicaciones (API) es un conjunto de definiciones de subrutinas, protocolos de comunicación y herramientas para crear software. En términos generales, es un conjunto de métodos de comunicación claramente definidos entre varios componentes. Una buena API hace que sea más fácil desarrollar un software al proporcionar todos los bloques de construcción, que luego son ensamblados por el programador .

Una API puede ser para un sistema basado en web, sistema operativo , sistema de base de datos, hardware de computadora o biblioteca de software .

En el caso de la API de Twitter, estamos ante la presencia de una API Web, en el cual las consultas son hechas a través de solicitudes GET o POST a la misma.

Entre las funciones que proporciona la API de Twitter para los desarrolladores tenemos:

1. Integrar Twitter a nuestra aplicación web.
2. Twitrear o retwitrear desde nuestra aplicación automáticamente.
3. Realizar campañas de publicidades.
4. Capturar tweets en tiempo real, o streaming.
5. Búsqueda de tweets históricos (limitado para usuarios gratuitos).

9.1. Objeto Tweet

Los tweets son el bloque de construcción atómico básico de todas las cosas de Twitter. Los tweets son también conocidos como “actualizaciones de estado.” El objeto Tweet tiene una larga lista de ‘nivel raíz’ atributos, incluidos los atributos fundamentales tales como id, created_at, y text. Los objetos de tweet son también el objeto "padre" de varios objetos hijos. Los objetos anidados incluyen user, entities y extended_entities. Los tweets que están etiquetados geográficamente tendrán un objeto place secundario.

9.1.1. Diccionario de datos del objeto Tweet

Atributo	Tipo
created_at	String
id	Int64
id_str	String
text	String
source	String
truncated	Boolean
in_reply_to_status_id	Int64
in_reply_to_status_id_str	String
in_reply_to_user_id	Int64
in_reply_to_user_id_str	String
in_reply_to_screen_name	String
user	User object
coordinates	Coordinates
coordinates	Places
quoted_status_id	Int64
quoted_status_id_str	String
is_quote_status	Boolean
quoted_status	Tweet
retweeted_status	Tweet
quote_count	Integer
reply_count	Int
retweet_count	Int
favorite_count	Integer
entities	Entities
extended_entities	Extended Entities
favorited	Boolean
retweeted	Boolean
possibly_sensitive	Boolean
filter_level	String
lang	String
matching_rules	Array of Rule Objects

10. Sentiment

Sentiment es un paquete de software alojado en NPM que permite el análisis de sentimientos de cadena de textos utilizando el listado de palabras AFINN-165 y el Emoji World Sentiment Ranking.

Entre las características principales que lo destacan es:

1. Licencia MIT.
2. Análisis de sentimiento en inglés.
3. Posibilidad de agregar nuevos lenguajes.
4. Opciones de benchmark.

10.1. Funcionamiento

A la cadena de texto a analizar se le aplica un proceso de tokenización. Esto significa dividir el conjunto de caracteres en “tokens”, es decir, subcadenas de texto formadas a partir de separar la cadena en los espacios, y eliminando los caracteres no alfanuméricos.

Cada uno de los tokens obtenidos es comparado con su valor en el diccionario AFFIN-165 y el Emoji World Sentiment Ranking. En base a ello, cada token obtiene un valor de sentimiento, que es un valor numérico entre -5 y 5. La suma de los valores de sentimientos corresponden al puntaje S:

$$S = \sum_1^n t_1 + t_2 + \dots + t_n$$

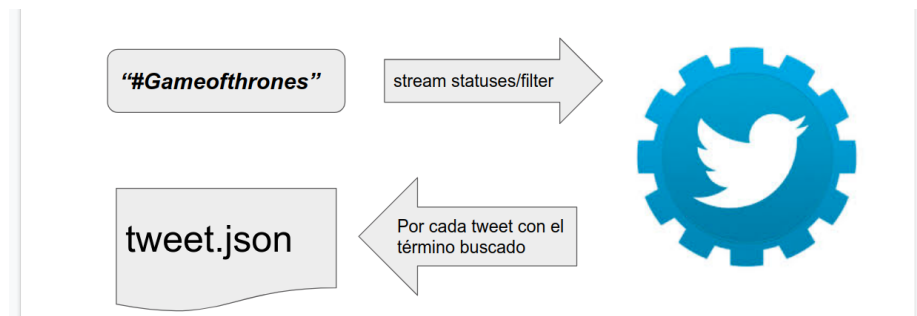
Y siendo el valor comparativo respecto a la cantidad de tokens:

$$C = \sum_1^n \frac{t_1 + t_2 + \dots + t_n}{n}$$

Parte IV. Prueba de Sentball

11. #GameOfThrones

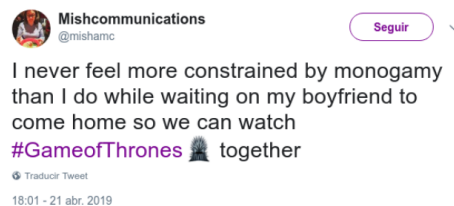
El día 21 de Abril del año 2019 se realizó la captura y análisis de sentimientos de los tweets enviados en tiempo real durante la emisión del segundo capítulo de la octava temporada de la popular serie de televisión “Game of Thrones”. Para ello, se utilizó Sentball para capturar aquellos tweets con el hashtag “#gameofthrones”.



Sentball envía una petición POST por medio de la API de Twitter del tipo “stream statuses/filter” con el hashtag “#Gameofthrones”.

Esto abre un canal de streaming, donde por cada tweet que coincida con el término indicado es transmitido en tiempo real a Sentball en formato json. Estos datos de acuerdo a la documentación de Twitter, es llamado el “Objeto Tweet”, que contiene todas los datos almacenados de dicho tweet, como sería el texto, autor del mismo, datos de geolocalización, plataforma desde donde fue enviado y más.

Por ejemplo, uno de los Tweets capturados fue el siguiente:



“Nunca me sentí más atada por la monogamia que cuando espero que mi novio regrese a casa así podemos ver #GameofThrones juntos”

Uno podría pensar que es difícil obtener datos estructurados a partir de algo tan cotidiano como un Tweet de una persona que espera su pareja para ver su serie favorita, sin embargo Sent Ball procesa este tweet con Sentiment tokenizado y calculando su valor de sentimiento.

Esto puede ser visto en la siguiente captura de lo almacenado en la base de datos:

```
▼ tokens : Array
  0 : "i "
  1 : "never "
  2 : "feel "
  3 : "more "
  4 : "constrained "
  5 : "by "
  6 : "monogamy "
  7 : "than "
  8 : "i "
  9 : "do "
  10 : "while "
  11 : "waiting "
  12 : "on "
  13 : "my "
  14 : "boyfriend "
  15 : "to "
  16 : "come "
  17 : "home "
  18 : "so "
  19 : "we "
  20 : "can "
  21 : "watch "
  22 : "gameofthrones "
  23 : "together "
```

Según el diccionario AFFIN-165, el único token que corresponde a una palabra de sentimiento no neutral es “constrained” (atada), con un puntaje de -2.

Esto nos da un puntaje de -2, y un valor comparativo de -0,09.

Siguiendo el esquema de datos de la colección Tweet, los valores son almacenados en nuestra base de datos de MongoDB.

11.1. Resultados

Durante la hora de duración del episodio se han capturado 58020 tweets con el hashtag #Gameofthrones.

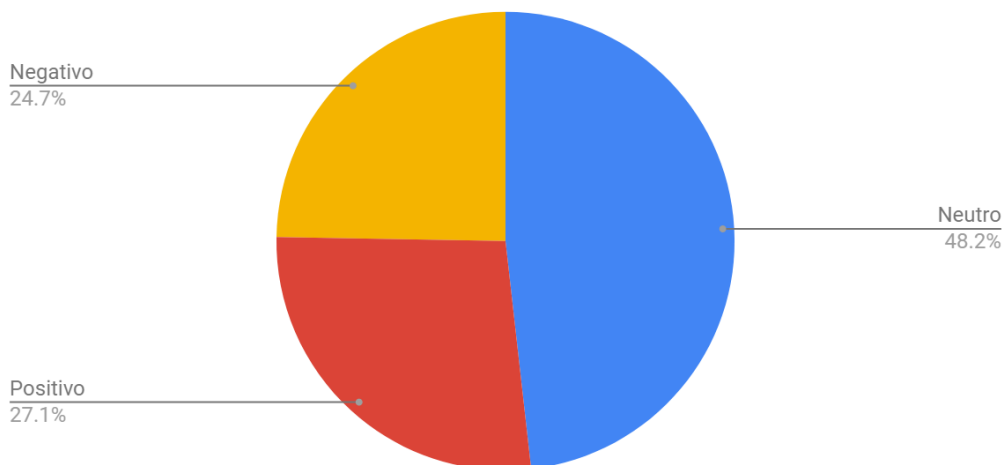
De los cuales:

- 27972 tienen un valor de sentimientos neutro (igual a 0).
- 30048 tienen un valor de sentimientos no neutro (distinto de 0).
 - 15710 tienen un valor de sentimientos positivo.
 - 14338 tienen un valor de sentimientos negativo.

En un diagrama de torta, esto puede ser graficado por porcentajes de la siguiente manera:

Distribución de sentimientos del hashtag #gameofthrones en Twitter

Sentball, 21/04/2019 entre las 22hs y 23hs. Elaboración propia.

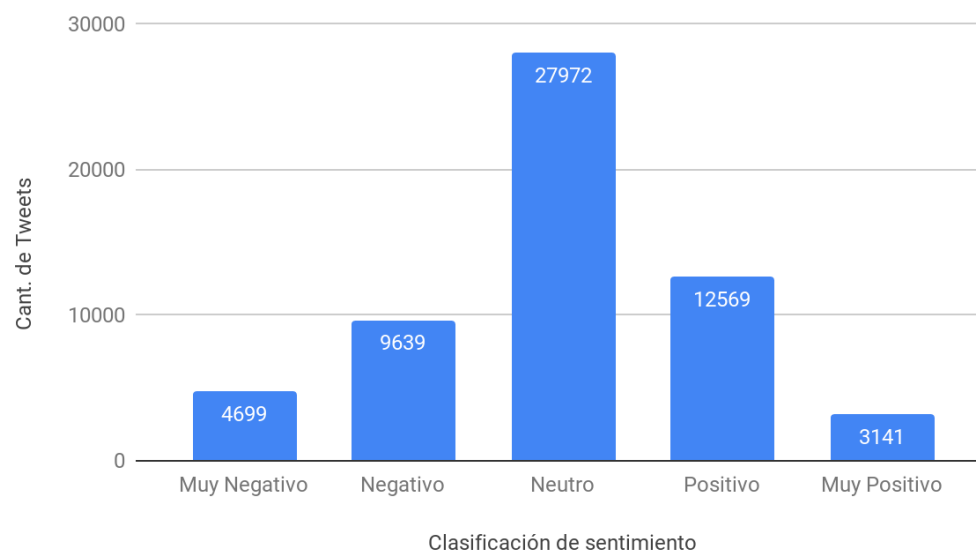


Tomando los intervalos para clasificar de mejor manera los sentimientos siendo:

1. Muy negativos entre -5 y -3.
2. Negativos entre -3 y 0.
3. Neutro 0.
4. Positivo entre 0 y 3.
5. Muy positivos entre 3 y 5.

Distribución de sentimientos del hashtag #gameofthrones en Twitter

Sentball, 21/04/2019 entre las 22hs y 23hs. Elaboración propia.



Podemos apreciar que la distribución de sentimientos corresponde a lo que uno esperaría, siendo los extremos poco frecuentes, y elevando la misma en los valores neutros.

Parte V. Conclusión

12. Conclusión

De este trabajo se puede concluir que:

1. El análisis de sentimientos es una herramienta sumamente potente con gran panorama a futuro.
2. SentBall puede cumplimentar los objetivos trazados inicialmente, que eran la captura de Tweets en tiempo real y su correspondiente análisis de sentimiento.

13. Evolución a futuro

1. Mejora del diccionario de español utilizando expresiones regulares para reconocer patrones como por ejemplo palabras con género.
2. Mejora de la interfaz de usuario.

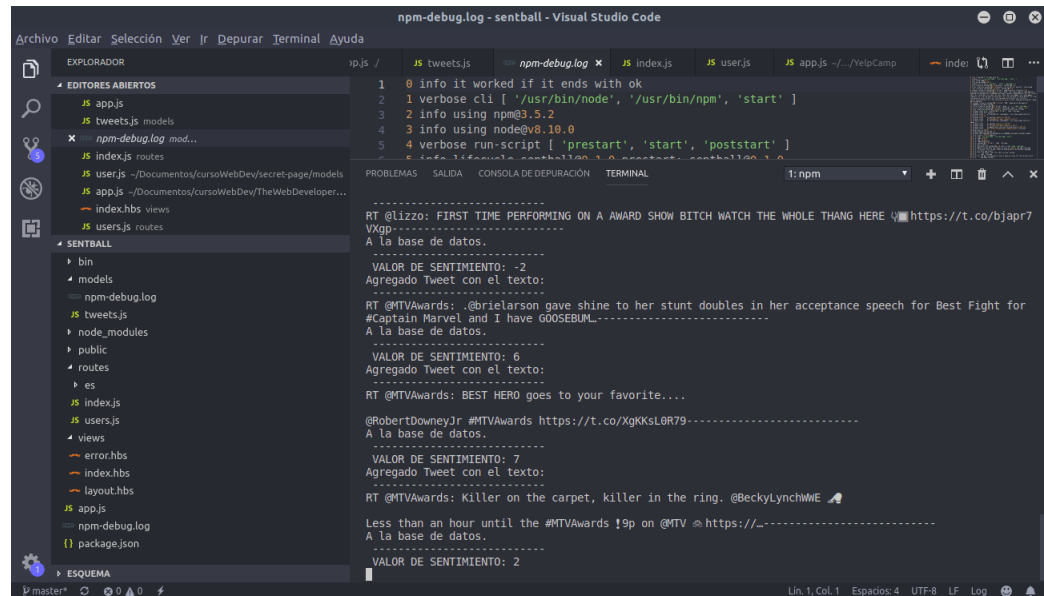
Parte VI. Bibliografía

1. IBM Big Data & Analytics Hub. <http://www.ibmbigdatahub.com/infographic/four-vs-big-data>
2. Bo Pang and Lillian Lee. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval* 2 (2008), no. 1-2, 1–135.
3. Fawcett, Robin P. y Gordon H. tucker, 1990: “Demonstration of GenesYs: a very large, semantically based systemic functional grammar” en *Proceedings of the 13th International Conference on Computational Linguistics*, Helsinki, 47-49.
4. Ferrari, Giacomo, 2004: “state of the art in computational linguistics” en Piet van sterkenBurg (ed.): *Linguistics today: facing a greater challenge*, amsterdam-Filadelfia: John Benjamins, 163-186.
5. Fillmore, charles J., 1968: “The case for case” en emmon W. Bach y Robert T. harms (ed.): *Universals in linguistic theory*, nueva York: Holt, Rinehart and Winston, 1-88.
6. <https://developer.twitter.com/en/docs.html>
7. <https://docs.npmjs.com/>
8. <https://www.npmjs.com/package/sentiment>
9. <https://www.npmjs.com/package/twitter>
10. <https://docs.mongodb.com/>
11. <https://mongoosejs.com/docs/> <https://expressjs.com/en/4x/api.html>

Parte VII. ANEXO

Capturas de pantalla de Sentball

Sentball capturando y almacenando tweets analizados en la base de datos.



```
npm-debug.log - sentball - Visual Studio Code
Archivo Editar Selección Ver Ir Depurar Terminal Ayuda

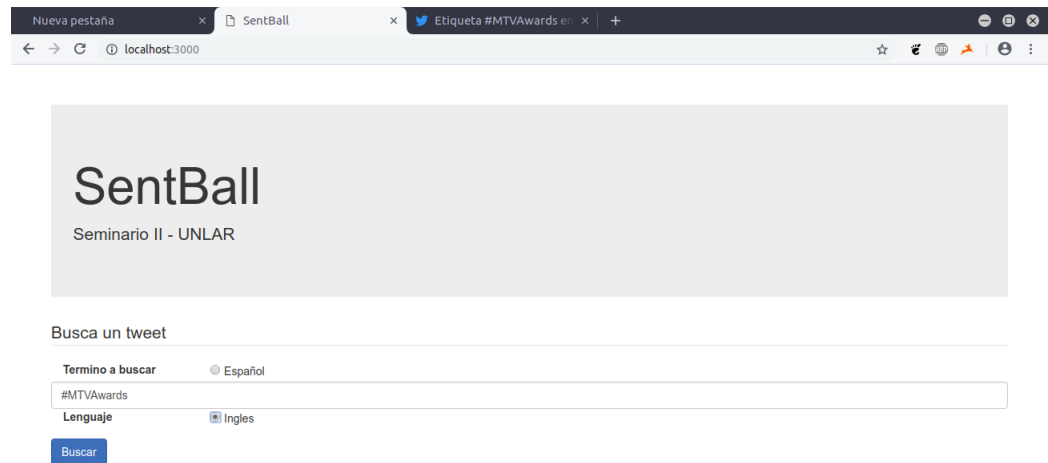
EXPLORADOR
  EDITORES ABIERTOS
    JS app.js
    JS tweets.js models
    X npm-debug.log mod...
    JS index.js routes
    JS user.js ~/Documentos/cursWebDev/secret-page/models
    JS app.js ~/Documentos/cursWebDev/TheWebDeveloper...
    index.hbs views
    JS users.js routes
  SENTBALL
    bin
    models
    npm-debug.log
    JS tweets.js
    node_modules
    public
    routes
    es
    index.js
    JS users.js
    views
    error.hbs
    index.hbs
    layout.hbs
    JS app.js
    npm-debug.log
    package.json

  ESQUEMA

1 0 info it worked if it ends with ok
2 1 verbose cli [ '/usr/bin/node', '/usr/bin/npm', 'start' ]
3 2 info using npm@3.5.2
4 3 info using node@v8.10.0
5 4 verbose run-script [ 'prestart', 'start', 'poststart' ]
6 5 info Lifecycle script 'start' has been successfully launched

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL 1: npm
-----
RT @lizzo: FIRST TIME PERFORMING ON A AWARD SHOW BITCH WATCH THE WHOLE THANG HERE 🍷 https://t.co/bjapr7
VXgp-----
A la base de datos.
-----
VALOR DE SENTIMIENTO: -2
Agregado Tweet con el texto:
-----
RT @MTVAwards: @briarlson gave shine to her stunt doubles in her acceptance speech for Best Fight for
#Captain Marvel and I have GOOSEBUM-----
A la base de datos.
-----
VALOR DE SENTIMIENTO: 6
Agregado Tweet con el texto:
-----
RT @MTVAwards: BEST HERO goes to your favorite....
-----
@RobertDowneyJr #MTVAwards https://t.co/XgKksLOR79-----
A la base de datos.
-----
VALOR DE SENTIMIENTO: 7
Agregado Tweet con el texto:
-----
RT @MTVAwards: Killer on the carpet, killer in the ring. @BeckyLynchWWE 🍷
-----
Less than an hour until the #MTVAwards !9p on @MTV 🍷 https://------
A la base de datos.
-----
VALOR DE SENTIMIENTO: 2
```

Interfaz en front-end de Sentball al momento de buscar #gameofthrones.



The screenshot shows a web browser window with three tabs: 'Nueva pestaña', 'SentBall', and 'Etiqueta #MTVAwards en...'. The address bar shows 'localhost:3000'. The main content area has a light gray header with the text 'SentBall' and 'Seminario II - UNLAR'. Below this is a search section titled 'Busca un tweet'. It includes a 'Termino a buscar' dropdown menu set to 'Español', a text input field containing '#MTVAwards', a 'Lenguaje' dropdown menu set to 'Inglés', and a blue 'Buscar' button.

Nueva pestaña x SentBall x Etiqueta #MTVAwards en x +

← → ↻ 📄 localhost:3000 ☆ 🌐 🚦 📱 ⋮

SentBall

Seminario II - UNLAR

Busca un tweet

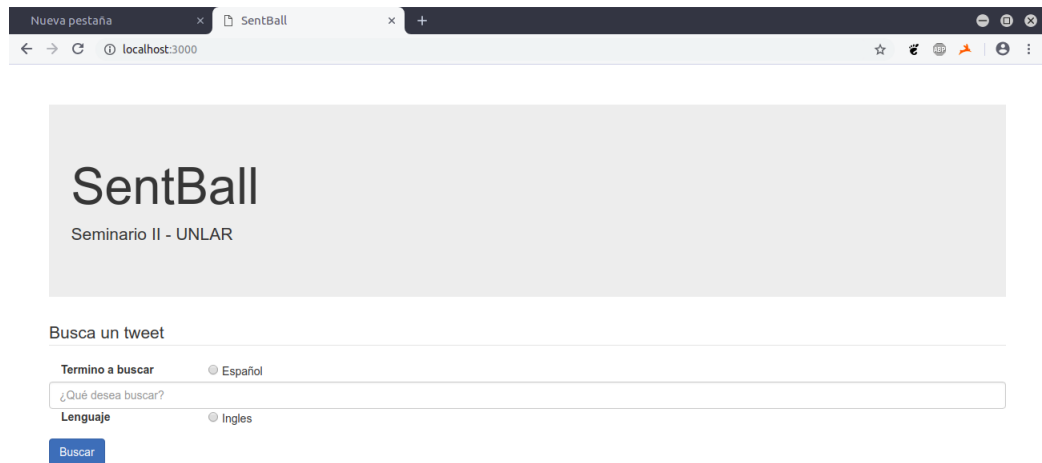
Termino a buscar ☐ Español

#MTVAwards

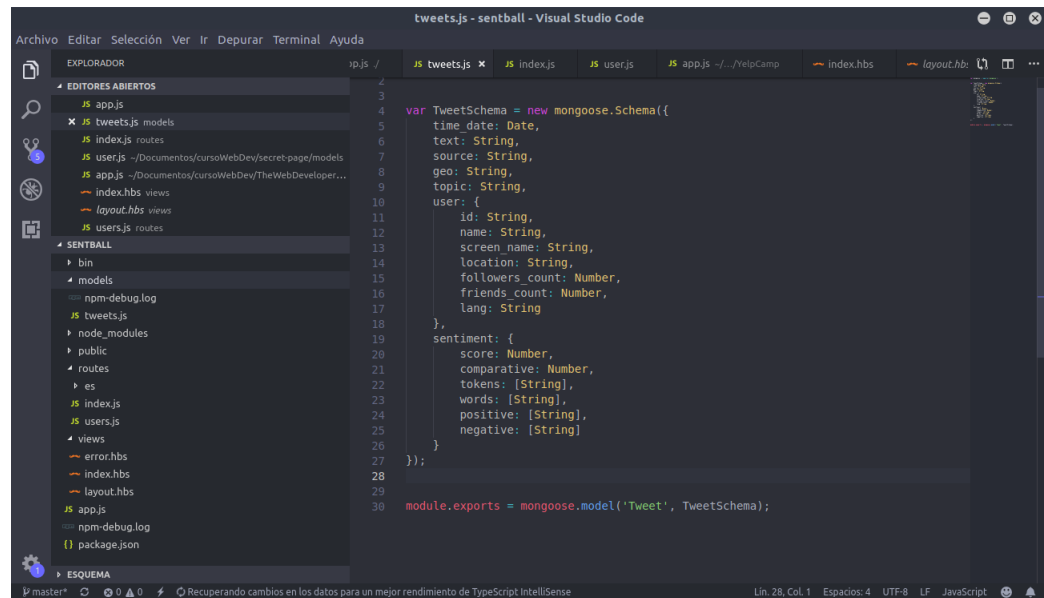
Lenguaje ☒ Inglés

Buscar

Interfaz de front-end por defecto.



Esquema de datos de Tweet. Captura del código.



The screenshot shows the Visual Studio Code editor with the file `tweets.js` open. The left sidebar displays the Explorer view with the project structure. The main editor area shows the following code:

```
var TweetSchema = new mongoose.Schema({
  time_date: Date,
  text: String,
  source: String,
  geo: String,
  topic: String,
  user: {
    id: String,
    name: String,
    screen_name: String,
    location: String,
    followers_count: Number,
    friends_count: Number,
    lang: String
  },
  sentiment: {
    score: Number,
    comparative: Number,
    tokens: [String],
    words: [String],
    positive: [String],
    negative: [String]
  }
});

module.exports = mongoose.model('Tweet', TweetSchema);
```

The status bar at the bottom indicates the current line and column: `Lin. 28, Col. 1`. The file encoding is `UTF-8` and the line ending is `LF`. The language is `JavaScript`.

Filtrado de tweets con el hashtag #gameofthrones que tengan un score mayor a 3.

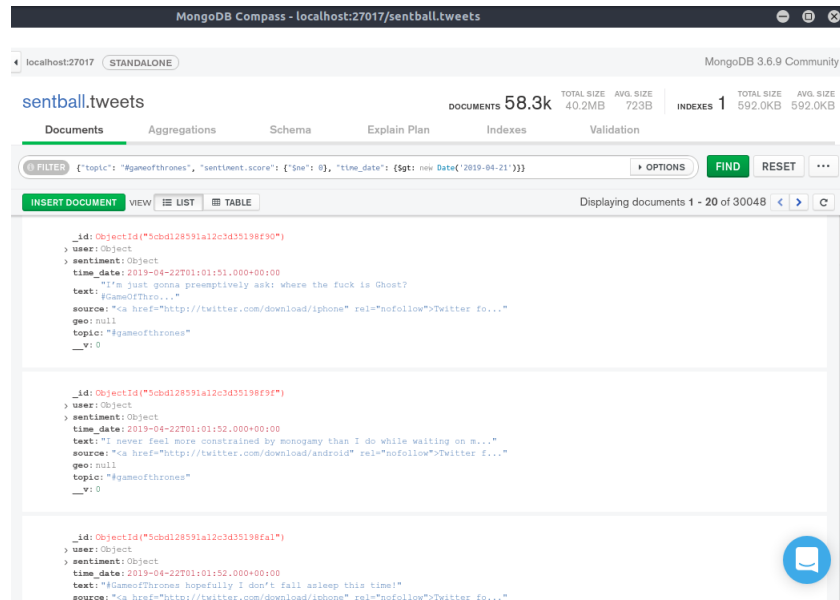
The screenshot shows the MongoDB Compass interface for a database named 'sentball.tweets'. The filter applied is: `{ "topic": "#gameofthrones", "sentiment.score": { "$gt": 3 }, "time_date": { "$gt": new Date('2019-04-21') } }`. The results are displayed in a list view, showing three documents. Each document contains fields for _id, user, sentiment, time_date, text, source, geo, and topic. The first document has a text about a comedy show, the second is a retweet about OriginalPunko, and the third is another retweet about OriginalPunko.

```
{
  "_id": ObjectId("5c6d128991a12c3d3519906b"),
  "user": Object,
  "sentiment": Object,
  "time_date": 2019-04-22T01:01:56.000+00:00,
  "text": "Gimme real comedy for this episode's title tonight nice one @hbo #Gameo...",
  "source": "ca href='\"http://twitter.com\"' rel='\"nofollow\">Twitter Web Client/a",
  "geo": null,
  "topic": "#gameofthrones",
  "__v": 0
}
```

```
{
  "_id": ObjectId("5c6d128b91a12c3d3519909c"),
  "user": Object,
  "sentiment": Object,
  "time_date": 2019-04-22T01:01:57.000+00:00,
  "text": "RT @OriginalPunko: RT &amp; Follow @OriginalPunko for a chance to win ...",
  "source": "ca href='\"http://twitter.com/download/iphone\"' rel='\"nofollow\">Twitter fo...",
  "geo": null,
  "topic": "#gameofthrones",
  "__v": 0
}
```

```
{
  "_id": ObjectId("5c6d128b91a12c3d351990b6"),
  "user": Object,
  "sentiment": Object,
  "time_date": 2019-04-22T01:01:58.000+00:00,
  "text": "RT @OriginalPunko: RT &amp; Follow @OriginalPunko for a chance to win ...",
  "source": "ca href='\"http://twitter.com/download/iphone\"' rel='\"nofollow\">Twitter fo...",
  "geo": null,
  "topic": "#gameofthrones",
  "__v": 0
}
```

Filtrado de tweets con el hashtag #gameofthrones que tengan un score no igual a 0.

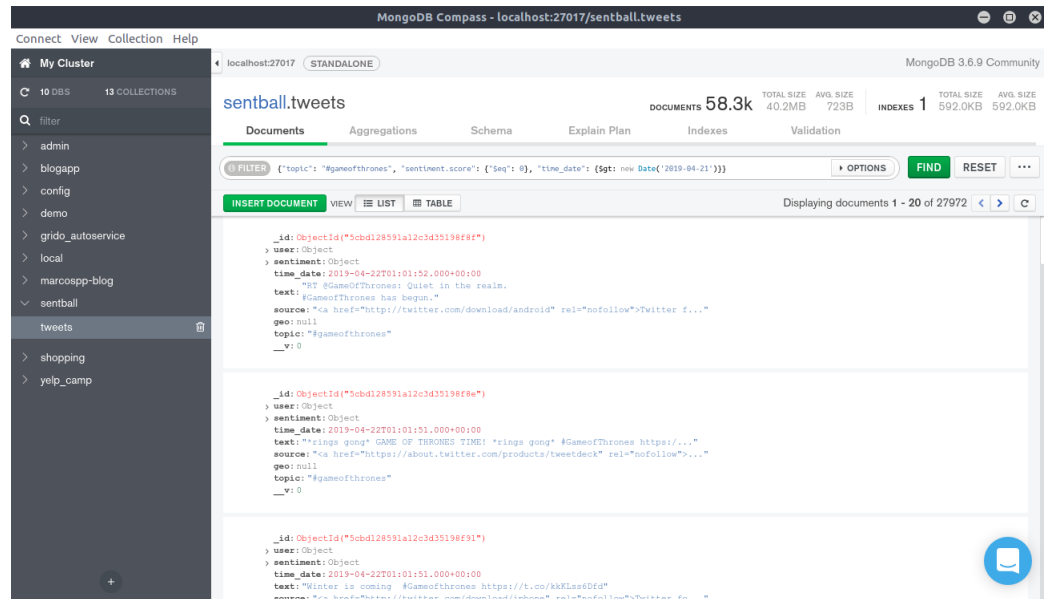


Filtrado de tweets con el hashtag #gameofthrones que tengan un score menor a -3.

The screenshot shows the MongoDB Compass interface for a database named 'sentball.tweets'. The filter applied is: `{ "topic": "#gameofthrones", "sentiment.score": { "$lt": -3 }, "time_date": { "$gt": new Date('2019-04-21') } }`. The results are displayed in a table view, showing three documents. Each document contains fields for _id, user, sentiment, time_date, text, source, geo, topic, and __v.

Document	_id	user	sentiment	time_date	text	source	geo	topic	__v
1	ObjectId("5c0d128591a12c3d35198f90")	Object	Object	2019-04-22T01:01:51.000+00:00	"I'm just gonna preemptively ask: where the fuck is Ghost?"	"ca href=http://twitter.com/download/iphone rel=nofollow">Twitter fo...	null	#gameofthrones	0
2	ObjectId("5c0d128891a12c3d35199020")	Object	Object	2019-04-22T01:01:55.000+00:00	"Go time bitches The True Born King Aegon T bout to lead us into battle..."	"ca href=http://twitter.com/download/iphone rel=nofollow">Twitter fo...	null	#gameofthrones	0
3	ObjectId("5c0d128891a12c3d35199021")	Object	Object	2019-04-22T01:01:55.000+00:00	"WE BACK BITCHES #GameofThrones"	"ca href=http://twitter.com/download/android rel=nofollow">Twitter f...	null	#gameofthrones	0

Filtrado de tweets con el hashtag #gameofthrones que tengan un score igual a 0.



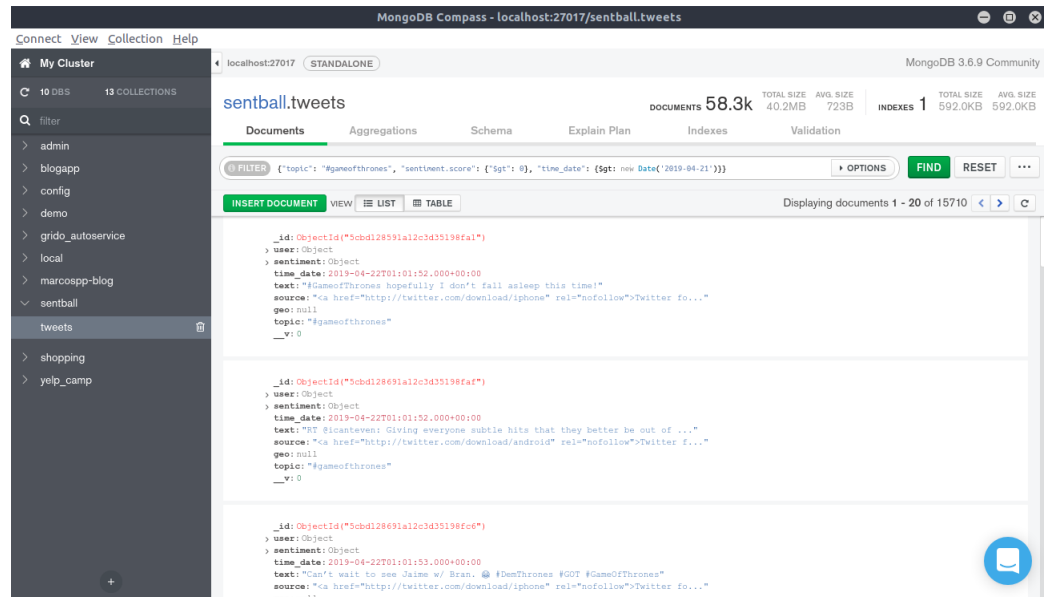
The screenshot shows the MongoDB Compass interface for a cluster named 'sentball.tweets'. The left sidebar lists collections, with 'tweets' selected. The main panel displays a list of documents filtered by the query: `{ "topic": "#gameofthrones", "sentiment.score": { "$eq": 0 }, "time_date": { "$gt": new Date('2019-04-21') } }`. The interface shows 58.3k documents in total. The first three documents are displayed, each with a red ID and a sentiment score of 0. The first document's text is: "It @GameOfThrones: Quiet in the realm. #GameOfThrones has begun." The second document's text is: "rings gong! GAME OF THRONES TIME! rings gong! #GameOfThrones https://...". The third document's text is: "Winter is coming #GameOfThrones https://t.co/kKLasDfd".

```
{
  "_id": ObjectId("5cdd128591a12c3d3d35198f9f"),
  "user": Object,
  "sentiment": Object,
  "time_date": 2019-04-22T01:01:52.000+00:00,
  "text": "It @GameOfThrones: Quiet in the realm. #GameOfThrones has begun.",
  "source": "Ca href='http://twitter.com/download/android' rel='nofollow'>Twitter f...",
  "geo": null,
  "topic": "#gameofthrones",
  "_v": 0
}
```

```
{
  "_id": ObjectId("5cdd128591a12c3d3d35198f9e"),
  "user": Object,
  "sentiment": Object,
  "time_date": 2019-04-22T01:01:51.000+00:00,
  "text": "rings gong! GAME OF THRONES TIME! rings gong! #GameOfThrones https://...",
  "source": "Ca href='https://about.twitter.com/products/tweetdeck' rel='nofollow'>...",
  "geo": null,
  "topic": "#gameofthrones",
  "_v": 0
}
```

```
{
  "_id": ObjectId("5cdd128591a12c3d3d35198f91"),
  "user": Object,
  "sentiment": Object,
  "time_date": 2019-04-22T01:01:51.000+00:00,
  "text": "Winter is coming #GameOfThrones https://t.co/kKLasDfd",
  "source": "Ca href='http://twitter.com/download/iphone' rel='nofollow'>Twitter fo...",
  "geo": null,
  "topic": "#gameofthrones",
  "_v": 0
}
```

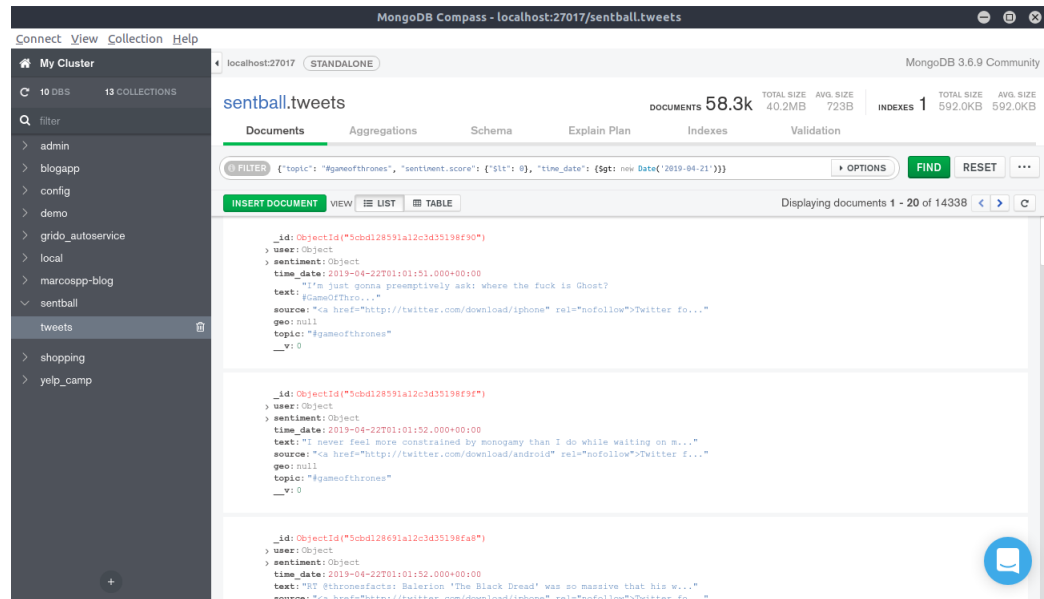
Filtrado de tweets con el hashtag #gameofthrones que tengan un score mayor a 0.



The screenshot shows the MongoDB Compass interface for a cluster named 'sentball.tweets' on localhost:27017. The left sidebar lists 13 collections, with 'tweets' selected. The main panel displays a list of documents filtered by the query: `{ "topic": "#gameofthrones", "sentiment.score": { "$gt": 0 }, "time_date": { "$gt": new Date('2019-04-21') } }`. The interface shows 58.3k documents in total. The first three documents are displayed, each containing fields for _id, user, sentiment, time_date, text, source, geo, topic, and __v. The first document has a sentiment score of 0, while the second and third have scores of 1 and 2 respectively. The third document's text mentions 'Can't wait to see Jaime w/ Bran' and 'DenThrones #DOT #GameOfThrones'.

Document ID	user	sentiment	time_date	text	source	geo	topic	__v
ObjectId("5c6d128591a12c3d35198fa1")	Object	Object	2019-04-22T01:01:52.000+00:00	"#gameofthrones hopefully i don't fall asleep this time!"	"ca href=http://twitter.com/download/iphone rel=nofollow">Twitter fo..."	null	"#gameofthrones"	0
ObjectId("5c6d128691a12c3d35198fa2")	Object	Object	2019-04-22T01:01:52.000+00:00	"ET @casteven: Giving everyone subtle hits that they better be out of ..."	"ca href=http://twitter.com/download/android rel=nofollow">Twitter f..."	null	"#gameofthrones"	1
ObjectId("5c6d128691a12c3d35198fa3")	Object	Object	2019-04-22T01:01:53.000+00:00	"Can't wait to see Jaime w/ Bran. @ #DenThrones #DOT #GameOfThrones"	"ca href=http://twitter.com/download/iphone rel=nofollow">Twitter fo..."	null	"#gameofthrones"	2

Filtrado de tweets con el hashtag #gameofthrones que tengan un score menor a 0.



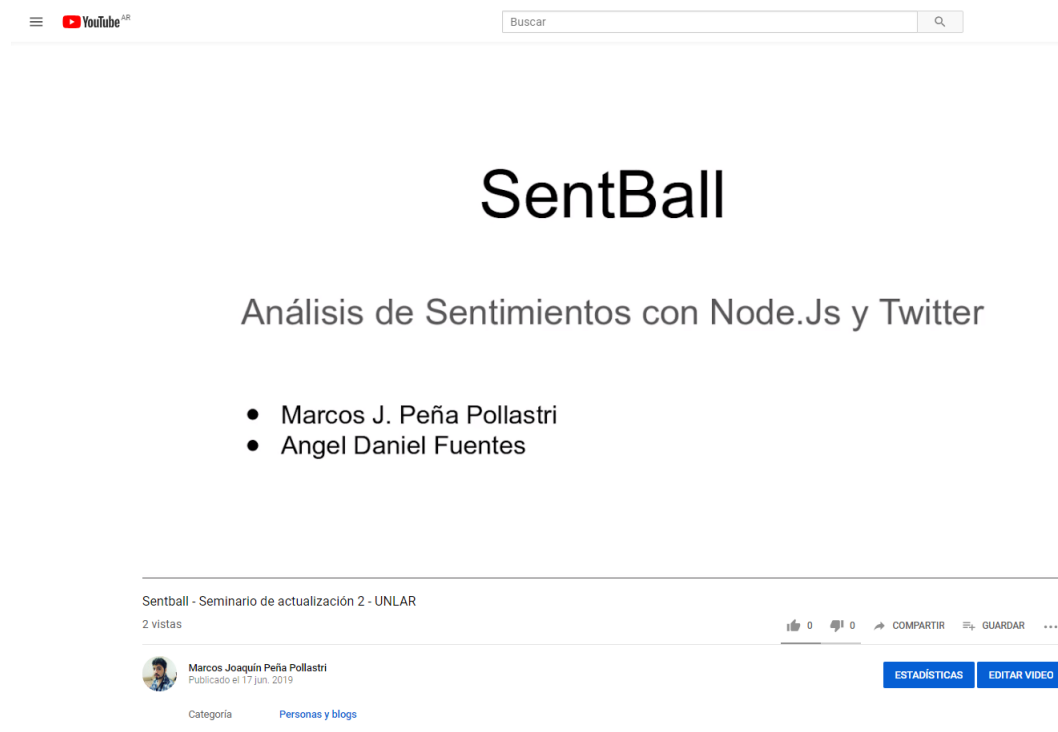
The screenshot shows the MongoDB Compass interface for a cluster named 'sentball.tweets'. The left sidebar lists collections, with 'tweets' selected. The main panel displays a list of documents filtered by the query: `{ "topic": "#gameofthrones", "sentiment.score": { "$lt": 0 }, "time_date": { "$gt": new Date('2019-04-21') } }`. The results show three documents, each with a negative sentiment score and a text field containing a tweet about Game of Thrones. The first document has a sentiment score of -0.1, the second -0.2, and the third -0.3. The text fields contain tweets from users like 'I'm just gonna preemptively ask: where the fuck is Ghost?' and 'I never feel more constrained by monogamy than I do while waiting on M...'. The interface also shows document statistics: 58.3k documents, 40.2MB total size, and 723B average size.

Document	sentiment.score	text
1	-0.1	"I'm just gonna preemptively ask: where the fuck is Ghost?"
2	-0.2	"I never feel more constrained by monogamy than I do while waiting on M..."
3	-0.3	"I never feel more constrained by monogamy than I do while waiting on M..."

Video de presentación en línea

El video de la presentación de Sentball se encuentra disponible en línea en la plataforma de YouTube. Para acceder debe ingresar al siguiente link o escaneando el código QR a continuación:

<https://www.youtube.com/watch?v=AesZUVBbGb4>



The screenshot shows a YouTube video player interface. At the top, there is a search bar with the text 'Buscar' and a magnifying glass icon. Below the search bar, the video title 'SentBall' is displayed in a large, bold font. Underneath the title, the subtitle 'Análisis de Sentimientos con Node.Js y Twitter' is shown. A list of authors is provided: Marcos J. Peña Pollastri and Angel Daniel Fuentes. Below the authors, the video title 'Sentball - Seminario de actualización 2 - UNLAR' is displayed, followed by '2 vistas'. To the right of the video title, there are icons for likes (0), dislikes (0), and a share icon. Below the video title, there is a profile picture of Marcos Joaquín Peña Pollastri, followed by his name and the text 'Publicado el 17 jun. 2019'. To the right of the profile picture, there are two buttons: 'ESTADÍSTICAS' and 'EDITAR VIDEO'. Below the profile picture, the text 'Categoría' is followed by a link to 'Personas y blogs'.