Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

Trabajo de Fin de Grado

Grado en Ingeniería Informática

Automatización de respuestas lógicas a tweets mediante PLN

Automation of logical replies to tweets using NLP

Noel Padrón Díaz

La Laguna, 11 de septiembre de 2020

D. **José Luis González Ávila**, con N.I.F. 78.677.390-W profesor Titular de Universidad adscrito al Departamento de Informática de la Universidad de La Laguna, como tutor

C E R T I F I C A (N)

Que la presente memoria titulada:

"Automatización de respuestas lógicas a tweets mediante PLN"

ha sido realizada bajo su dirección por D. **Noel Padrón Díaz**,

con N.I.F. 78.645.284-G.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 11 de septiembre de 2020

Agradecimientos

En primer lugar agradecer a mi tutor José Luis González Ávila, porque no ha sido un año fácil tanto en lo laboral como en lo personal y me ha dado todo tipo de facilidades para sacar el proyecto adelante.

A mis compañeros de facultad, por la ayuda prestada y el apoyo en todo momento.

Y por último a mi familia y a mis amigos, que literalmente me han ayudado en todo lo que he necesitado para sacar esto adelante.

Licencia

Resumen

Twitter es una plataforma social, un servicio de comunicación bidireccional con el que puedes compartir información de diverso tipo de una forma rápida, sencilla y gratuita. Posee una API REST que permite acceder a leer y escribir datos de la misma aplicación, una herramienta común en las redes sociales actuales.

Muchas empresas (periódicos, webs de divulgación, tiendas online, etc.) usan esta API para generar automáticamente tweets anunciando sus artículos, productos, etc. No es la idea de nuestro trabajo, pero es similar.

El Procesamiento del Lenguaje Natural es el campo de conocimiento de la Inteligencia Artificial que se ocupa de investigar la manera de comunicar las máquinas con las personas mediante el uso de lenguas naturales, como el español, el inglés o el chino. (Moreno, 2017)

Las posibilidades que nos ofrece esta tecnología son innumerables y, en nuestro caso, nos permitirá comunicar los tweets con la base de datos.

La idea del trabajo es facilitar la divulgación de noticias e imágenes, obteniendo y analizando los tweets de nuestra TimeLine. Esto se conseguiría extrayendo palabras clave que nos permitan clasificarlos para poder responderlos.

Palabras clave: *Twitter, Tweet, TimeLine, FreeLing, Procesamiento del Lenguaje Natural, Base de Datos, Noticias, Imágenes.*

Abstract

Twitter is a social platform, a bidirectional communication service which you can rapidly share any kind of information through easily and freely. It owns an API REST that allows to access to read and write data from the same application, a common tool on current social networks.

Many firms (diaries, outreach websites, online shops, etc.) use this API to automatically generate tweets advertising their articles, products, etc. It is not the idea of our project, but it is similar.

The Natural Language Processing is the field of knowledge of the Artificial Intelligence which takes care of investigating the way to communicate machines with people using natural languages, like Spanish, English or Chinese. (Moreno, 2017)

The possibilities this technology offers us are endless and, in our case, they will allow us to communicate the tweets with the database.

The project idea is to facilitate the news and images divulgation, obtaining and analyzing our TimeLine tweets, extracting keywords that allow us to classify them in order to answer them.

Keywords: *Twitter, Tweet, TimeLine, FreeLing, Natural Language Processing, Data Base, News, Images.*

Índice general

Índice de figuras

Índice de tablas

Capítulo 1
Introducción

1.1 Motivación

En los últimos tiempos las redes sociales han dejado de ser simplemente un lugar en el que las personas comparten su vida personal con amistades. Según Amanda Adame (2019):

De acuerdo con el reporte anual The Global State of Digital in 2019 creado por Hootsuite y We Are Social, el 52% de la población mundial utiliza redes sociales. Esta enorme audiencia global que utiliza estos canales representa un mercado vasto de oportunidad para cualquier empresa, sin importar su tamaño.

Aunque nuestra aplicación tiene bastante en común con un bot, se podría decir que es una versión mejorada de estos. Un "bot 2.0", donde no solo se contesta en función de las palabras que se encuentren en un tweet, sino que analiza el mensaje para tener una mayor precisión para buscar la mejor respuesta.

¿Por qué usar el procesamiento del lenguaje natural? La sintaxis y la ortografía no es el principal foco de atención del usuario medio de las redes sociales. Las prisas, la falta de interés o el desconocimiento provocan que cada vez se escriba con más faltas ortográficas, lo que supone un problema a la hora de enviar respuestas en base al texto.

Para ello usamos la herramienta FreeLing, una potente librería de código abierto para el procesamiento multilingüe, que nos permitirá analizar los mensajes para tener un mayor porcentaje de acierto en nuestras respuestas.

1.2 Objetivo

El objetivo de este proyecto es desarrollar una herramienta para las respuestas automáticas a tweets en la red social Twitter, con un conjunto de librerías de datos con textos, fotos, enlaces, noticias y perfiles de usuarios.

La función de la herramienta es captar el mensaje cuando se publique un tweet, analizarlo con un procesador de lenguaje (FreeLing) y responder con un conjunto de respuestas cercanas al tweet.

La aplicación será desarrollada con el lenguaje Java, usando la librería para el procesado de lenguajes FreeLing, y contando con una base de datos no relaciona MongoDB, por lo que podemos dividir el proyecto en tres partes: el control de la API de Twitter, la integración de FreeLing y el uso de la base de datos MongoDB.

Capítulo 2
Estado del arte

2.1 Twitter

Twitter es una red social que funciona como plataforma de comunicación bidireccional, que limita sus publicaciones a 280 caracteres. Inicialmente, cuando fue abierta en 2006 por su fundador Jack Dorsey, los llamados "tweets" estaban limitados a 140 caracteres, pero en el año 2017 fue cuando se dobló el tamaño de los caracteres para incentivar el uso de Twitter de manera más activa.

También es importante hablar de los bots en Twitter.

Hay diferentes clases de bots en Twitter. Algunos son usuarios completamente automatizados, que publican mensajes definidos previamente, y detrás de los cuales jamás hay una persona redactando tweets. Estos bots son generalmente los peor vistos, y los que hartan a los usuarios. Pero en algunos casos, se trata de bots «simpáticos» que están expresamente identificados como bots y tienen la función de hacer alguna broma de oportunidad, o lanzar alguna frase cómplice, relacionada con algo que acabamos de escribir en Twitter. (Quaglia, 2012)

Twitter permite la existencia de bots mientras estos no tengan comportamientos abusivos. En estos casos la aplicación puede llegar a suspender la cuenta e incluso cerrarla definitivamente. Se ha lanzado hace poco una red de bots, que calcula que pueden haber cientos de miles en esta red social. Twitter también pone a disposición de sus usuarios unas reglas y mejores prácticas de automatización.

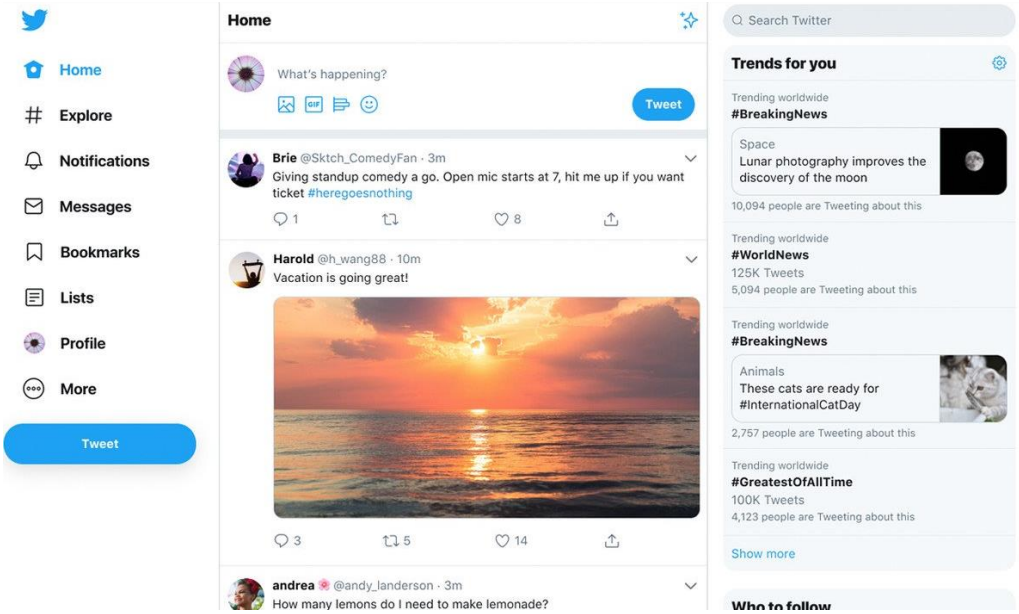

Figura 2.1-1Vista general de Twitter

En nuestro caso nos interesa ver los bots como una poderosa herramienta de marketing, ya que se pueden automatizar para responder a determinadas frases o palabras, como ya existen actualmente. El "problema" de estos bots es que siempre tiene que haber alguien (normalmente un community manager) que, en caso de respuesta a nuestro tweet "automático", siga con la conversación de forma natural.

Aquí un pequeño resumen del "vocabulario" que se usa en Twitter:

- Tweet o tuit: La palabra inglesa, o su transcripción a la española, significa trino, pero aquí se refiere a ese pequeño mensaje que se publica en la red social y que tiene un máximo de 280 caracteres.

- Seguidores o followers: Son los usuarios (personas o empresas) que nos siguen y que pueden ver en su timeline todo aquello que publicamos.

- Seguidos o followed: Son aquellos usuarios a los que seguimos y cuyas publicaciones veremos en nuestro timeline.

- Timeline: Es el conjunto de publicaciones que vemos en nuestra pantalla principal de Twitter. Allí aparecen en orden cronológico las publicaciones más recientes de todos los usuarios a los que seguimos y las nuestras propias.

- Retweet o retuitear: Es la acción de republicar un mensaje original de otro usuario. Normalmente, se realiza cuando uno se siente identificado con lo que se expresa, cuando se quiere ayudar a una promoción de algún mensaje o,

simplemente, cuando se quiere compartir entre los seguidores algo visto en el timeline propio. A veces se abrevia poniendo RT.

- Retuitear con comentario: Esto lo que hace es añadir al retuit clásico un comentario propio, para matizar o responder al tuit original. Puede ser considerado también otro tipo de respuesta, aunque su alcance es mayor, ya que les llegará a todos los seguidores y no solo al usuario original.

- Respuesta: Tan sencillo como eso mismo, lo que se tuitea respondiendo a una persona sobre un tuit anterior suyo.

- Menciones: En un tuit se puede mencionar a otro tuitero de un modo tan fácil como poner una "@" seguida de su nombre de usuario.

- Hashtag: También conocidas como etiquetas, son los temas en los que se pueden categorizar los tuits y se crean poniendo una '#' delante de una palabra o una frase.

- Trending Topic o tendencias: Son los temas de los que más se hablan en un momento determinado y en una ubicación geográfica concreta. Su abreviatura más utilizada es TT.

- Direct Message: Twitter permite enviar mensajes directos a algunos usuarios. Lo más habitual es que estos mensajes se puedan enviar a los seguidores propios, aunque también es posible a veces hacerlo a otros usuarios con los que no hay vínculos, según la configuración de la cuenta de cada uno. Las abreviaturas para esto son DM o MD, según el idioma que se utilice.

- Listas: Una lista de usuarios es la que sirve para configurar un timeline sobre un tema en concreto. Se pueden hacer listas para seguir a periódicos deportivos, a gente que hable de música o a una etiqueta determinada, de modo que se pueda tener una visión más ordenada de los temas de interés.

- Me gusta: Antes eran unas estrellitas que significaba favorito, o FAV, y ahora son unos corazones que expresan que un tuit es del gusto de un usuario. Cuantos más MG y RT tenga un tuit, más exitoso se puede considerar.

2.2 Procesamiento del lenguaje natural

Por Procesamiento del Lenguaje Natural, estamos hablando de técnicas computacionales que procesan lenguaje humano hablado y escrito. Esto es una definición inclusiva que abarca todo desde aplicaciones mundanas como conteo de palabras, hasta aplicaciones de punta como sistemas de respuesta de preguntas en la Web, o traducción automática en tiempo real. Lo que distingue a estas aplicaciones de procesamiento del lenguaje de otros sistemas de procesamiento de datos es el uso del conocimiento del lenguaje. (Jurafsky, D. y Martin, J (2008). Speech And Language Processing. Upper Saddle River: Pearson Prentice Hall)

Las personas pueden hablar en inglés, español o francés, pero un ordenador no puede comprender por si sola estos idiomas, ya que a bajo nivel los ordenadores se comunican con otro tipo de lenguajes. Hace años la computación estaba reducida a órdenes binarias muy simples, incluso se usaban tarjetas perforadas para comunicarse con los ordenadores. Hoy en día hay dispositivos que reconocen la voz y con un simple "¿Qué tiempo hace hoy Siri?" son capaces de devolverte las condiciones climatológicas de tu zona o al decir "Quiero escuchar música clásica Siri" ponga en marcha tu aplicación predeterminada de música con una playlist con las sonatas de Beethoven o las fugas de Bach.

Para conseguir esto, el dispositivo se activó cuando reconoció una voz, capto el mensaje y mediante PLN fue capaz de ejecutar una orden a bajo nivel.

Es posible dividir el estudio del lenguaje natural en cuatro niveles de análisis: morfológico, sintáctico, semántico y pragmático. El morfológico consiste en detectar la relación entre las unidades mínimas que forman una palabra (morfemas), reconocerla y construir una representación estructurada. El sintáctico tiene como cometido el etiquetado de cada componente sintáctico (sintagma) que aparece en una oración, construyendo en el proceso un árbol de análisis sintáctico, el cual tiene como nodo raíz el símbolo de oración, y tiene como hojas a las palabras. Por último el semántico es la tarea de procesar el lenguaje natural y comprender su significado, mientras que la pragmática provee el contexto del lenguaje.

El PLN es usado en muchos ámbitos:

- Para detectar SPAM en nuestra bandeja de entrada, analizando el asunto del correo y buscando similitudes.
- Cuando se navega en un sitio web usando su barra de búsqueda, o con las etiquetas sugeridas, usando modelado de remas, extracción de entidades y categorización de contenido, métodos de los PLN.
- En la traducción automática, resúmenes automáticos, etc.

2.3 Base de datos no relacional (NoSQL)

Casi todas las aplicaciones (sobre todo las aplicaciones web) utilizan las bases de datos para guardar y servir la información. Hasta ahora estábamos acostumbrados a utilizar bases de datos SQL como son MySQL, Oracle o MS QL, pero desde hace ya algún tiempo han aparecido las llamadas bases de datos no relacionales, para hacer frente a algunos problemas que no alcanzaban a corregir las "rígidas" bases de datos SQL.

El termino NoSQL aparece con la "web 2.0", es decir, con la llegada de aplicaciones como Facebook, YouTube o el propio Twitter, ya que suponía un crecimiento exponencialmente de los datos. La manera de estructurar los datos que proponen las bases de datos relacionales es muy rígida, obliga a emplear estructuras muy estrictas en las que manejamos un número fijo de campos que además tienen que almacenar datos de un determinado tipo. Esto hizo que empezaran a plantearse opciones más flexibles.

Por ello surgen las bases de datos no relacionales, que nos permiten trabajar con estructuras que pueden almacenar información en situaciones que las bases de datos relacionales generan problemas principalmente por sus problemas de escalabilidad y rendimiento debido a las miles de conexiones que deben de hacer diarias para consultas o de usuarios concurrentes. Estas no tienen un identificador que sirva de relación entre un conjunto de datos y otros. Los datos se organizan mediante documentos y esto es útil cuando no tenemos un esquema exacto de lo que se va a almacenar.

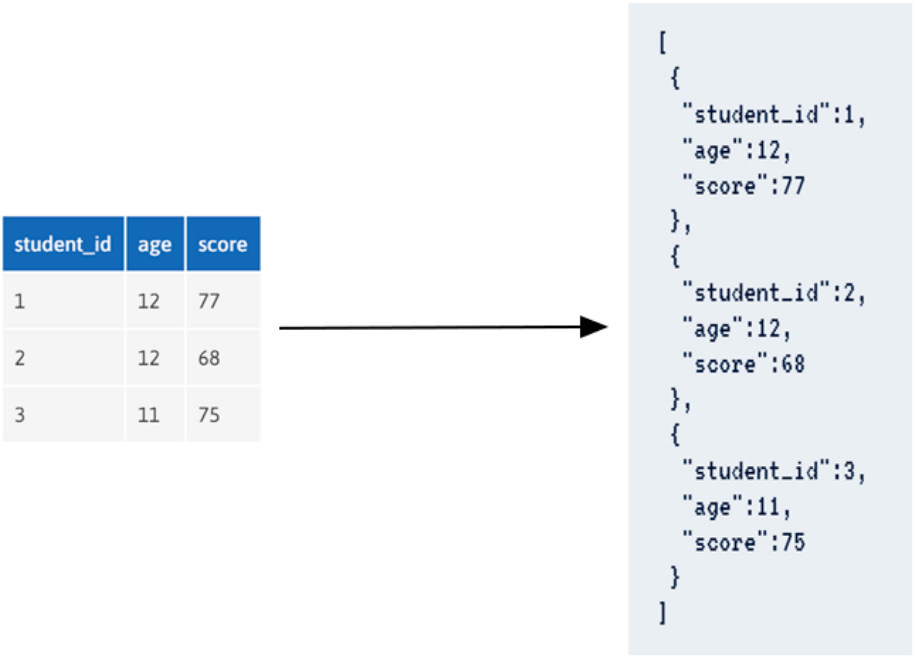

Figura 2.3-1: Formato de "esquema" de una NoSQL

¿Por qué se usa este tipo de "esquema"? A veces una unidad de datos puede llegar a ser demasiado compleja como para estructurarla en una tabla. Un ejemplo puede ser un documento JSON como el que vemos en la figura 2, en una base de datos relacional sería complicado al tener elementos jerárquicos.

Hay varios tipos de NoSQL:

- Clave-Valor: Son las más sencillas, se puede ver como una lista de parejas, con una clave a la que corresponde un único valor. Buscan la sencillez y rapidez en la recuperación de datos.

- Orientadas a Documentos: Se utiliza para manejar estructuras de datos más complejas. Los documentos se agrupan en colecciones. Si queremos hacer un símil con bases de datos relacionales, un documento sería un registro mientras que una colección equivaldría a una tabla.

Figura 2.3-2: Estructura de una NoSQL orientada a documentos

- Orientadas a columna: Tienen una estructura similar a las tablas de una base de datos relacional, la diferencia está en que agrupan las columnas por familias que permiten acceder a ellas de una manera más rápida y eficiente.

- Orientadas a grafo: La información estaría almacenada en los nodos, y las flechas que conectan unos con otros representan las relaciones. Están pensadas para manejar los tipos de estructura de datos en grafos.

Esto supone una serie de ventajas, como que se pueden ejecutar en máquinas con menos recursos, tienen una escalabilidad horizontal para mejorar el rendimiento (añadiendo más nodos por ejemplo), pueden manejar una gran cantidad de datos y no generan cuellos de botella.

Capítulo 3
Tecnologías utilizadas

3.1 Twitter API

Según la documentación oficial de Twitter:

Twitter es lo qué está pasando en el mundo y sobre lo que las personas están hablando en este momento. Puedes acceder a Twitter en la Web o desde tu dispositivo móvil. Para compartir información en Twitter de la forma más amplia posible, también les proporcionamos a las empresas, los desarrolladores y los usuarios acceso programático a los datos de Twitter mediante nuestras API (interfaces de programación de aplicaciones). Este artículo explica qué son las API de Twitter, qué información podemos extraer de ellas y algunas protecciones que Twitter instauró para su uso.

Pero, ¿Qué es una API? Este término es una abreviatura de Application Programming Interfaces, lo que traducido sería interfaz de programación de aplicaciones. Es un conjunto de definiciones y protocolos que se utiliza para desarrollar e integrar el software de las aplicaciones. En otras palabras, permite comunicar dos aplicaciones entre si gracias a un conjunto de reglas.

Para acceder a la API de Twitter hay que estar registrado en la aplicación. Con la autentificación básica se puede acceder a la información pública de Twitter (que es la que nos interesa). Si se quiere acceder a información como los mensajes directos hay que otorgar permisos adicionales.

Esta API permite cinco puntos de conexión:

- Cuentas y usuarios: Se puede administrar el perfil y la configuración de una cuenta, silenciar y bloquear usuarios, administrar usuarios y seguidores, etc.

- Tweets y respuestas: Se pueden publicar tweets a través de la API, así como responder a otros usuarios. Se pueden filtrar tweets por palabras

específicas o solicitar una muestra de tweets de cualquier cuenta.

- Mensajes directos: Si se otorgan permisos de manera excepcional, se pueden enviar mensajes directos a cualquier usuario (que permita recibirlos). Esto se suele usar mucho para campañas de marketing entre otras utilidades.

- Herramientas y SDK del editor: Son herramientas que se usan para integrar las cronologías de Twitter, el botón de compartir y otras herramientas en cualquier página web.

En nuestro caso vamos a usar la API para obtener el TimeLine de nuestra cuenta, con el fin de poder analizar los tweets que se obtengan. La API también devuelve el nombre del usuario que escribió el tweet y una referencia al mismo tweet, para poder ser respondido.

Las respuestas las envía en formato JSON, y los usuarios de deben registrar usando la autenticación y autorización OAuth. También cuenta con un servicio de streaming, que proporciona un acceso a un alto volumen de tweets con una baja latencia. Se suelen combinar para conseguir un mejor resultado.

3.2 FreeLing

FreeLing es una librería de código abierto para el procesamiento multilingüe automático, que proporciona una amplia gama de servicios de análisis lingüístico para diversos idiomas. La versión actual, FreeLing 4.0, permite identificar el lenguaje al que pertenece una expresión lingüística, dividirla en oraciones, analizarla y etiquetarla morfosintácticamente, detectar y clasificar sus entidades con nombre, reconocer sus fechas, números, magnitudes físicas y porcentajes, analizar sus dependencias sintácticas, etiquetar su significado mediante synsets de (Euro)WordNet5 y resolver sus correferencias.

El proyecto se estructura como una librería que puede ser llamada desde cualquier aplicación de usuario que requiera servicios de análisis del lenguaje. Se distribuye bajo una licencia GNU y bajo licencia dual a empresas que deseen incluirlo en sus productos comerciales.

FreeLing cumple, entre otras, las funciones de tokenizer, etiquetador gramatical, y analizador sintáctico, de modo que se deben analizar los formatos esperados para cada tipo de funcionalidad. En el caso de la tokenización, la herramienta espera recibir sencillamente el texto a tokenizar.

El resultado devuelto es el texto tokenizado con exactamente un token por línea. Cuando actúa como etiquetador, se espera recibir los tokens del modo en que fueron devueltos por el tokenizer: uno por línea.

Es un proyecto realizado por Lluís Padró, de la Universitat Politècnica de Catalunya, que empieza en el año 2003 con tres idiomas, castellano, inglés y catalán. En cada actualización se han ido añadiendo idiomas con las diversas funcionalidades, además de la inclusión de la variante diacrónica del español de los siglos XII al XVI, teniendo actualmente los que vemos en la siguiente tabla:

	as	ca	cy	en	es	gl	it	pt	ru
Tokenization	X	X	X	X	X	X	X	X	X
Sentence splitting	X	X	X	X	X	X	X	X	X
Number detection		X		X	X	X	X	X	X
Date detection		X		X	X	X		X	X
Morphological dictionary	X	X	X	X	X	X	X	X	X
Affix rules	X	X	X	X	X	X	X	X	
Multiword detection	X	X	X	X	X	X	X	X	
Basic named entity detection	X	X	X	X	X	X	X	X	X
B-I-O named entity detection				X	X	X			
Named Entity Classification				X	X				
Quantity detection		X		X	X	X		X	X
PoS tagging	X	X	X	X	X	X	X	X	X
WN sense annotation		X		X	X				
UKB sense disambiguation		X		X	X				
Shallow parsing	X	X		X	X	X		X	
Full/dependency parsing	X	X		X	X	X			
Coreference resolution					X				

Tabla 3-1: Idiomas soportados por FreeLing

La herramienta FreeLing no fue concebida como una aplicación de propósito general, es decir, no está diseñada para ser fácil de usar, o para mostrar sus resultados mediante una interfaz amigable. Aun así, su API (Application Programming Interface, Interfaz de Programación de Aplicaciones) incluye una aplicación bastante completa, que permite a un usuario final, sin conocimientos de programación, obtener el análisis lingüístico de un cierto texto. Aunque la interfaz básica ofrece un conjunto de opciones que cubren la mayor parte de sus funciones, es mucho más potente cuando se usa como API. De esta forma, se puede sacar mucho más partido a esta herramienta.

Aquí algunas de las características de FreeLing:

- Tokenización
- Segmentación en oraciones.
- Análisis morfológico.
- Tratamiento de sufijos.
- POS-tagging.
- Shallow-parsing.
- NER.
- Predicción probabilística de categorías de palabras desconocidas.

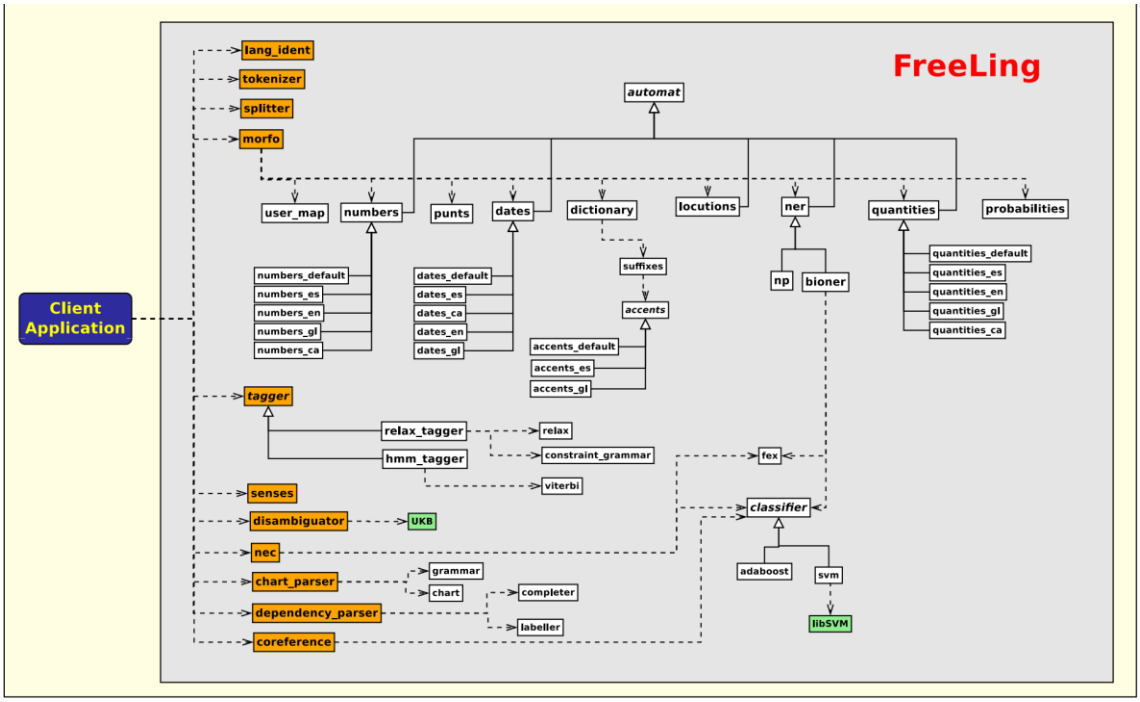

Figura 3.2-1: Clases de procesamiento FreeLing

3.3 MongoDB

Según la página oficial de MongoDB:

MongoDB es una base de datos documental, lo que significa que almacena datos en forma de documentos tipo JSON. Creemos que esta es la forma más natural de concebir los datos; frente al tradicional modelo de filas y columnas, esta es mucho más expresiva y potente.

Como bien se detalla en el párrafo anterior, MongoDB se estructura en "documentos" que tienen formato JSON, lo que quiere decir es que los datos se guardan en estos documentos, en lugar de en registros como las bases de datos relacionales. Puede contener strings, arrays, subdocumentos y números. La peculiaridad es que estos documentos no tienen por qué seguir un patrón o estructura, son "independientes" unos de otros.

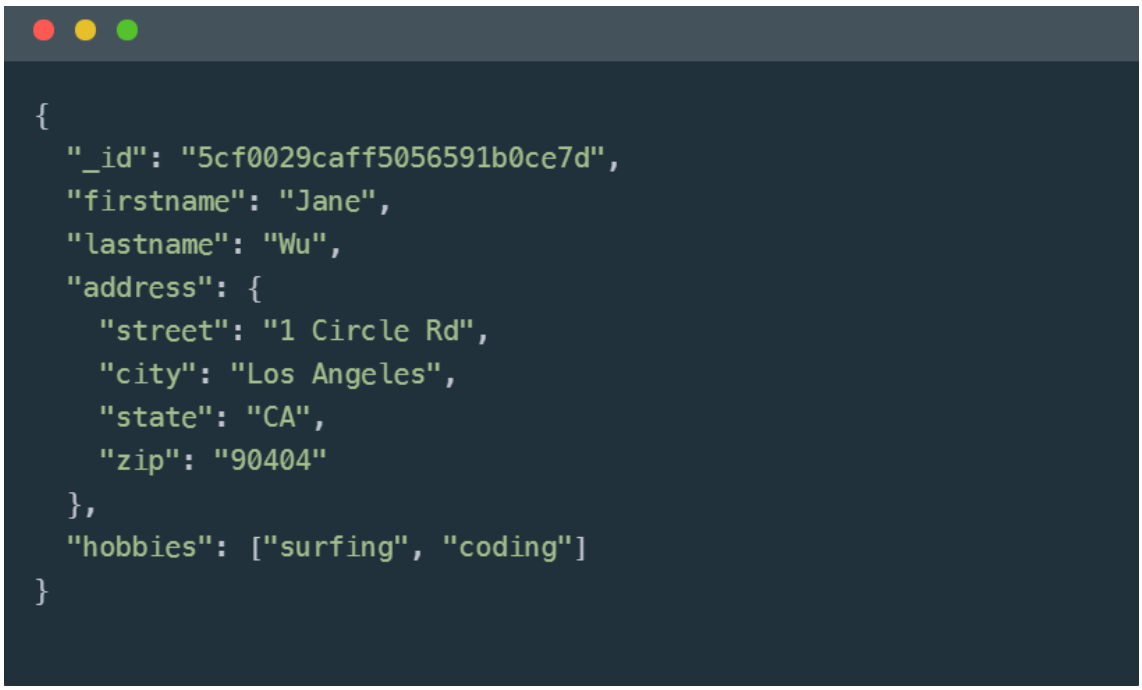

Figura 3.3-1: "Documento" en formato JSON de una base de datos MongoDB

MongoDB está escrito en C++, aunque las consultas se hacen pasando objetos JSON como parámetro. Viene de serie con una consola desde la que podemos ejecutar los distintos comandos. Esta consola está construida sobre JavaScript, por lo que las consultas se realizan utilizando ese lenguaje. En dicha consola se pueden declarar variables, funciones y usar bucles. También podemos utilizar muchas de las funciones propias de JavaScript.

Existen drivers y conectores para poder usarlo en otros lenguajes de programación, como C#, Java, Node.js, PHP, Python, Ruby, C, C++, Perl o Scala. Hay que tener en cuenta que no en todos los lenguajes están bien desarrollados estos drivers, por ejemplo en C está en una versión alpha, mientras que en Java está en una versión bastante avanzada.

La utilidad de MongoDB es que nos permitirá definir "documentos" a modo de registros en los que podamos variar el número de palabras clave, poder usar imágenes o enlaces, incluso en el formato de las descripciones. Esta escalabilidad es lo que nos permite agrandar la base de datos sin tener una estructura compleja definida.

3.4 Visual Studio Code

Visual Studio Code es un editor de código fuente desarrollado por Microsoft, que está disponible para Windows, macOS y Linux. Viene con soporte integrado para JavaScript, TypeScript y Node.js y tiene un rico ecosistema de extensiones para otros lenguajes (como C++, C #, Java, Python, PHP, Go) y runtimes (como .NET y Unity). Visual Studio Code se basa en Electron, un framework que se utiliza para implementar Chromium y Node.js como aplicaciones para escritorio, que se ejecuta en el motor de diseño Blink. Aunque utiliza el framework Electron, el software no usa Atom y en su lugar emplea el mismo componente editor (Monaco) utilizado en Visual Studio Team Services (anteriormente llamado Visual Studio Online).

Lanzado en 2015 por Microsoft, ha ido ganando fama entre los programadores a lo largo de estos años, hasta incluso ser nombrado como el editor preferido de los desarrolladores en múltiples ocasiones. Este editor se impone ante otros como su predecesor Visual Studio, Atom o Sublime.

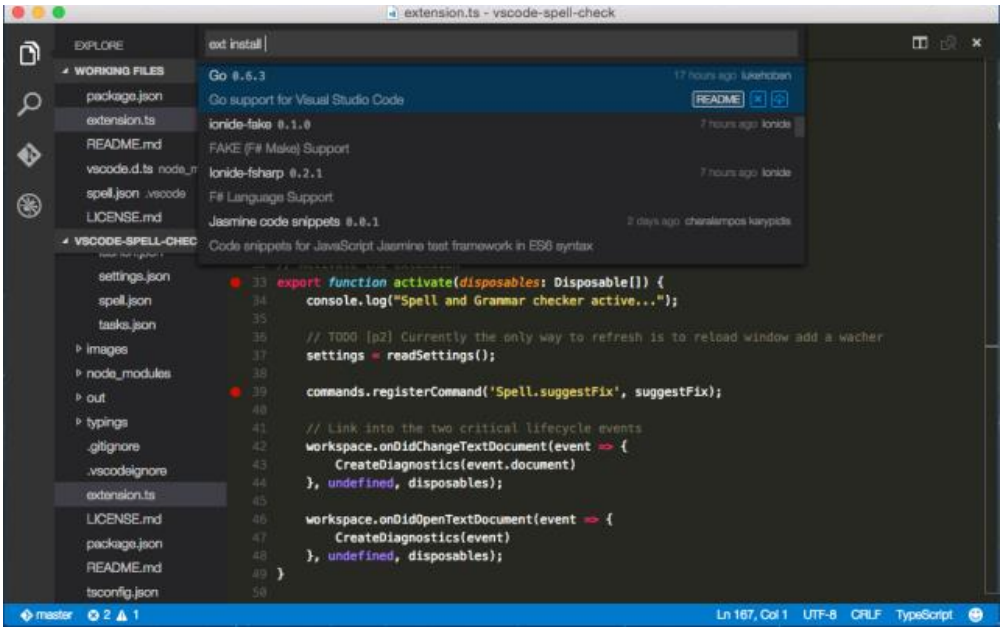

Figura 3.4-1: Vista del Visual Studio Core

Una de las grandes ventajas del VSC son sus extensiones. Dispone de una gran cantidad de recursos y plugins a través de su marketplace, que sirven para optimizar y agilizar el desarrollo de aplicaciones entre otras cosas. En este proyecto hemos usado Remote – ssh, JavasCript code snippets, Maven For Java o Git History entre otros.

3.5 Java

Java es un lenguaje de programación y una plataforma informática comercializada por primera vez en 1995 por Sun Microsystems", resume la propia web de Java qué es esta tecnología. Nació con el objetivo de ser un lenguaje de programación de estructura sencilla que pudiera ser ejecutado en diversos sistemas operativos. En lo que respecta a su nombre, inicialmente iba a denominarse Oak pero como esta marca ya estaba registrada se acabó optando por Java.

Java sirve para crear aplicaciones y procesos en una gran diversidad de dispositivos. Se base en programación orientada a objetivos, permite ejecutar un mismo programa en diversos sistemas operativos y ejecutar el código en sistemas remotos de manera segura. Su ámbito de aplicación es tan amplio que Java se utiliza tanto en móviles como en electrodomésticos.

3.5.1 Maven

Apache Maven es una herramienta de comprensión y gestión de proyectos de software. Basado en el concepto de un modelo de objetos de proyecto (POM), Maven puede administrar la construcción, informes y documentación de un proyecto a partir de una pieza central de información.

3.6 JSON

Es un formato de texto ligero para el intercambio de datos que puede ser leído desde cualquier lenguaje de programación. Este tipo de fichero nos permitirá transferir datos usados por la aplicación web de forma simple y rápida. Su sencilla estructura facilita la tardea del programador a la hora de gestionar dichos ficheros.

.

3.7 GitHub

Para empezar, debemos definir que es Git: es un software de control de versiones. El control de versiones, resumiéndolo mucho, es la gestión de los diversos cambios que se realizan sobre un repositorio (un repositorio es el nombre que recibe el lugar donde se aloja el código de un proyecto de desarrollo en algún lenguaje de programación).

Entonces, ¿Qué es Github? Github es una plataforma de desarrollo colaborativo de software para alojar proyectos usando el sistema de control de versiones Git. El código se almacena de forma pública, aunque también se puede hacer de forma privada, creando una cuenta de pago. También se pueden obtener repositorios privados (de pago) si se es estudiante. Github no sólo ofrece alojamiento del código si no muchas más posibilidades asociadas a los repos como son, forks, issues, pull requests, diffs, etc.

Capítulo 4
Desarrollo

4.1 Metodología y fases del proyecto

El proyecto se dividió en tres grandes partes: la integración de la API de Twitter, el análisis de FreeLing y la implementación de la base de datos MongoDB.

4.2 Twitter4J

El primer paso fue crear una aplicación en java para poder integrarla con la API de Twitter. Se instalaron las siguientes extensiones: Java Extension Pack, Debugger for Java, Git History, GitLens, Java Language Support, Java Test Runner, Language Support for Java, Maven for Java,Project Manager for Java y Visual Studio IntelliCode.

Luego en la paleta de comandos selecionamos “archetype-quickstart-jdk8” para crear el proyecto con el arquetipo de Maven. Este comando nos genera la estructura de nuestro proyecto. Llamamos al proyecto **Twitter4j**.

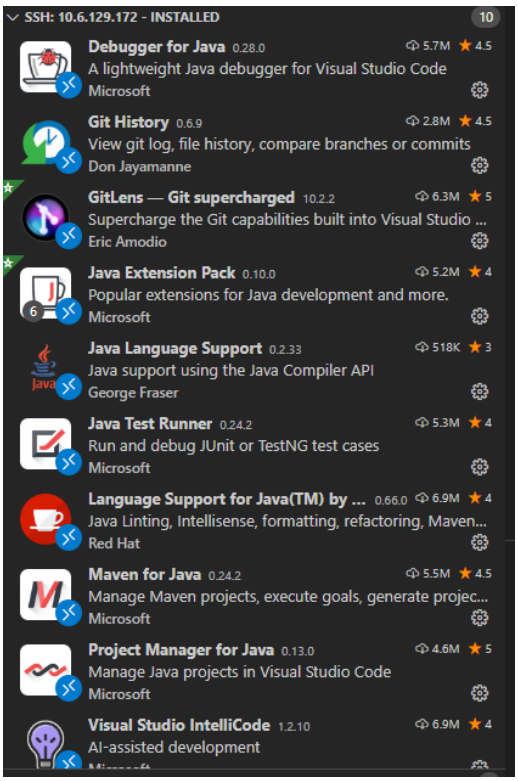

Figura 4.2-1: Extensiones VSC

Una vez la estructura creada, se genera un archivo llamado *pom.xml*, con el que controlaremos las librerías, que son tratadas como "artefactos", lo que incluye toda la información necesaria para su correcta gestión (grupo, versión, dependencias, etc). Para añadir la librería que controla la API de Twitter para java (Twitter4j) vamos a la página de Maven donde tienen todos sus repositorios (https://mvnrepository.com/).

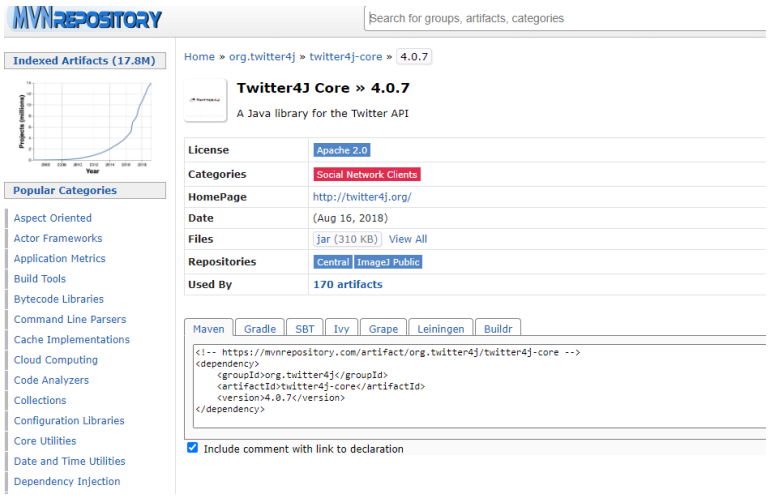

Figura 4.2-2: Librería de Twitter4j

Este código lo introducimos en el fichero *pom.xml* y hacemos un "build" de maven para integrar la librería. Con esto podemos acceder a los métodos y objetos que nos ofrece Twitter en su API.

Para ello tenemos que darle permisos a Twitter para controlar nuestra cuenta con su API desde su pagina **developers.twitter.com**. Debemos crear una "app", en nuestro caso se va a llamar "TFG NoelPadron", lo que nos generará un token de acceso y un token de acceso secreto. Estos datos estarán en un archivo llamado "twitter4j.properties".

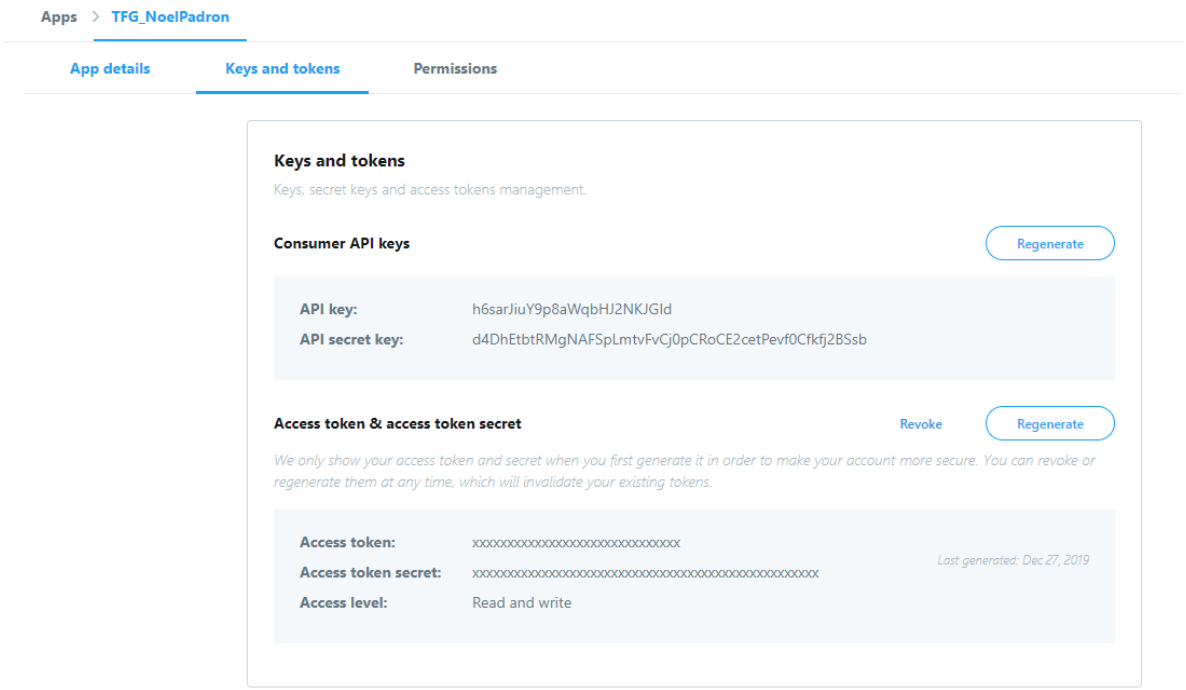

Figura 4.2-3: Tokens de acceso y app de Twitter

Con el acceso a Twitter lo primero que probamos es obtener tweets. Lo primero es crear un objeto de la clase TwitterController (que es la que controla la API), y seguido una lista de "Status", que serán los tweets que obtendremos de nuestra TimeLine filtrados por la palabra que decidamos. De cada objeto "Status" podemos obtener el id del tweet, el nombre de usuario y el contenido.

Con estos datos ya podemos responder a los tweets, pero para ello necesitamos un objeto de la clase “StatusUpdate”, que es la que nos permite construir la estructura del tweet. Para publicarlos usamos el método “reply” de la clase “TwitterController”, y le pasamos como parámetro el objeto de la clase “StatusUpdate”.

```
TwitterController twittear = new TwitterController(consolePrint); //Objeto de la clase TwitterController
String message = "Prueba"; //Texto del tweet
long inReplyToStatusId; //Variable en la que ira el id del tweet a responder
String command = QueryReader.readLine("Query in twitter (quit for exit)->>"); //Palabra a analizar
while (!command.equals("quit")) {
    List<Status> result = twittear.query(command); //Funcion de la clase TwitterController para almacenar los tweets que coincidan
    for (Status tweet : result) {
        inReplyToStatusId = tweet.getId();
        message = message + " @" + tweet.getUser().getScreenName();
        System.out.println(tweet.getUser().getScreenName());
        StatusUpdate stat= new StatusUpdate(message);
        stat.setInReplyToStatusId(inReplyToStatusId);
        twittear.reply(stat);
    }
    twittear.printStatus(result);
    //twittear.tweetOut(message);       You, a few seconds ago • Uncommitted changes
    command = QueryReader.readLine("Query in twitter: ");
}
```

Figura 4.2-4: Código descrito anteriormente

```
usuario@ubuntu:~/TFG_Noel_Padron$  /usr/lib/jvm/java-11-openjdk-amd64/bin/java -Dfile.encoding=UTF-8 @/tmp/cp_dbeo3f34mrp0dh05exbqp0vf3.argfile twitter4j.TwitterMain
Query in twitter (quit for exit)->>cerveza
rosyavf
elhugodice
Carloscerda9
Ymobad
signvftimes
yeygse
GxldSauc3
Rare246
martagarciag_
songforMozart
AndyAnd3rs0n
viojodevenado
adrianrdaadri
ismaahn6
FilomenoGarchit
Tweet Numero: 1
------------------------------------------------------------
User [rosyavf]
RT @cherovapete_: De la rabia voy a tomar una cerveza. En eso nomas luego iba a terminar este asunto
Id tweet: 1303812931499503616
09/09/2020 22:50:01
RT[73] FAV[0]
------------------------------------------------------------
Tweet Numero: 2
------------------------------------------------------------
User [elhugodice]
RT @sopitas: Esa camioneta no es para repartir cervezas, oye... 😂😂😂 😂😂😂  https://t.co/gDq1Yp5msI
Id tweet: 1303812900805541896
09/09/2020 22:49:53
RT[61] FAV[0]
------------------------------------------------------------
Tweet Numero: 3
------------------------------------------------------------
```

Figura 4.2-5: Salida del código

4.3 FreeLing

Para integrar la librería de FreeLing en nuestro proyecto también usamos maven. Se volvió a consultar la web de los repositorios de maven para ver el código de la dependencia, y a hacer un "build" del *pom.xml* de maven.

Aquí nos encontramos bastantes problemas a la hora de usar la librería, asi que se decidió usar FreeLing a través de comandos, ya que estaba instalado de forma local en la máquina virtual que nos proporcionó el IaaS. Para ello necesitamos instalar previamente las siguientes librerías:

- libboost, libicu, and libz libraries.
- libboost-regex
- libboost-system
- libboost-thread
- libboost-program-options
- zlib

Una vez instaladas las dichas librerías, debemos descargarnos el paquete de la página oficial de GitHub (https://github.com/TALP-UPC/FreeLing/releases). Luego hay que modificar el fichero "CMakeLists.txt" para que construya el API de Java. Una vez modificado, ejecutamos el comando "cmake" y se instalaría en el SO.

```
cmake_minimum_required(VERSION 3.8)

# disallow in-source builds
if(${CMAKE_SOURCE_DIR} STREQUAL ${CMAKE_BINARY_DIR})
   message("")
   message(" You are attempting to build in your source Directory.")
   message(" You should run cmake from a build directory.")
   message("")
   message(FATAL_ERROR "Build aborted.")
endif()

project(FreeLing)
set(PACKAGE_NAME "FreeLing")
set(VERSION "4.1")
set(PACKAGE_STRING "\"${PACKAGE_NAME} ${VERSION}\"")
set(CMAKE_INSTALL_RPATH "${CMAKE_INSTALL_PREFIX}/lib") # Needed to find libraries if not installed in system path

# Add compiler definitions
add_definitions(-DPACKAGE_STRING=${PACKAGE_STRING} -DVERSION=${VERSION})

# Global configuration
set(CMAKE_CXX_STANDARD 11)
option(BUILD_SHARED_LIBS "Build shared libraries" ON)

# compilation options
option(BUILD_TESTS "Build tests" OFF)
option(TRACES "Enable traces" OFF)
option(WARNINGS "Enable warnings" ON)
option(XPRESSIVE "Xpressive regex" OFF)
option(EMBEDDINGS "Download word embeddings" OFF)
option(JAVA_API "Build Java API" ON)
option(PYTHON3_API "Build Python 3 API" OFF)
option(PYTHON2_API "Build Python 2.7 API" OFF)
option(PERL_API "Build Perl API" OFF)

# Check for dependencies -- ZLIB
find_package(ZLIB REQUIRED)

# Check for dependencies -- Boost
if(BUILD_TESTS)
  find_package(Boost COMPONENTS regex filesystem thread program_options unit_test_framework REQUIRED)
else()
  find_package(Boost COMPONENTS regex filesystem thread program_options REQUIRED)
endif()

# Check for dependencies -- ICU
if (WIN32)
```

Figura 4.3-1: Fichero CMake con la API de Java "ON"

Con FreeLing instalado, desde el propio programa vamos a ejecutarlo mediante un comando del sistema (*analyze –f es.cfg <ejemplo.tct*). Para ello usamos la librería "Process", que nos permite ejecutar una instrucción en una línea de comandos externa desde la ejecución de nuestro programa en Java.

Como el comando era complejo y daba fallos, se creó un script con el comando para ejecutar sin errores (*freeling.sh*). El comando tiene los siguientes argumentos:

- **-f:** *-from*, para seleccionar el idioma.
- **es.cfg:** La configuración del idioma que seleccionamos para analizar (en este caso el *es* representa el idioma de España).
- < **"fichero":** Se le pasa el fichero que contiene la cadena a analizar.

A la hora de analizar el contenido aprovechamos el bucle que recorremos obteniendo los *id's* de cada tweet, para, con una función llamada "analisis_freeling", ejecutar el análisis de FreeLing. La respuesta la parseamos para obtener las palabras que resultan del análisis, para así poder compararla con los registros de la base de datos que definiremos más adelante.

```
Mi mi DP1CSS 0.999325
nombre nombre NCMS000 0.995342
el el DA0MS0 1
Noel noel NP00000 1
y y CC 0.999989
me me PP1CS00 0.755196
gusta gustar VMIP3S0 0.99569
la el DA0FS0 0.98926
cerveza cerveza NCFS000 1
```

Figura 4.3-2: Resultado de un análisis de un texto de ejemplo

```
public static String analisis_freeling(String contenido) throws IOException{
    String ruta = "analizar.txt";
    File file = new File(ruta);
    // Si el archivo no existe es creado
    if (!file.exists()) {
        file.createNewFile();
    }
    FileWriter fw = new FileWriter(file);
    BufferedWriter bw = new BufferedWriter(fw);
    bw.write(contenido);
    bw.close();

    Process proc = Runtime.getRuntime().exec("./freeling.sh");

    BufferedReader reader =
    new BufferedReader(new InputStreamReader(proc.getInputStream()));

    String line = "";
    ArrayList<String> analisis = new ArrayList<String>();
    while((line = reader.readLine()) != null) {
        analisis.add(line);
    }
    for(int i=0 ; i<analisis.size(); i++){
        String[] parts = analisis.get(i).split(" ");
        for(int j=0; j<parts.length-1; j++){
            if(parts[j].charAt(0) == 'N' && (j-1) > 0){
                System.out.println(parts[j-1]);
                return parts[j-1];
            }
        }
    }
    analisis.remove((analisis.size()-1));
    return "";
}
```

Figura 4.3-3: Función "analisis_freeling"

La respuesta tiene, como se puede comprobar en la figura 4.3.2, la palabra original, la palabra “raíz” que obtiene después del análisis, una serie de caracteres que nos indican el tipo de palabra que analiza y un número entre 0 y 1 que corresponde al nivel de acierto con el que calculó la palabra “raíz”. Son estos parámetros los que usaremos para determinar la respuesta a cada tweet.

Lo que buscamos en cada tweet son los nombres, ya sean nombres comunes como nombres propios. Por ello es que se parsea el resultado y se busca, en la tercera posición de cada array (que es donde se define siempre el tipo de palabra) todos aquellos que empiecen por la letra “N”. Es posible que la función nos devuelva “” como respuesta, y se da cuando el tweet no contiene nombres comunes. En ese caso se descarta y se pasa al siguiente.

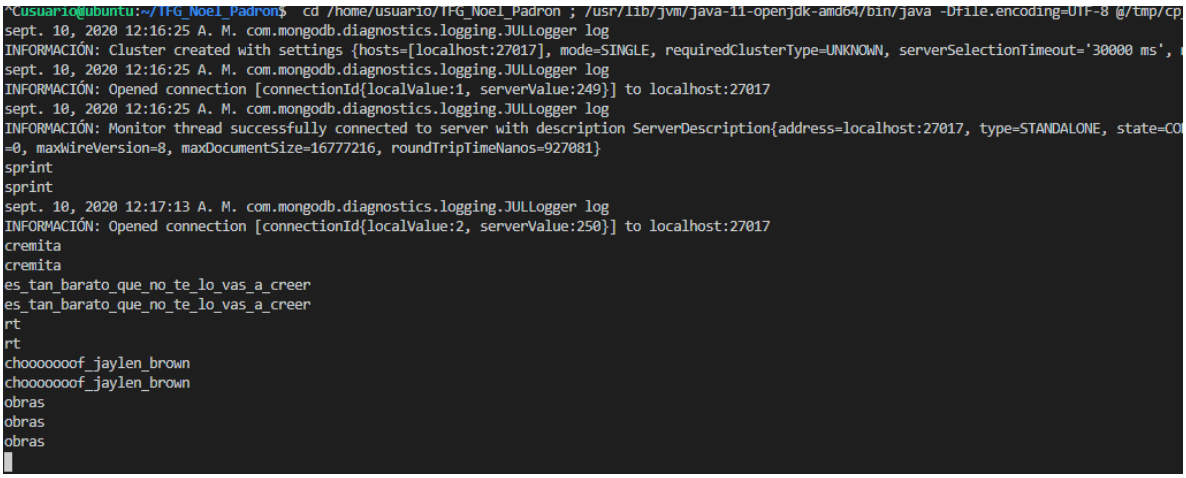

Figura 4.3-4: Resultado del filtro de FreeLing con los nombres propios

4.4 Base de datos MongoDB

Usaremos MongoDB porque como describimos antes, nos permite una flexibilidad con la estructura de las tablas que no permiten las bases de datos relacionales. Para añadirlo a nuestro proyecto usaremos maven como hemos hecho con las dos librerías anteriores, consultando en los repositorios de maven las dependencias para la versión 3.2.2 de MongoDB para Java.

Una vez añadida la librería, lo siguiente era crear la base de datos. Desde línea de comandos ejecutamos "use twitter", para crear la base de datos o la colección en caso de que todavía no existiese. Con el comando "show dbs" podemos comprobar que está creada, ya que nos devuelve la lista de las bases de datos que tenemos.

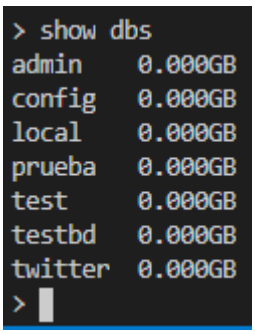

Figura 4.4-1: Lista de base de datos en nuestra máquina

Desde que tengamos una base de datos, podemos crear una colección. Con el comando "twitter.createCollection('newCollection1')" creamos la colección en la que van a insertarse los documentos con las palabras clave, descripciones, enlaces, etc.

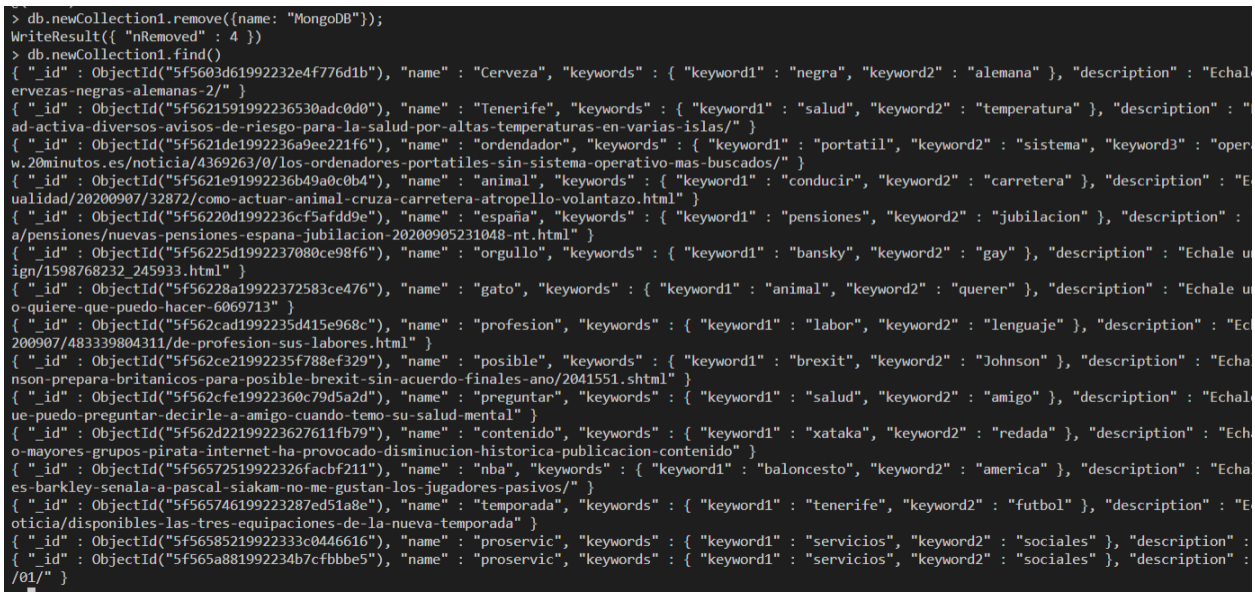

Figura 4.4-2: Muestra de documentos de la colección "newCollection1"

Con la base de datos y la colección creada, empezamos a introducir los documentos. Para ello establecemos primero la conexión con la base de datos, con la clase "MongoClientURI", a la que pasamos la dirección donde esta nuestra base de datos. En este caso la tenemos en local, asi que la dirección seria "mongodb://localhost:27017". Con estos parámetros definidos podemos crear un objeto de la clase "MongoClient", que tiene un método llamado "getDatabase" que conecta directamente con la base de datos que le pasemos como parámetro.

Ya definida la conexión al cliente MongoDB y a la base de datos, tenemos que acceder a la colección, para ello generamos una lista con cada documento de la colección usando el método "find" de la clase "MongoCollection". Si queremos insertar documentos en la colección, se repite el proceso de antes pero sin usar el método "find". Esto lo hacemos a la hora de insertar nuevos registros a la base de datos.

Una parte importante en la implementación de la base de datos es insertar los documentos en ella, que en nuestro caso van a tener tres registros fijos, y otro que puede variar dependiendo de las palabras que encontremos.

- **name:** El nombre principal por el que se va a filtrar.
- **keywords:** Un nuevo documento dentro del documento principal, en el que se van a definir más palabras clave para precisar al máximo la búsqueda.
- **description:** El texto descriptivo que aparecerá en el tweet acompañando al enlace.
- **link:** Guardamos el enlace a la noticia que se va a enviar en caso de coincidencia.

Una vez creado el documento lo insertamos con el método "insertOne" de la clase "MongoCollection", pasándolo como parámetro. Aparte tenemos una función "printBlock" para mostrar la colección en formato JSON.

```
public class MongoDB {
    public static void base_datos() {
        You, 2 days ago | 1 author (You)
        Block<Document> printBlock = new Block<Document>() {
            @Override
            public void apply(final Document document) {
                System.out.println(document.toJson());
            }
        };
        MongoClientURI connectionString = new MongoClientURI("mongodb://localhost:27017");
        MongoClient mongoClient = new MongoClient(connectionString);

        MongoDatabase database = mongoClient.getDatabase("twitter");
        MongoCollection<Document> collection = database.getCollection("newCollection1");


        Document doc = new Document("name", "proservic")
                            .append("keywords", new Document("keyword1", "servicios")
                                                    .append("keyword2", "sociales"))
                            .append("description", "Echale un vistazo a esta noticia! ")
                            .append("link", "http://www.n2bsolutions.com/es/blog/01/");
        collection.insertOne(doc);
    }       You, 3 months ago • Añadiendo freeling al maven y otros cambios
}
```

Figura 4.4-3: Función para introducir documentos

4.5 Integración y resultado final

Una vez tenemos todas las librerías funcionando por separado llega el momento de unificarlas para que se obtengan los tweets, se analicen y se respondan acorde a su contenido. Todo ello queda integrado en la clase de java "TwitterMain.java".

Lo primero que hacemos es la conexión a la base de datos como definimos en el apartado anterior, ya que tendremos que hacer una consulta por cada palabra "valida" que nos devuelva FreeLing, y abrir una conexión por cada consulta consume muchos recursos del ordenador.

El siguiente paso es declarar las variables que necesitamos para la ejecución, en este caso son dos **Strings** (*contenido* y *message*) y un **long** (*inReplyToStatusId*):

- **contenido:** Es donde se almacena el texto del tweet obtenido para después ser analizado.
- **message:** Es la variable donde se va a formar la respuesta a los tweets analizados.
- **inReplyToStatusId:** Guarda el id del tweet a responder.

El bloque principal estará englobado en un try/catch, por si salta alguna excepción de Twitter4j. Declaramos las instancias de las clases de la librería de Twitter necesarias (*Twitter, Twittercontroler, User*) y una lista de la clase "Status" donde se almacenan los tweets. Con un bucle *for* recorremos la lista de tweets, sobrescribiendo la variable "contenido" con el texto del tweet.

Ahora entra el análisis de FreeLing con la función "analisis_freeling" dentro de una condición *if*. Si el análisis no nos devuelve ninguna palabra que sea un nombre, no entra al código y pasa al siguiente tweet. En caso de devolver alguna palabra sigue avanzando.

Cuando tengamos las palabras claves del tweet, procedemos a hacer la búsqueda en la base de datos. Para ello tenemos la función "consulta_base_datos", a la que le pasamos la función "analisis_freeling" ya que devuelve la palabra clave para la búsqueda. Dentro de la función declaramos una lista de documentos, filtrando con el método "find" en el campo *name* de los registros. Si hay alguna coincidencia, la función devuelve la variable "respuesta", que contiene la descripción y el enlace del documento que fue devuelto y los guarda en la variable "message".

Con la variable "message" definida, obtenemos el *id* del tweet con el método "getId()". Se declara un objeto de la clase StatusUpdate (que como dijimos antes es la que se encarga de dar formato al tweet) y se envía con el objeto "twittear" de la clase "TwitterControler".

```
public static void main(String[] args) throws IOException, TwitterException {
    // MongoDB.base_datos();
    MongoClientURI connectionString = new MongoClientURI("mongodb://localhost:27017");
    MongoClient mongoClient = new MongoClient(connectionString);

    MongoDatabase database = mongoClient.getDatabase("twitter");
    MongoCollection<Document> collection = database.getCollection("newCollection1");

    String contenido = "";
    String message = "";
    long inReplyToStatusId;
```

Figura 4.5-1: Conexión a la base de datos y declaración de variables

```
try {
    // gets Twitter instance with default credentials
    Twitter twitter = new TwitterFactory().getInstance();
    TwitterController twittear = new TwitterController(consolePrint); //Objeto de la clase TwitterController
    User user = twitter.verifyCredentials();
    List<Status> statuses = twitter.getHomeTimeline();
    for (Status status : statuses) {
        contenido = status.getText();
        if(analisis_freeling(contenido) != ""){
            message = consulta_base_datos(analisis_freeling(contenido), collection) + " @" + status.getUser().getScreenName();
            inReplyToStatusId = status.getId();
            StatusUpdate stat= new StatusUpdate(message);
            stat.setInReplyToStatusId(inReplyToStatusId);
            twittear.reply(stat);
        }
    }
    //twittear.printStatus(statuses); //Muestra todos los tweets que ha almacenado del TimeLine con el formato que tiene en la defi
} catch (TwitterException te) {
    te.printStackTrace();
    System.out.println("Failed to get timeline: " + te.getMessage());
    System.exit(-1);
}
```

Figura 4.5-2: Bloque de código principal

Figura 4.5-3: Respuesta a un tweet despues de analizarlo y encontrar la palabra clave "gato"

Capítulo 5
Problemas encontrados

5.1 Limitaciones de la API de twitter

Este fue el primer problema encontrado a la hora de desarrollar. Una vez obtenido todos los datos de configuración y acceso (tokens) a Twitter, cada cierto tiempo la aplicación no se podía conectar con los servidores de Twitter.

Al principio pensábamos que era un problema de conexión interna, hasta que analizando el problema, buscando en la documentación oficial y buscando noticias se descubrió que Twitter había tomado medidas para prevenir su propia "Cambridge Analytica", con lo que redujo el número de conexiones diarias a la API, así como el número de tweets, mensajes directos, cambios en el correo electrónico, gente a la que sigues (por día) y gente que sigues (según la cuenta).

API	Petición	Max. datos por petición (página)	Cada 15 minutos
REST	GET statuses/user_timeline	200 tuits	900 * 200 = 180.000 tuits
REST	GET users_show	1 perfil	1 * 900 = 900 perfiles
REST	GET follwers_list	200 perfiles	200 * 15 = 3.000 perfiles
REST	GET followers_ids	5.000 ids de usuario	5.000 * 15 = 75.000 Ids
Search	GET search/tweets	100 tuits	180 * 100 = 18.000 tuits
Streaming	POST statuses_filter	–	Máximo de 45.000 tuits

Tabla 5-1: Límites de la API de Twitter

5.2 Integración FreeLing

Este fue el mayor problema en todo el desarrollo del proyecto. Tanto su instalación en el sistema operativo, como su integración con el proyecto.

La idea era integrar FreeLing con maven, pero para ello debía estar previamente instalado en el sistema operativo. Las dificultades vinieron al no tener una fuente de información fiable, ni una documentación bien redactada. Lo único que se encontraba en internet eran repositorios de GitHub con descripciones escuetas, y poca respuesta a los errores que aparecían (que no eran pocos).

Con mucha dificultad se logró instalar en el sistema operativo, y era cuando se tenía que probar con la librería ya integrada en la aplicación, pero esta al realizar las primeras pruebas no devolvía resultados. En un primer momento se pensó que el error estaba en las variables de entorno, que no estaban bien definidas o directamente no estaban definidas. Para ver las variables de entorno ejecutamos el comando "printenv". Para añadir la variable de FreeLing teníamos que ejecutar "export FREELINGDIR="/usr/local"".

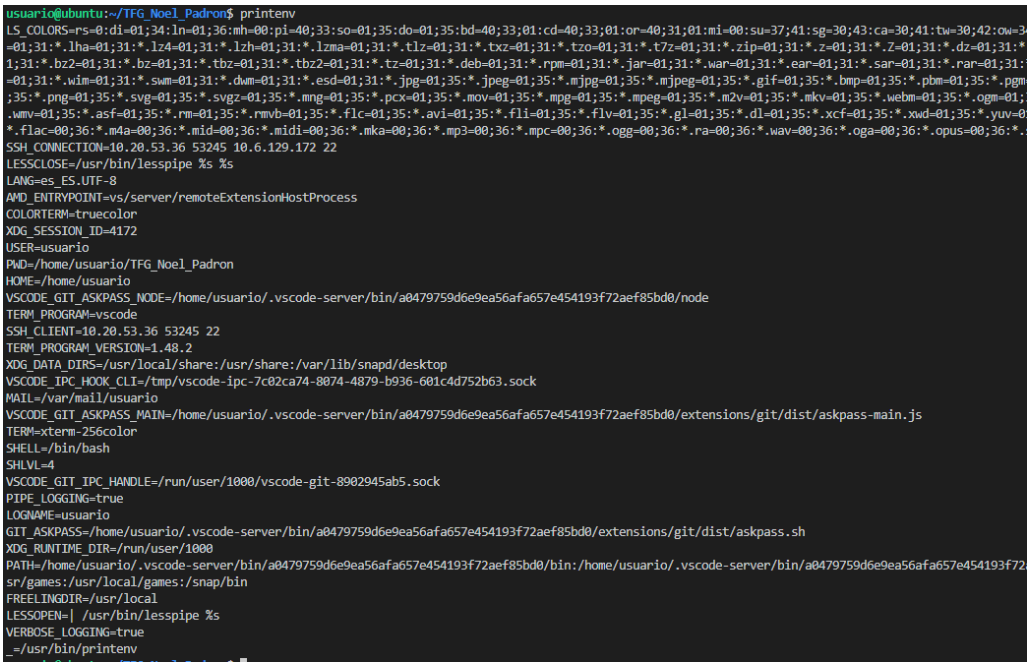

Figura 5.2-1: Variables de entorno de la máquina del IaaS

Con esto se logró que funcionara en local, pero no en la aplicación, así que la solución fue ejecutar el análisis mediante comandos. Se pasaba el texto a un fichero, y mediante un script (*freeling.sh*) ejecutar el análisis y guardar el resultado en una variable.

Capítulo 6
Conclusiones y líneas futuras

En relación a la idea inicial del trabajo, se han satisfecho las tres grandes partes que constituyen el proyecto, a excepción de la parte de FreeLing, ya que no se ha llevado a cabo una integración completa, pero cumple con éxito su cometido. La aplicación ha quedado completamente funcional, que cumple su cometido e incluso podría ser utilizada por alguna empresa o entidad que necesite difundir información con cierto criterio y con un margen de error aceptable.

Personalmente ha sido un año complicado, las circunstancias no han sido las idóneas para poder disfrutar de este proyecto, pero igualmente nunca había trabajado con el lenguaje de Java ni con bases de datos no relacionales, pero si era algo que tenía en mente y lo he aprovechado para aprender de cara a un futuro laboral. El caso del procesamiento de lenguaje natural era distinto, porque apenas tenía información y no era un campo que me planteaba abarcar en el futuro, y aunque la experiencia no fue la idónea (muchos errores, complicaciones y poca documentación) he descubierto que es un campo de la inteligencia artificial muy interesante y poco reconocido.

La aplicación, aunque sea funcional, tiene un potencial bastante grande a mi parecer. Para empezar se podría tener un criterio para elegir las noticias. Me explico, ahora mismo la aplicación tiene una base de datos con noticias variadas de todo tipo, pero se podría "especializar" en un tipo de noticias, y así dirigir la aplicación a un público determinado.

También se podría enfocar como una herramienta de marketing para cualquier empresa. Anunciando sus productos, usando imágenes en lugar de enlaces para realizar campañas de nuevos productos.

Twitter nos da la posibilidad a través de su API de enviar también mensajes privados, y esa es otra línea de futuro que puede ser explorada. Por ejemplo, Twitter es una red social en la que la gente muestra sus sentimientos de manera abierta, y muchas de esas personas quizás necesitan ayuda (ya sea psicológica, económica, etc.) que puede ser delicada de exponer en público. Se podría mejorar el sistema de análisis para reconocer estos patrones y enviar mensajes directos con pautas de auto-ayuda, información de ayudas económicas, etc.

El volumen de información y tweets que hay actualmente es muy elevado, y hay veces que nos perdemos información por no seguir a la gente que tiene gustos similares a los nuestros. El API de Twitter te da la opción de seguir a otras personas solo obteniendo los tweets, por lo que se podría desarrollar la aplicación para que a partir de determinado número de tweets que coincidan con las preferencias que se han descrito se empiece a seguir a esa persona.

Capítulo 7
Summary and Conclusions

Related to the first idea of the project, the three main parts that of the project have been satisfied, except the FreeLing part, because full integration has not been carried out, but it fulfills its mission. The application has been fully functional, as it fulfills its task and it could be even used by any company or entity which need to spread information with a certain criterion and with an acceptable error range.

Personally, it has been a tough year, the circumstances have not been ideal to be able to enjoy this project, but, equally, I had never worked with Java language or non-relational databases, but I did have that in my mind and I have taken advantage of it to learn looking to the Laboral future. The natural language processing case was different, because I hardly had any information and it was not a subject I had considered to approach in the future, and, although this experience has not been ideal (many mistakes, complications and few documentation) I have discovered it is a very interesting and little known field of the artificial intelligence.

I believe that the application, though it is functional, has a pretty great potential. To begin with, it would be possible to have a criterion to choose the news. Let me explain, currently, the application has a database with all kind of varied news, but it is able to "specialize" it in a determined kind of news, so you can aim the application to a specific audience.

Also, it would be possible to use it as a marketing tool for any firm. Advertising its products and using pictures rather than links to make their new products campaigns.

Twitter also allows us to send private messages through its API, and that is another future line that can be explored. For instance, Twitter is a social network where the people convey their feeling openly, and many of those people might need help (either psychological, economic, etc.) that could be sensible to expose in public. The analysis system could be improved to recognize these patterns and send direct messages with self-aid guidelines, economic aids information, etc.

The volume of information and tweets that is currently available is very high, and sometimes we miss information because we do not follow people who has similar interests than we have. The Twitter API give you the chance of following other people just obtaining tweets, so the application could be developed to star following a person who has a determined number of tweets that match your preferences.

Capítulo 8
Presupuesto

Ahora vamos a detallar las tareas realizadas, las horas invertidas y los servicios que son necesarios.

Descripción	**Precio**	**Cantidad**	**Total**
Planificación	15	5	75
Lectura de bibliografía y documentación	15	20	300
Formación PLN	15	15	225
Implementación Twitter API	15	25	375
Implementación FreeLing	15	45	675
Implementación MongoDB	15	20	300
Insertar datos en base de datos	15	10	150
Testeo	15	5	75
Ordenador	800	-	800
Servidor	700	-	700
		145	**3.675 €**

Tabla 8-1: Presupuesto del proyecto

Capítulo 9
Bibliografía

[1] Documentación oficial Twitter API. Recuperado de https://developer.twitter.com/en/docs

[4] Josh Constine(19 marzo, 2019). TechCrunch. Twitter cracks down on API abuse, will charge B2B devs [Mensaje en blog]. Recuperado de https://techcrunch.com/2019/03/19/twitter-developer-review/

[5] Laboratorio de Innovación en Humanidades Digitales de la UNED(10 enero, 2019). Cómo usar Freeling en línea. Etiquetador morfológico automático [Archivo de vídeo]. Recuperado de https://www.youtube.com/watch?v=UrLjV3xnV14

[6] Francisco Javier Solans Benedí(18 abril, 2018). Introducción a la librería Twitter4J y OAuth [Mensaje en blog]. Recuperado de https://blog.neodoo.es/2011/04/18/introduccion-a-la-libreria-twitter4j-y-oauth/

[7] MongoDB Driver Quick Start, Oficial Documentation. Recuperado de https://mongodb.github.io/mongo-java-driver/3.4/driver/getting-started/quick-start

[8] Ainhoa Lafuente (1 agosto, 2018). Bases de datos relacionales vs. no relacionales: ¿qué es mejor? [Mensaje en blog]. Recuperado de https://aukera.es/blog/bases-de-datos-relacionales-vs-no-relacionales/

[9] Average Software Engineer Salary in Spain (2020). Recuperado de https://www.payscale.com/research/ES/Job=Software_Engineer/Salary

[10] FreeLing Home Page. Recuperado de http://nlp.lsi.upc.edu/freeling/node/1

[11] Padró, L. (2011). Analizadores Multilingües en FreeLing. *Lingüamática,* 3 (1), pp. 13-20.

[12] Antonio Moreno(17 octubre, 2017). Procesamiento del lenguaje natural ¿qué es? [Mensaje en blog]. Recuperado de https://www.iic.uam.es/inteligencia/que-es-procesamiento-del-lenguaje-natural/

[13] Laíno, M. (2015). *Integración de Herramientas para Procesamiento de Lenguaje Natural* (Proyecto de Grado, Universidad de la República) Recuperado de https://www.fing.edu.uy/inco/grupos/pln/prygrado/Informe_libPLN.pdf7

[14] Torrijos, C. (17 enero, 2019). La lingüística computacional, el campo donde se unen las ciencias y las letras [Mensaje en blog]. Recuperado de https://retina.elpais.com/retina/2019/01/15/tendencias/1547545169_410011.html

[15] Moreno, A. (17 octubre, 2017). Procesamiento de lenguaje natural ¿qué es? [Mensaje en blog]. Recuperado de https://www.iic.uam.es/inteligencia/que-es-procesamiento-del-lenguaje-natural/

[16] Mistapotta (15 febrero, 2018). *Make your own Twitter Bot with twitter4j API* [Archivo de video]. Recuperado de https://www.youtube.com/watch?v=kgj3mjclAsM