Universidad Rey Juan Carlos

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA

Curso académico 2015/2016

Trabajo Fin de Grado

Clustering multilingüe de tweets

Autor: Manuel Alejandro Díaz Muñoz

Tutora: Soto Montalvo Herranz

Agradecimientos

Quisiera dar las gracias en esta sección a toda persona que me ha acompañado a lo largo de mi aventura universitaria, gracias de corazón.

A mi tutora Soto por las 3 asignaturas en las que fue mi profesora y por haberme dado la idea y oportunidad de crear este TFG y especialmente, por haber tenido la paciencia de colaborar conmigo en el desarrollo del trabajo que aquí se presenta y en esta memoria.

A mis padres por creer en mí y sin los cuales ni este trabajo ni este viaje hubieran existido, gracias por vuestra paciencia y apoyo incondicional.

Por último a mis compañeros y amigos de clase y de otros grados por haber hecho de mi paso por la universidad una experiencia única, especial e inolvidable.

Resumen

Este documento es la memoria del TFG realizado sobre *clustering* multilingüe de tweets y constituye la culminación de los estudios del Grado en Ingeniería Informática.

En este TFG se pretende crear un sistema capaz de completar la tarea de agrupar tweets por temas escritos en más de un idioma. Además, se miden los resultados y se busca estudiar diferentes opciones para realizar los clusters y obtener una idea de cuales son más y menos apropiadas en cuanto a la calidad de resultados.

Entre otras, algunas opciones de las escogidas para realizar esta tarea han sido la representación *multivectorial* de los tweets, el uso de lógica borrosa para hallar la similitud entre tweets, o diferentes opciones para abordar el reto de agrupar documentos en diferentes idiomas.

Listado de acrónimos

- **API:** del inglés *Application Programming Interface*, Interfaz de Programación de Aplicaciones.
- **CMT:** acrónimo de *Clustering Multilingüe de Tweets.*
- **CLEF:** siglas de la iniciativa *Cross-Language Evaluation Forum*, dedicada a la promoción de sistemas de información con énfasis en el multilingüismo y la multimodalidad de la información.
- **CU:** Caso de Uso.
- **EN:** Entidad Nombrada.
- **FAP:** del inglés, *Fast Affinity Propagation*.
- **HTML:** del inglés, *HyperText Markup Language,* Lenguaje de Etiquetado de Hipertexto.
- **IDF:** del inglés, *Inverse Document Frecuency,* Frecuencia Inversa de Documento.
- **JSP:** del inglés, *JavaServer Pages.*
- **PHP:** del inglés, *PHP Hypertext Pre-processor*, Procesador PHP de Hipertexto.
- **PLN:** Procesamiento del Lenguaje Natural.
- **REST:** del inglés *Representational State Transfer*, Transferencia de Estado Representacional.
- **RQF:** Requisito funcional.
- **RQNF:** Requisito No Funcional.
- **TF:** del inglés *Term Frecuency,* Frecuencia de término.
- **TFG:** Trabajo Fin de Grado.
- **UNED:** Universidad Nacional de Educación a Distancia.
- **VSM:** del inglés *Vector Space Modelling,* Modelo de Espacio Vectorial.

- **URL:** del inglés, *Unifrom Resource Locator*, Identificador de Recurso Uniforme.
- **WAR:** del inglés, *Web Application Archive,* Archivo de aplicación web.
- **XML:** del inglés *eXtensible Markup Language*, Lenguaje de Marcas Extensible.
- **XP:** del inglés *eXtremme Programming*, Programación Extrema.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

En este capítulo se introduce el problema que queremos resolver, los motivos por los que buscamos resolverlo y los objetivos que nos planteamos alcanzar con este proyecto. Al final también se incluye una descripción de la estructura de este documento.

1.1 Contexto

En esta edad en la que nos encontramos, la edad contemporánea no cabe duda que lo que más ha marcado la diferencia son las tecnologías. En los últimos años, la velocidad con la que aparecen estas tecnologías no ha hecho más que aumentar y parece que no va a detenerse ni a frenar, sino más bien todo lo contrario, se va a acelerar el ritmo con el que aparecen nuevas y mejores tecnologías. En este momento, internet se ha convertido en la fuente de información más grande del mundo, con contenido histórico y en tiempo real, y justo aquí Twitter marca la diferencia.

Twitter es una red social con una gran diferencia con respecto a la mayoría, la brevedad de los mensajes y el carácter público de la gran mayoría de ellos, al contrario de lo que ocurre con otras redes sociales, donde se da más importancia a la privacidad.

De todos los mensajes que se envían por Twitter públicamente, se pueden extraer grandes cantidades de información, dado que existen más de 320 millones de usuarios activos que publican más de 500 millones de tweets al día en una gran variedad de idiomas, según los datos encontrados en la página web ExpandedRamblings [1]. Es una cantidad enorme de datos con la que trabajar.

Podemos encontrar ejemplos de trabajos que utilizan Twitter como fuente de información tales como:

- *Sentiment Analysis of Twitter Data* [2], donde se busca clasificar tweets entre positivo, negativo y neutral.
- *Building a Social Dictionary based on Twitter Data* [3], donde se pretende averiguar el significado de palabras de nuevo uso en la red mediante la coocurrencia entre las palabras y los Hashtags.
- *Named Entity Recognition in Tweets* [4], donde se trabaja en mejorar el degradado rendimiento que tienen los reconocedores de Entidades Nombradas que se utilizan habitualmente en otro tipo de textos al utilizarlos sobre los tweets.
- *Twitter Trending topic Classification* [5], donde se desarrolla un método para clasificar los *trending topic* [1] de Twitter.

1.2 Motivaciones

La gran cantidad de tweets que se envían al día y la cantidad de idiomas en los que los encontramos genera un problema de sobrecarga de información para los usuarios. Esto hace que resulte cada vez más atractiva la idea de poder agruparlos por temas automáticamente de alguna manera sin las barreras que nos crean los idiomas.

La idea de agrupar tweets puede ser muy útil para usuarios que vivan en zonas donde exista más de un idioma en la región o que manejen más de un idioma en su vida cotidiana. Un algoritmo de *clustering* de tweets multilingüe haría posible acceder a tweets sobre el tema que les interese rápidamente, evitando hacer una búsqueda para cada idioma.

[1] *Trending Topic*: es una expresión proveniente de Twitter que significa tema de los más relevantes del momento en la red social.

El *clustering* es una tarea que consiste en la creación de agrupaciones de objetos partiendo de un conjunto de objetos inicial, de tal manera que los objetos de cada grupo (llamado cluster) sean más similares entre ellos que con los de otros grupos.

Además del interés del agrupamiento o *clustering* de tweets para los citados usuarios multilingües, agrupar los tweets con independencia del idioma también puede ser muy útil para empresas de carácter internacional. Estas empresas pueden querer saber opiniones y comentarios acerca de su marca y productos sin limitarse a los comentarios dirigidos directamente a ellos y así, teniendo acceso a más y mejor información sobre su reputación.

"La determinación de temas principales en una colección de tweets puede ser un primer paso útil para organizarlos y abordar el problema de la visualización de datos, recuperación de información semántica (no basada en búsquedas con palabras clave), extracción de información, detección de usuarios con los mismos intereses, recomendación de hashtags, etc." [6]

La idea de agrupar temáticamente tweets multilingües termina siendo una tarea muy atractiva, ya que puede servir como paso previo de otras tareas más complejas como detección de eventos y emergencias en el mundo real, detección de comunidades de interés, recomendaciones y averiguación de los *trending topic* entre otras.

1.3 Objetivos

En este proyecto se pretende desarrollar un sistema de *clustering* de tweets multilingüe, tratando de explotar de alguna forma las características inherentes a los tweets a la hora de compararlos.

El sistema, a partir de una colección de tweets de entrada escritos en idiomas diferentes, obtendrá un conjunto de clusters de tweets, donde todos los tweets de un

mismo cluster compartirán una temática. Un cluster puede contener tweets en diferentes idiomas o en un único idioma.

De forma más particular, como subobjetivos podemos plantear los siguientes:

- Se plantearán diferentes estrategias en el manejo de la información multilingüe, valorando la posibilidad o no de traducir y, en caso de hacerlo, valorando cuándo y qué se traduce.
- Se plantearán diferentes estrategias a la hora de representar los tweets. Se analizará qué puede ser más o menos importante dentro del contenido de los tweets y si es mejor una representación tradicional para el *clustering* de tweets o representaciones novedosas, basadas en las características propias de los tweets. Se evaluará qué representación logra mejores resultados de *clustering*.
- Se utilizarán todas las tecnologías y herramientas de software de código abierto que resulten más adecuadas al problema que queremos resolver.
- Se procurará que el coste en tiempo de la solución propuesta sea razonable, teniendo en cuenta que serán necesarios diferentes procesamientos sobre los tweets.
- El diseño y la propuesta final deben ser fácilmente adaptables para funcionar con colecciones de tweets con más idiomas o con idiomas diferentes a los presentes en las colecciones que se van a utilizar en este trabajo.

Capítulo 2

Estado del arte

En este capítulo haremos un recorrido estructurado por las tecnologías, estudios y documentos que desempeñan una labor similar o relativa a este proyecto. El objetivo es poder compararlos, comprender cómo se han hecho las cosas hasta ahora en el contexto de este trabajo y establecer similitudes y diferencias con respecto a nuestra propuesta inicial para guiarnos a la hora de tomar decisiones de desarrollo.

En la revisión de los trabajos que hacen *clustering* de tweets nos hemos fijado en varios aspectos principales. En aquellos documentos donde aparecía disponible, nos hemos fijado en cómo representan los tweets, qué algoritmos de *clustering* utilizan y en cómo se las han ingeniado para medir las similitudes entre los tweets.

2.1 Trabajos consultados

A continuación se presentan los diferentes trabajos del estado del arte revisados:

1. *Unsupervised topic discovery in micro-blogging networks* [6]: en este artículo se detalla una nueva metodología automática no supervisada para la detección de temas en redes de micro-blogging. Esta metodología está basada en la expansión de Hashtags a términos en la red y la agrupación de esos términos donde la semántica tiene un papel central. Los términos con los que se extendían los tweets son extraídos de WordNet y Wikipedia. Por cada Hashtag de los tweets, se accede a estas webs para buscar información semántica que permita relacionar tweets más allá de las coocurrencias en el vocabulario. Se crean clusters de Hashtag con las expansiones semánticas de éstos, y las agrupaciones resultantes se utilizan para determinar el cluster de cada tweet. La similitud entre los Hashtags se averigua con una función creada para este trabajo que se

sirve de las expansiones semánticas. Emplea un algoritmo jerárquico que no necesita saber el número de clusters para crear las agrupaciones de Hashtags.

2. ***Clustering Twitter Feeds using Word Co-ocurrence*** [7]: aquí se presenta un método para agrupar tweets que agrupa o *clusteriza* las palabras halladas en los tweets usando la coocurrencia como medida de similitud. Las palabras son filtradas para eliminar stopwords y son *stemizadas* para reducir el vocabulario de trabajo. Se crea una matriz de similitud de palabras donde se utiliza como medida de pesado el número de par de tweets en los que aparece, y esa matriz se utiliza con una modificación del algoritmo de K-Means para obtener los clusters de palabras. Para obtener los clusters de tweets, por cada tweet se escogerá la agrupación de palabras que más palabras del tweet contenga y se crearán los nuevos clusters.
3. ***Classification of tweets via clustering of Hashtags*** [8]: en este documento se habla de dos métodos para ayudar en la extracción de información de tweets. El primero es una técnica para *clusterizar* Hashtags en grupos usando una combinación de coocurrencia, *clustering* de grafos y similitud textual. El segundo método emplea al primero para *clusterizar* Hashtags y con estos clusters clasificar tweets basándose en su contenido usando una combinación de técnicas de *clustering* y clasificación y técnicas de reducción de dimensionalidad. En resumen, en este trabajo los únicos datos necesarios para *clusterizar* tweets son sus Hashtags y el resto de información es descartada.
4. ***Topical clustering of tweets*** [9]: aquí se presenta un estudio sobre clasificación automática y clasificación de mensajes de Twitter inspirado en el enfoque tomado por servicios como Google News, donde el número de clusters es conocido ya de antemano. En este proyecto se utilizan todas las palabras que contienen los tweets para representarlos. Cada una de ellas es tokenizada, pasada a minúsculas y eliminada si aparece menos de 5 veces en total en todos los tweets. Además, de las URL se extrae

información para ampliar el contenido de los tweets. Se utilizan tanto algoritmos de *clustering* no supervisados como supervisados en este trabajo. En el caso del *clustering* no supervisado, se utiliza el algoritmo K-Means con una matriz de similitud para la que se emplea VSM *(Vector Space Modelling)* y la medida de pesado TF-IDF (*Term Frecuency – Inverse Document Frecuency*). Para el *clustering* supervisado se utiliza el clasificador Rocchio que emplea VSM para la representación de los tweets.

5. ***Event detection in Twitter using Aggressive Filtering and Hierarchical Tweet Clustering*** [10]: en este documento se presenta una técnica empleada para el SNOW Data Challenge 2014 [11]. Al ser un trabajo enfocado en la detección de eventos, se utilizan una serie de filtros sobre los tweets para descartar tweets que por estructura, contenido o longitud no pueden indicar ningún evento. La técnica creada emplea filtrado agresivo de tweets basado en longitud y estructura, combinado con un algoritmo de *clustering* customizado que no necesita conocer el número de clusters y con una clasificación de los clusters resultantes para detectar eventos sucedidos en el mundo real. Con los tweets que superan los filtros, se emplea todo su contenido para representarlos exceptuando los *stopwords*, URLs y menciones a otros usuarios. La similitud entre cada par de tweets se determina con la medida coseno (*cosine measure*) [33].
6. ***Efficient Clustering of Short Messages into General Domains*** [12]**:** en este artículo se presenta un método para la creación de clusters de tweets según su temática, utilizando un algoritmo propio derivado del conocido K-Means. Se presentan dos maneras de representar los tweets. Primero se utiliza el modelo de "bolsa de palabras" (*bag of words*) con los valores TF-IDF de las *n* palabras más comunes en la colección de documentos y como segundo método utiliza únicamente los Hashtags y la coocurrencia de éstos.
7. ***New Clustering proposals for Twitter based on the Louvain algorithm*** [13]: en este trabajo se aborda el problema de agrupar tweets con el algoritmo de *Louvain*. Este algoritmo no requiere que se especifique el

número de clusters resultantes pero necesita un parámetro que le indique la granularidad de los mismos, y aquí que se exploran las posibilidades que ofrece alternar la granularidad de las agrupaciones. Como medidas de similitud entre tweets, en este trabajo se emplea el índice *Jaccard (Jacard Index),* variaciones de éste, la similitud coseno *(cosine similarity)* y la distancia Euclídea. Con todas las medidas de similitud, exceptuando únicamente el índice *Jaccard*, se utiliza como medida de pesado TF-IDF. Se presentan dos métodos diferentes para representar tweets y en ambos se emplean todos los términos del tweet además de extensiones del contenido. En el primero de los métodos, los tweets se expanden con la parte importante del contenido HTML de los links que aparecen en los tweets. En el segundo método se utiliza Wikipedia para asignar una página a cada término y después encontrar la similitud entre las páginas y tener la similitud entre términos, permitiendo relacionar documentos más allá de la coocurrencia entre términos.

8. ***Real-Time Classification of Twitter Trends*** [14]**:** en esta memoria se expone un estudio sobre los disparadores de las tendencias o *trendings* en Twitter y un trabajo sobre la clasificación de estas tendencias. La herramienta implementa un eficiente método de clusterizar inmediatamente *Trending topics* sin la ayuda de datos externos. Utiliza diagramas de distribución de datos con todo el contenido de los tweets para averiguar a qué categoría pertenecen las tendencias. En este desarrollo se utiliza VSM para realizar las evaluaciones de los clusters y estudiar la utilidad de los servicios que implementan.
9. ***On the Difficulty of Clustering Company Tweets*** [15]: el objetivo de este trabajo es presentar y comparar un número de enfoques diferentes basados en *clustering* que determinan si un tweet se refiere o no a una compañía. El trabajo se centra en la ambigüedad de algunos términos que pueden hacer creer que en un tweet se menciona una empresa cuando en realidad no ocurre. En la creación de los clusters, se utiliza el algoritmo K-

Means con una representación de los tweets con pesado TF-IDF y como medida para el cálculo de la similitud entre tweets la medida coseno.

10. ***Improving Web Page Clustering Through Selecting Appropiate Term Weighting Functions*** [16]: este documento no incluye un estudio de *clustering* de tweets, pero se menciona aquí porque presenta una evaluación de diferentes funciones de pesado a la hora de hacer *clustering* de páginas web. Las medidas de pesado que se han extraído de esta memoria para nuestro trabajo son:
    - TF, frecuencia del término o *Term Frecuency*
    - IDF, frecuencia en documentos invertida o *Inverse document Frecuency*
    - TF-IDF, multiplicación de las medidas TF e IDF

2.2 Conclusiones extraídas

Los documentos que se han presentado son algunos de los más importantes que se consultaron para conocer el estado del arte previo al desarrollo de nuestra propuesta. Tras la lectura de todos ellos podemos sacar una serie de conclusiones.

Lo primero que se deduce es que el *clustering* de tweets es una tarea que a día de hoy aún se puede considerar como un problema abierto, ya que existen muchas maneras de enfocarlo y los resultados son muy variados pero no lo suficientemente buenos como para que no se deba explorar más. El *clustering* multilingüe de tweets se encuentra incluso en un estado más “inicial” y los avances son menos visibles, así que parece necesaria una profunda investigación para hallar la mejor estrategia para enfocar el problema de trabajar con diferentes idiomas en los tweets.

De forma más particular, de los documentos consultados también se puede deducir que la gran mayoría utilizan algoritmos donde el número de clusters es conocido y las ventajas de no tener un número fijo de clusters están aún por explotar.

Existe una notable cantidad de documentos en los que el enfoque a la hora de hacer *clustering* de tweets sugiere que la cantidad de información que está en los

tweets no es suficiente y es necesaria una extensión de información mediante algún método. Se puede observar en la gran mayoría de documentos revisados que los tweets son representados mediante vectores de acuerdo al contenido del tweet y sus extensiones si existieran.

La intención del presente trabajo es proponer un sistema para hacer *clustering* multilingüe de tweets donde no se necesite conocer a priori el número de clusters final, ni tampoco se utilice información adicional al propio contenido. Según algunos trabajos, la información que nos dan los Hashtag es la más importante y se tendrá esto en cuenta para crear nuestra aplicación.

Capítulo 3

Metodologías y Tecnologías

En este capítulo se detalla la metodología escogida para el TFG y cómo se ha desarrollado, además de la variedad de tecnologías que se han utilizado y con qué fines.

3.1 Metodología escogida: Programación extrema

La metodología escogida para el desarrollo ha sido la Programación Extrema, o como se conoce internacionalmente *eXtremme Programming* (XP), que es un enfoque de la *Ingeniería del Software*, formulado por Ken Beck [17], considerado como el más destacado de los procesos ágiles.

Al tratarse de un Trabajo de Fin de Grado, se simulará que el cliente es el tutor y el alumno el desarrollador único.

Como propone la metodología XP, se ha intentado llevar un diseño simple durante el desarrollo, con un enfoque incremental en el que se hacían paso por paso las etapas y con reuniones con el cliente donde se detallaban y repasaban los requisitos de la aplicación.

3.2 Tecnologías

Las tecnologías que se han empleado para el desarrollo de la aplicación capaz de crear clusters de tweets y el servicio y aplicación web son:

- **Java** [18]: es el lenguaje de programación que se ha utilizado principalmente para el desarrollo de la aplicación de *clustering* de tweets. Es un lenguaje de Programación Orientado a Objetos, de los más famosos del mundo, con APIs y software comunitarios que dan cobertura a prácticamente todas las necesidades

de la investigación de este trabajo, así como su extensión con la creación de servicios y aplicaciones web con facilidad.

- **Python** [19]: es un lenguaje de programación interpretado, multiparadigma, que utiliza tipado dinámico y es multiplataforma. Es invocado en este proyecto desde Java para que ejecute un código escrito en Python relacionado con el algoritmo de *clustering* de *Louvain*.
- **PHP** [20]: éste es un lenguaje de programación de uso general, originalmente para el desarrollo web de contenido dinámico ejecutado en el lado del servidor. Aquí es empleado para invocar una API necesaria en la invocación de un servicio web.
- **JSP** [21]: es una tecnología Java que ayuda a crear páginas web dinámicas con HTML o XML y ejecutando código Java. Son utilizadas aquí para el desarrollo de la aplicación web que hace uso del servicio web, para permitir utilizar el código implementado en esta investigación a cualquier persona con acceso a un navegador web.
- **HTML** [22]: son siglas que significan *HyperText Markup Language*. Es el lenguaje predominante para la construcción de páginas web, describiendo la estructura y el contenido en formato de texto así como permitiendo referencias a otros objetos en la web.
- **XML** [23]: siglas representantes de *eXtensible Markup Language*, y es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web (W3C). En este proyecto se ha utilizado para crear ficheros de configuración que serán utilizados por la aplicación que agrupa tweets.
- **Java RESTful services** [24]: es una API del lenguaje de programación Java que proporciona soporte en la creación de servicios web de acuerdo con el estilo arquitectónico REST (*Representational State Tranfer*). Se ha utilizado para desplegar la aplicación de *clustering* como servicio web.
- **NUSOAP** [25]: es una API del lenguaje de programación PHP que permite implementar y consumir de manera sencilla servicios web en PHP. Se usa en esta aplicación en conjunción con un Script PHP que invocan un servicio web desarrollado con NUSOAP.

- **StanfordNLP** [26]: es una API escrita en varios lenguajes de programación, que provee un conjunto de herramientas de análisis del lenguaje natural. En este proyecto se utiliza la API escrita en Java para varias tareas a lo largo del proceso de *clustering*.
- **OpenNLP** [27]: es una librería de la fundación Apache Software Foundation, que provee un conjunto de herramientas para procesar texto en lenguaje natural. Tiene la misma finalidad que la tecnología mencionada anteriormente, *StanfordNLP*, pero demostró ser menos efectiva para analizar tweets. Se decidió optar para todas las operaciones de PLN por *StanfordNLP* para maximizar la calidad de resultados.
- **Normalizador de tweets** [28]: es una tecnología ofrecida en forma de aplicación y servicio web que nos permite normalizar tweets, es decir, corregir sus errores gramaticales para facilitar el análisis del texto. Es un servicio web para el que hace falta emplear NUSOAP y PHP para invocarlo pero que funciona muy bien.
- **Language Detection Library for Java** [29]: es una API escrita en Java alojada en Github para la detección del idioma de textos. Es código abierto a la comunidad y permite a cualquier persona que lo necesite implementar detección de idiomas fácilmente. Detecta 47 idiomas distintos con un alto porcentaje de aciertos.
- **Translator API for Yandex** [30]: es una API escrita en Java y publicada en Github que da acceso al traductor Yandex, con traducciones entre más de 60 idiomas tanto para palabras sueltas como para frases completas. Es un servicio de pago con el que tras pagar la cuota, se te otorga una clave que se incluye en la API y hace que funcione el traductor. No obstante, es posible obtener una clave gratuitamente para estudiantes.
- **APRO** [31]: es una API escrita en Java con una implementación del algoritmo de *clustering, Fast Affinity Propagation*, empleado en este trabajo para realizar los clusters de tweets.
- **NetworkX** [32]: es un paquete Python para la creación, manipulación y estudio de la estructura, dinámicas y funciones de redes complejas. Es empleada por algunas de sus estructuras de datos que son utilizadas en un código Python que implementa el algoritmo de *clustering* de *Louvain*

Capítulo 4

Propuesta de *clustering* multilingüe de tweets

Este capítulo está destinado a hablar de la propuesta planteada en este trabajo, presentando los problemas asociados con el procesamiento de tweets, y qué soluciones se han adoptado ante dichos problemas.

4.1 Problemas y soluciones para procesar los tweets

Un tweet es un texto publicado en la red social Twitter. Los tweets son textos cortos, tienen una restricción de 140 caracteres de longitud. Debido a la naturaleza de Twitter, servicio de *microblogging*, los usuarios usan acrónimos, cometen errores ortográficos, usan emoticonos y otros elementos con carácter especial.

Estos mensajes cortos contienen menciones a otros usuarios de la red social, con su nombre de usuario precedido por el carácter '@'. Referirse a un usuario en concreto de esta manera automáticamente alerta al usuario de la mención. Un ejemplo de un tweet con una mención a una cuenta de Twitter (@michaeljackson) es:

"¿Ya viste estas raras fotos de @michaeljackson "El Rey del Pop"?"

También los tweets contienen Hashtags, que son cadenas de texto sin espacios precedidas por el carácter '#'. Los usuarios normalmente usan estas etiquetas para especificar el tema de su mensaje o unirse a conversaciones sobre el tema que representa el Hashtag y así aumentar la visibilidad del mensaje. Un ejemplo de un tweet con un Hashtag (#FreddieMercury) es:

"<<No voy a ser una estrella, voy a ser una leyenda>> Recordando al gran #FreddieMercury"

Nuestro objetivo es agrupar estos mensajes cortos por temas, abarcando tweets de más de un idioma. Para que un algoritmo de *clustering* pueda saber si dos tweets se refieren a la misma temática, tiene que ser capaz de compararlos y para ello el algoritmo tiene que manejar un vocabulario común, es decir, a partir de un mismo vocabulario hay que generar una representación común. Solo cuando tengamos representaciones comunes del grupo de tweets que queremos agrupar, seremos capaces de comparar cada par de tweets y encontrar la similitud entre ellos.

El primer y más evidente problema que nos aparece para obtener un vocabulario común es la diferencia de idiomas. Si los tweets están en distintas lenguas, contendrán palabras diferentes y no podremos obtener un vocabulario común a todos los tweets, es necesaria una solución.

La primera solución y la más obvia es la de traducir el contenido de los tweets, si pasamos todos los tweets a un único idioma podremos encontrar un vocabulario común para todos los tweets y por lo tanto, seremos capaces de compararlos entre sí.

Ésta no es la única solución que se ha desarrollado en esta investigación, hay una segunda posible solución. En lugar de traducir todos los tweets a un único idioma, este nuevo planteamiento propone crear clusters por idioma, es decir, coger los tweets de cada idioma y agruparlos separadamente, y una vez se tengan los clusters monolingües, si existe similitud entre clusters en idiomas diferentes, éstos son fusionados creando clusters multilingües.

Existen más problemas a la hora de procesar tweets, tras la problemática con la diferencia de idiomas, lo siguiente más llamativo es la gran cantidad de errores gramaticales que se cometen en estos mensajes, ya que pueden ser escritos por cualquier persona y que además están acotados en longitud (lo que genera eliminar caracteres para expresar más en menos espacio). El problema con los errores gramaticales es que entorpecen la labor de las herramientas de PLN y hacen que el

vocabulario común de todos los tweets sea más disperso. Un par de ejemplos de tweets con errores típicos serían:

> *"Ezo les pasa x no darse cuenta antes de komo es su parejaa"*
>
> *"U ever met someboy of twitter than ya meet up and they got the same alphet they had on their profile pic lol"*

Estos tweets son una representación de lo que ocurre en todos los idiomas en los mensajes de Twitter, se acortan palabras, se cometen faltas de ortografía y se prolongan algunos caracteres para dar énfasis. Las versiones corregidas de estos tweets que serían ideales para las herramientas de PLN son:

> *"Eso les pasa por no darse cuenta antes de cómo es su pareja"*
>
> *"You ever met some boy of twitter then you meet up and they got the same alphabet they had on their profile pic lol"*

La tarea encargada de "arreglar" textos y eliminar los errores gramaticales es conocida como Normalización, y en nuestro caso, existen herramientas que ya están preparadas para normalizar tweets. Para algunos idiomas como el inglés, estas herramientas abundan en comparación con otros idiomas donde escasean o incluso no existen. Para esta tarea, se ha escogido una herramienta de normalización especializada en tweets, proporcionada por la UNED en forma de servicio web. De esta herramienta se espera que, mediante la corrección de los tweets, reduzca el vocabulario común y cree más coincidencias entre palabras de unos tweets con otros. Corregir los tweets no es una tarea fácil y por ello, seguro que se cometen errores en la normalización pero si se utiliza una herramienta con resultados medianamente aceptables, se mejoraría los registros de la calidad de los clusters.

Para obtener el vocabulario común de todos los tweets que nos permita compararlos y hallar la similitud entre ellos, además de lo anteriormente expuesto, se han hecho una serie de operaciones sobre el contenido de los tweets con la intención de lograr el mejor resultado de *clustering* posible:

- Identificación de Entidades Nombradas (ENs): la identificación de ENs en textos es una tarea conocida en el mundo de PLN, pero en los tweets puede resultar incluso más útil que en otro tipo de textos dada la poca información que contienen por su longitud. Por eso, se extraen las ENs y se almacenan todas las encontradas para ser añadidas al vocabulario común de tweets. Una EN puede ser una organización, un lugar o una persona física entre otras cosas.
- Lematización: la lematización es un proceso lingüístico que consiste en, dada una forma flexionada de una palabra, hallar el lema correspondiente. El lema es la forma que por convenio se acepta como representante de todas las formas flexionadas de una misma palabra. Por ejemplo, en el caso de los verbos, el lema de cualquiera de sus expresiones sería la forma infinitiva, en los sustantivos, el lema sería su forma en masculino singular. Este proceso nos puede ayudar en el *clustering* de tweets reduciendo las dimensiones del vocabulario común, ya que si nos encontramos ante una palabra que se puede lematizar, no incluiremos sus diferentes flexiones, sino que únicamente añadiremos el lema.
- Filtro en los datos para el vocabulario común: el contenido de los tweets es muy completo en el sentido de que incluye palabras de todas las categorías gramaticales, además de los elementos propios de los tweets (Hashtags y menciones a usuarios). Se ha determinado para este trabajo que de todo el contenido de los tweets, únicamente se utilizarán los Hashtags, las menciones a usuarios, las ENs, los sustantivos, los adjetivos y los verbos. Cualquier otra palabra que se encuentre en el contenido de un tweet, será descartado. Esta decisión se tomó porque se consideró que cualquier otro dato sería menos relevante que los seleccionados a la hora de relacionar temáticamente los tweets entre sí.
- Eliminación de palabras con una sola aparición: a la hora de encontrar similitud entre documentos, no ayuda en nada incluir al vocabulario común datos que solo aparecen en un tweet, por lo que si un dato solo aparece una vez será eliminado del vocabulario común de todos los tweets sin importar su tipo.

Sobre la propuesta anterior de procesamiento de los tweets se hicieron diferentes pruebas de *clustering*, comprobando que se podían incluir mejoras en este procesamiento para tratar de obtener un vocabulario que ayudase a agrupar mejor los tweets.

Se propuso entonces un análisis ampliado de los tweets, que incluye una serie de operaciones sobre los distintos tipos de datos considerados en los tweets. Las operaciones fueron testeadas varias veces y se demostró que sí que influyen positivamente en los resultados finales de todas las pruebas realizadas.

Las operaciones que emplea el análisis extendido son las siguientes:

- **Hashtags dentro de Hashtags:** Se observó durante las pruebas que en las colecciones de tweets suele haber Hashtags que no son idénticos a otros, pero sí puede ocurrir que uno contenga al otro y coincide que la temática de sus tweets es la misma. Un ejemplo:

  *"una historia sin fin, por muchos años mas **#Ford** #Falcon"*
  *"Hoy ví al amor de mi vidaa **#fordmustang**. Para que quiero un novio si con uno de esos estaría sobrada!!"*

  Ambos tweets hablan sobre modelos de coches de la marca Ford, pero sus Hashtags no son el mismo por lo que resulta más complicado relacionar temáticamente estos dos tweets. Para solucionar estos casos, se busca en todos los Hashtag y si un tweet T1 contiene un Hashtag H1 y se encuentra en otro tweet un Hashtag H2 contenido en H1, H2 es añadido a los Hashtag que contiene T1.
- **Entidades mejoradas:** Se observó que en determinados casos, existían Entidades Nombradas que una contenía a la otra. Un ejemplo:

  *"La tecnología inteligente se funde con la potencia dinámica. Pásate al cambio automático **BMW** http://www.bmw.es/home/footer/1/glosario/cambio-automatico.html … "*
  *"Elige una vida sin límites con tu **BMW Motorrad** y #MakeLifeARide"*

Las Entidades Nombradas en estos tweets son "*BMW"* y "*BMW Motorrad*", y como tal, son entidades diferentes pero claramente ambos tweets hablan de una temática similar aunque no generen ninguna similitud entre sus tweets. Lo que se ha hecho para estos casos es un análisis sobre las entidades de los tweets donde, cuando tengamos un tweet T1 con la entidad E1, si encontramos en algún otro tweet una entidad que comparta alguna de las palabras con la entidad E1, por ejemplo las entidades "*BMW*" y *"BMW Motorrad"*, la entidad más larga será cambiada por la más pequeña. En nuestro ejemplo, en el tweet que contiene la enitdad *"BMW Motorrad"*, ésta será cambiada por la entidad *"BMW"*.

- Entidades en Hashtags: Uno de los fallos más repetidos que se encontraban en los primeros clusters, antes de incluir esta extensión del contenido de los tweets, era que se creaban clusters de un tema alrededor de un Hashtag y otros alrededor de una entidad, siendo ambos lo mismo (Hashtag y EN) pero diferente tipo de datos. Esto hacía que hubiera más de un cluster por tema cuando en realidad podrían ser un único cluster. Un ejemplo de dos tweets con estas características:

  "La mayoría de los alpinistas que mueren en el ***Everest*** *lo hacen en los últimos 850 metros, por falta de oxígeno."*
  "Ya es posible #viajar como #turista en un #globo aerostático por el ***#Everest****. Inténtenlo http://bit.ly/1ePRTRG"*

  Vemos que ambos tweets comparten temática y el nexo que tienen en común es una Entidad Nombrada y un Hashtag, datos distintos en el análisis original del contenido de los tweets de este trabajo. Lo que hacemos con estos casos es que si tenemos un tweet T1 con un Hashtag H1 y encontramos en otro algún otro tweet una EN idéntica en el contenido, la EN será añadida a las entidades del tweet T1 donde se encuentra el Hashtag.
- Nombres y adjetivos: El último patrón que observamos en el contenido de los tweets y los clusters que se formaban corresponde a los nombres y adjetivos. Este fenómeno se daba en menor medida, pero claramente se podían ver

pares de tweets del mismo tema con Hashtags o entidades que contenían algunos nombres y adjetivos de otros. Esto creaba clusters para más de un tema cuando podrían estar juntos. Un ejemplo con dos tweets:

> *"Más cerca de volver, al lugar donde nace mi verdadera felicidad y tranquilidad* ***#MyHouse****"*
> *"It's too cold to leave* ***my house****"*

Es obvio que ambos tweets hablan de la casa de uno mismo, y la temática es cercana entre ambos tweets pero no comparten ningún dato en el análisis original. Para estos casos, si un tweet T1 contiene un Hashtag H1 y buscando en el resto de tweets, encontramos que hay nombres o adjetivos completos dentro de H1, se añadirá la palabra a la lista de nombres y adjetivos de T1, como en el ejemplo de *"#MyHouse"* y *"house"*. Lo mismo ocurriría si encontramos una EN en lugar de un Hashtag conteniendo un sustantivo o adjetivo.

Estas operaciones de extensión del análisis son en cierto grado "arriesgadas", dado que pueden agregar ruido al contenido de los tweets y generar errores. También se cree que estas operaciones influyen directamente en la granularidad final de los clusters, haciendo que éstos sean de grano más grueso.

4.2 Propuesta de representación de tweets

Una vez tenemos listo el vocabulario común y hemos extendido con nuestro análisis ampliado el contenido de los tweets, debemos representar cada uno de los tweets con respecto al vocabulario que hemos hallado. Se han empleado varias maneras para representar los tweets pero todas han sido representaciones basadas en vectores. Una representación de un documento basada en un vector es muy simple, tenemos para cada documento un vector de tantas posiciones como el tamaño del vocabulario común. Se busca todos aquellos datos que aparecen tanto en el vocabulario común como en el documento, y por cada coincidencia, se marca en el

vector con un número, que representa la importancia del término del vocabulario en el documento. En aquellas posiciones del vector correspondientes a términos del vocabulario que no están en el documento el valor asignado será 0. Este modelo es muy común en la representación de documentos y es conocido como VSM (*Vector Space Modeling*).

4.2.1 Medidas de ponderación

Las unidades de pesado o ponderación que se han propuesto en esta investigación son cuatro, ***·Binario, ·Binario-IDF, ·TF*** y ***·TF-IDF***. Para explicarlas correctamente es necesario conocer qué son las medidas de normalización *TF* e *IDF*.

La medida TF (*Term Frecuency*): mide el número de veces que aparece un dato en el documento. En nuestro caso, esto se traduce en cuántas veces aparece un dato en el tweet. Por ejemplo, la medida TF del término *'go'* será 2 en el siguiente tweet:

"Go Raptors, go for the win!!"

La medida IDF (*Inverse Document Frecuency*): mide la importancia de un término en un documento pero teniendo en cuenta la ocurrencia en el resto de documentos de la colección. Se calcula de la siguiente manera:

$$IDF(t) = \log_{10}(\frac{Nº\ de\ documentos\ de\ la\ colección}{Nº\ de\ documentos\ en\ los\ que\ aparece\ t})$$

Ahora ya podemos definir las medidas de ponderación empleadas:

- **Binario:** el valor de cada término en el vector se representación será 1 si el término del vocabulario aparece en el documento.
- **Binario-IDF:** con esta medida, que un tweet contenga un dato *t* del vocabulario común se representa anotando en la posición correspondiente el resultado de 1 * *IDF (t)*.
- **TF:** cuando se utiliza esta medida, la manera de anotar que un tweet contiene un dato es añadiendo en la posición correspondiente al dato la medida TF de ese dato.

- **TF-IDF:** si utilizamos esta medida de ponderación, lo que se añadirá en la posición del vector correspondiente al dato *t* que queremos marcar, será: *TF (t) * IDF (t)*.

En todos los casos estudiados, cuando un tweet no contiene un dato, se representa en el vector con un 0.

4.2.2 Representaciones de los tweets

El objetivo de la representación de un tweet es utilizarla sobre todo el conjunto de tweets y averiguar con ella el grado de similitud entre cada par de tweets, representado por un número entre 0 y 1 (siendo 0 ninguna similitud y 1 similitud absoluta). Con esas similitudes ya seríamos capaces de utilizar un algoritmo de *clustering* para averiguar los clusters finales. Todas las representaciones creadas para este trabajo emplean vectores y utilizan la distancia coseno [33] para calcular la similitud entre los vectores.

En este trabajo se han presentado dos formas de presentación diferentes: representación con un vector único y representación mediante vectores parciales.

4.2.2.1 Representación mediante un vector único

El primer método de representación se basa en representar cada tweet mediante un único vector, que hemos denominado aquí *Vector Único*. Se toma el vocabulario completo para representar el tweet y se genera un vector con tantas posiciones como datos tiene el vocabulario donde cada posición representa un dato. La importancia se calcula con alguna de las medidas de ponderación vistas anteriormente.

Una vez se han representado todos los tweets, se utiliza la distancia coseno para calcular la similitud entre cada par de tweets y obtener una matriz de similitud. La fórmula de la distancia coseno es la siguiente:

$$\cos\theta = \frac{A \cdot B}{\left||A|\right| \cdot ||B||}$$

La matriz de similitud tendrá una dimensiones *M x M*, donde *M* es el número de tweets que queremos agrupar. Cada posición de la matriz representará la similitud entre los tweets correspondientes al número de columna y número de fila de la posición.

Se ha considerado que era una opción adecuada porque es muy habitual a la hora de representar documentos, y debido al poco contenido de información de los tweets, tener toda la información junta en un único vector puede ser beneficioso.

4.2.2.2 Representación mediante vectores parciales

Este método de representación es más complejo y requiere mayor detalle. Se ha desarrollado para esta representación de tweets dos maneras de trabajar que parten de una idea en común. En el contenido de un documento, ya sea un tweet o cualquier otro, hay algunos datos que son más relevantes que otros a la hora de hacer *clustering*. Por ello, una buena idea es investigar representaciones que nos permitan dar más importancia a unos datos que a otros.

Para crear maneras de representar tweets donde se pudiera dar más importancia a unos datos que a otros, se determinó en momentos tempranos del desarrollo que se deberían crear representaciones *multivectoriales*, es decir, diferentes vectores parciales representando los diferentes tipos de datos del vocabulario común.

Finalmente, lo que se implementó para estas representaciones *multivectoriales*, fueron 4 vectores parciales, cada uno correspondiente a una de estas categorías:

- **Hashtags:** en una categoría independiente tendremos todos los Hashtags del vocabulario común.
- **Entidades:** en esta categoría incluiremos todas las Entidades Nombradas y menciones a usuarios del vocabulario común, que tienen un propósito similar, el de llamar o mencionar una entidad real o virtual.
- **Sustantivos y adjetivos:** aquí incluiremos todos los nombres y adjetivos que se encuentren en el vocabulario común. Se unen en una única

categoría porque van íntimamente relacionados en la mayoría de los casos.

- **Verbos:** tendremos una categoría con todos los verbos del vocabulario común.

Al igual que para la representación de tweets *Vector Único*, se utiliza una medida de ponderación, en este caso para crear los 4 vectores correspondientes a las categorías explicadas. De esta manera, tendremos 4 vectores representando cada tweet.

Con estos 4 vectores, podremos obtener 4 valores de similitud parciales entre cada par de tweets empleando la distancia coseno, que si los fusionamos, nos permitirán obtener un único valor de similitud. Para ello proponemos combinar las similitudes parciales de dos formas diferentes: mediante una *Combinación Lineal* y mediante un *Combinación Borrosa*.

La *Combinación Lineal* obtiene la matriz de similitud trabajando posición por posición, combinando los 4 valores de similitud parciales entre el par de tweets correspondiente para obtener cada valor de la matriz de similitud que queremos crear. Cómo se combinan los valores en *'Combinación Lineal'* es lo que diferencia a esta opción, aquí, para obtener la similitud entre dos tweets partiendo de 4 similitudes parciales, se toman las similitudes parciales y se multiplican estos 4 valores por un valor único correspondiente a su categoría de datos y se suman los resultados de esas multiplicaciones dando como resultado la similitud entre los dos tweets. El valor único correspondiente a cada categoría será distinto, cuanto más alto sea este valor, más importancia le estaremos dando a la similitud entre un componente concreto, y por consiguiente, cuanto más bajo, menos importancia. La suma de los 4 valores correspondientes a las 4 categorías debe ser 1, para que los resultados finales sean siempre entre 0 y 1. Aquí se muestra la fórmula para la *"Combinación Lineal":*

$$\begin{aligned} Similitud_{final} = \; & Similitud_H * Valor_H + \\ & Similitud_E * Valor_E + \\ & Similitud_{NA} * Valor_{NA} + \\ & Similitud_V * Valor_V \end{aligned}$$

Se ejecutará esa operación para cada posición de la matriz de similitud, creando un resultado listo para ejecutar el algoritmo de *clustering* que obtenga los clusters.

Se consideró esta opción como válida y prometedora desde el principio ya que permite combinar similitudes parciales y dando diferente importancia a unos tipos de datos sobre otros de una forma muy directa y sencilla.

Una vez se implementó este método de representación de tweets, se probaron varias combinaciones de valores para los pesos de cada categoría. La búsqueda de estos valores óptimos comenzó con la premisa de que el orden de importancia de las categorías a priori sería:

Hashtags > Entidades > Nombres y Adjetivos > Verbos

Con esa premisa, se hicieron pruebas con distintos valores y la conclusión fue que los valores que mejores resultados obtuvieron son:

- Hashtag: 0.35
- Entidades: 0.35
- Nombres y adjetivos: 0.2
- Verbos: 0.1

La segunda manera de combinar similitudes parciales que se ha creado para este trabajo es la *Combinación Borrosa*. Esta opción emplea un motor de lógica borrosa para averiguar las similitudes absolutas entre tweets, lo cual permite incluir razonamiento humano y conocimiento heurístico mediante una serie de reglas en el proceso.

El objetivo es el mismo que en la propuesta *Combinación Lineal*, hay que combinar similitudes parciales para obtener un único valor de similitud, y formar una matriz de similitud que represente la similitud entre cada par de tweets de la colección. Utilizaremos la lógica borrosa mediante un motor borroso, y con él obtendremos los valores de similitud absoluta.

El motor borroso nos permite incluir conocimiento humano mediante una serie de reglas en la base de conocimiento. El motor es accionado para cada par de tweets, empleando las similitudes parciales para averiguar su similitud absoluta.

El motor borroso que se ha desarrollado para este proyecto tiene como entrada 4 datos numéricos (las similitudes parciales correspondientes a cada par de tweets de acuerdo con los 4 tipos de datos considerados para cada tweet), y como salida un único valor (la similitud final entre el par de tweets). Si pensamos en el motor borroso como una caja negra en la que no podemos observar de qué está compuesto, el resultado sería algo como lo que se presenta en la Figura 4.1.

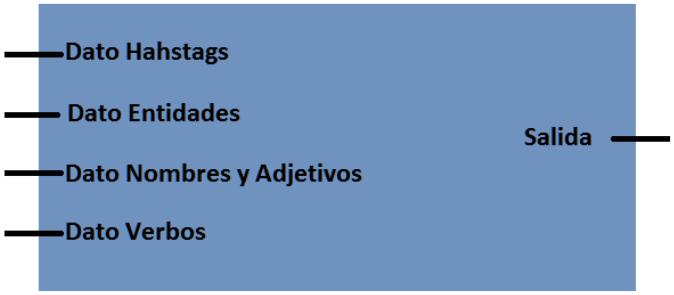

Figura 4.1. Caja negra motor borroso de la investigación.

Si miramos dentro del motor borroso, tal y como se puede ver en la Figura 4.2, encontramos varios componentes que permiten el uso de lógica borrosa.

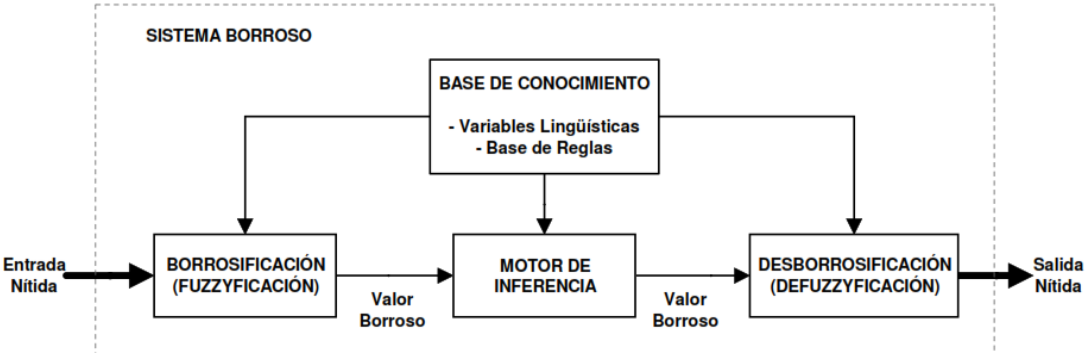

Figura 4.2. Elementos motor borroso genérico.

A continuación se detallará la función de cada componente.

- **Borrosificación**. En esta la primera etapa del proceso de inferencia, toda variable numérica de entrada debe ser borrosificada. Así, el borrosificador o fuzzificador toma los valores numéricos provenientes del exterior y los convierte en valores que puedan ser procesados por el motor de inferencia. Estos valores son los niveles de pertenencia de los valores de entrada a los diferentes conjuntos borrosos en los que se ha dividido el universo del discurso, es decir, a los diferentes conjuntos borrosos de las variables de entrada al sistema. Cada variable borrosa tiene su propia grafica que representa los rangos de sus valores y ésta determinará cuales son los rangos de cada valor borroso y los rangos donde convergen esos valores. En este trabajo se ha optado por que las cuatro variables de entrada tengan 2 valores borrosos, *High* y *Low* porque añadir más hubiera complicado la programación sin tener por qué mejorar los resultados. El rango común entre los dos valores es muy amplio, como se puede ver en la Figura 4.3, con el fin de fomentar que se activen varias reglas al mismo tiempo.

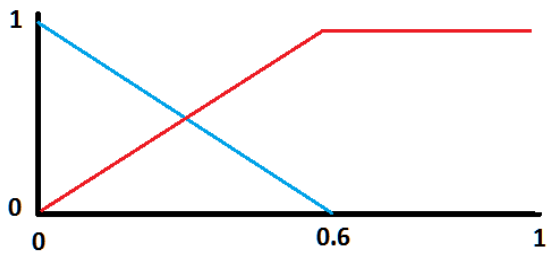

Figura 4.3. Gráfica común a variables borrosas de entrada.

- **Desborrosificación**. En esta parte del motor borroso lo que se pretende es obtener un valor numérico de un valor borroso de la variable de salida. Dado que puede que la variable borrosa de salida tenga activado más de un valor, depende de en qué grado estén activados esos valores para determinar el valor numérico final. Esta variable borrosa de salida tiene 4 valores borrosos posibles. Se establecieron 4 valores de salida para que en el caso de que dos tweets tengan cosas en común, las reglas de determinar similitud entre los tweets sin que resulte siempre en la inclusión de los tweets en el mismo

cluster. La Figura 4.4 representa los rangos de los valores de la variable de salida.

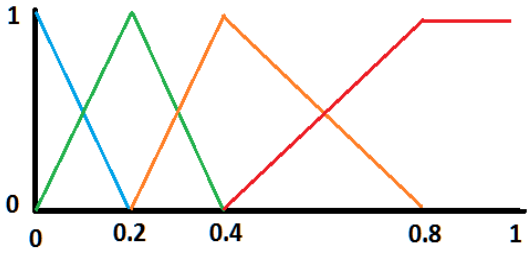

Figura 4.4. Gráfica de la variable borrosa de salida.

El color Azul representa el valor *Low*, el verde *Medium*, el naranja *Strong* y el rojo *High*. Era prioridad desde el principio que hubiera muchos rangos donde la variable tiene dos valores y que nos pudiéramos aprovechar al máximo el beneficio de la lógica borrosa de este motor.

- **Base de conocimiento:** Aquí es donde se encuentran las reglas del motor borroso. Las reglas son del estilo **SI (antecedente) ENTONCES (consecuente)**, donde en el antecedente encontramos las variables borrosas de entrada con uno de sus valores, y en el consecuente la variable borrosa de salida con uno de sus valores. El proceso de obtención de estas reglas partía del conocimiento humano. Las reglas se comenzaron a desarrollar teniendo en cuenta un orden preestablecido de la importancia de las categorías:

  Hashtags > *Entidades* > *Nombres y Adjetivos* > *Verbos*

  La Figura 4.5 presenta la base de reglas propuestas en este trabajo. Como se puede ver, en estas reglas queda reflejado que si ni los hashtags ni las entidades están *High*, no es posible alcanzar el resultado *High* en la salida o si solo los verbos están en *High*, la máxima similitud que se alcanza es *Medium*.

```
1  if DatoHashtags is Low and DatoEntidades is Low and DatoNombresYAdjetivos is Low and DatoVerbos is Low then Salida is Low
2  if DatoHashtags is High and DatoEntidades is Low and DatoNombresYAdjetivos is Low and DatoVerbos is Low then Salida is Strong
3  if DatoHashtags is Low and DatoEntidades is High and DatoNombresYAdjetivos is Low and DatoVerbos is Low then Salida is Strong
4  if DatoHashtags is Low and DatoEntidades is Low and DatoNombresYAdjetivos is High and DatoVerbos is Low then Salida is Strong
5  if DatoHashtags is Low and DatoEntidades is Low and DatoNombresYAdjetivos is Low and DatoVerbos is High then Salida is Medium
6  if DatoHashtags is Low and DatoEntidades is High and DatoNombresYAdjetivos is Low and DatoVerbos is High then Salida is High
7  if DatoHashtags is Low and DatoEntidades is High and DatoNombresYAdjetivos is High and DatoVerbos is Low then Salida is High
8  if DatoHashtags is High and DatoEntidades is High and DatoNombresYAdjetivos is Low and DatoVerbos is Low then Salida is High
9  if DatoHashtags is High and DatoEntidades is Low and DatoNombresYAdjetivos is High and DatoVerbos is Low then Salida is High
10 if DatoHashtags is High and DatoEntidades is Low and DatoNombresYAdjetivos is Low and DatoVerbos is High then Salida is High
11 if DatoHashtags is Low and DatoEntidades is Low and DatoNombresYAdjetivos is High and DatoVerbos is High then Salida is Strong
12 if DatoHashtags is High and DatoEntidades is High and DatoNombresYAdjetivos is High and DatoVerbos is Low then Salida is High
13 if DatoHashtags is High and DatoEntidades is High and DatoNombresYAdjetivos is Low and DatoVerbos is High then Salida is High
14 if DatoHashtags is High and DatoEntidades is Low and DatoNombresYAdjetivos is High and DatoVerbos is High then Salida is High
15 if DatoHashtags is Low and DatoEntidades is High and DatoNombresYAdjetivos is High and DatoVerbos is High then Salida is High
16 if DatoHashtags is High and DatoEntidades is High and DatoNombresYAdjetivos is High and DatoVerbos is High then Salida is High
```

Figura 4.5. Lista de todas las reglas del motor borroso.

- **Motor de inferencia**. El motor de inferencia es el que se encarga de ejecutar la lógica, consultando las reglas de la base de conocimiento y empleando la entrada en su forma borrosa para activar las reglas adecuadas. Con las reglas activadas calcula el/los valor/es de la variable borrosa y en qué grado está activado cada valor, ya que como se trata de lógica borrosa, una variable pueden tener más de un valor simultáneamente.

Cuando tenemos por fin la matriz de similitud absoluta entre todos los tweets con la representación basada en vectores parciales, ya estamos listos para pasar a la siguiente fase e invocar al algoritmo de *clustering*, el cual generará diferentes clusters de la colección de tweets que estemos analizando.

4.3 Algoritmo de *clustering*

Un algoritmo de *clustering* es un procedimiento de agrupación de una serie de documentos de acuerdo con un criterio. Esos criterios son generalmente distancia o similitud. Los algoritmos de *clustering* generalmente funcionan con una matriz de similitud o correlación.

Para este trabajo se han utilizado dos algoritmos diferentes con la intención de averiguar cuál es el más adecuado para la tarea de *clustering* que estamos desarrollando. Las opciones que se han escogido son *Fast Affinity Propagation* (FAP)

y el algoritmo de *Louvain* debido a que son algoritmos no supervisados (no requieren de entrenamiento) y no requieren especificar el número de clusters finales como dato de entrada.

4.3.1 *Fast Affinity Propagation*

Es un algoritmo de *clustering* basado en el concepto de "paso de mensajes" entre *data points*. FAP crea clusters mediante el envío de mensajes entre pares de ejemplos hasta encontrar convergencia. Es un algoritmo de complejidad $O(N^2 T)$ donde N es el número de elementos por agrupar y T el número de iteraciones, y esto hace que sea un algoritmo demasiado complejo para colecciones demasiado grandes pero usable en nuestro problema.

La entrada de este algoritmo es la matriz de similitud de los tweets y como salida nos da los clusters que ha determinado que existen en la colección.

4.3.2 Algoritmo de *Louvain*

El algoritmo de *Louvain* es un algoritmo originalmente creado para la extracción de comunidades de grandes redes. El método ha sido usado con éxito para redes de diferentes tipos con tamaños de hasta 100 millones de elementos [34]. Tiene un par de singularidades este método. La primera es que es un algoritmo no determinista y no garantiza que se extraigan los mismos resultados a partir de una misma entrada en distintas ejecuciones. La otra singularidad es que necesita un parámetro para funcionar, llamado granularidad. Éste especifica el tamaño de los clusters que debe extraer y por lo tanto, influye de manera muy directa en la cantidad y calidad de los *clusters*. El valor de granularidad utilizado para este trabajo se obtuvo mediante una experimentación previa con diferentes valores hasta encontrar el adecuado, 2.5 en nuestro caso.

Igual que el otro algoritmo, su entrada es la matriz de similitud de los tweets y nos responde con los clusters que determina que existen en la colección.

4.4 Propuestas del sistema global

Como se ha descrito a lo largo de este capítulo, se ha desarrollado más de una solución para distintos obstáculos, pero con todas ellas llegamos al mismo destino, con mejores o peores resultados. Algunas de estas opciones generan que el cauce de los datos sufra variaciones significativas en las fases por las que pasan los datos. Las opciones que modifican el cauce de los datos son:

- Traducción: ***Traducción Previa*** o ***Traducción Posterior***
- Representación de tweets: ***Vector Único*** o ***Combinación Lineal/Borrosa*** (ambas tienen el mismo efecto sobre el flujo)

A continuación se presentan diferentes figuras que representan las variaciones que provocan las diferentes opciones en la configuración. En primer lugar, la Figura 4.6 presenta el flujo de datos cuando se utiliza *"Traducción Previa"* y se representan los tweets mediante *"Vector Único"*.

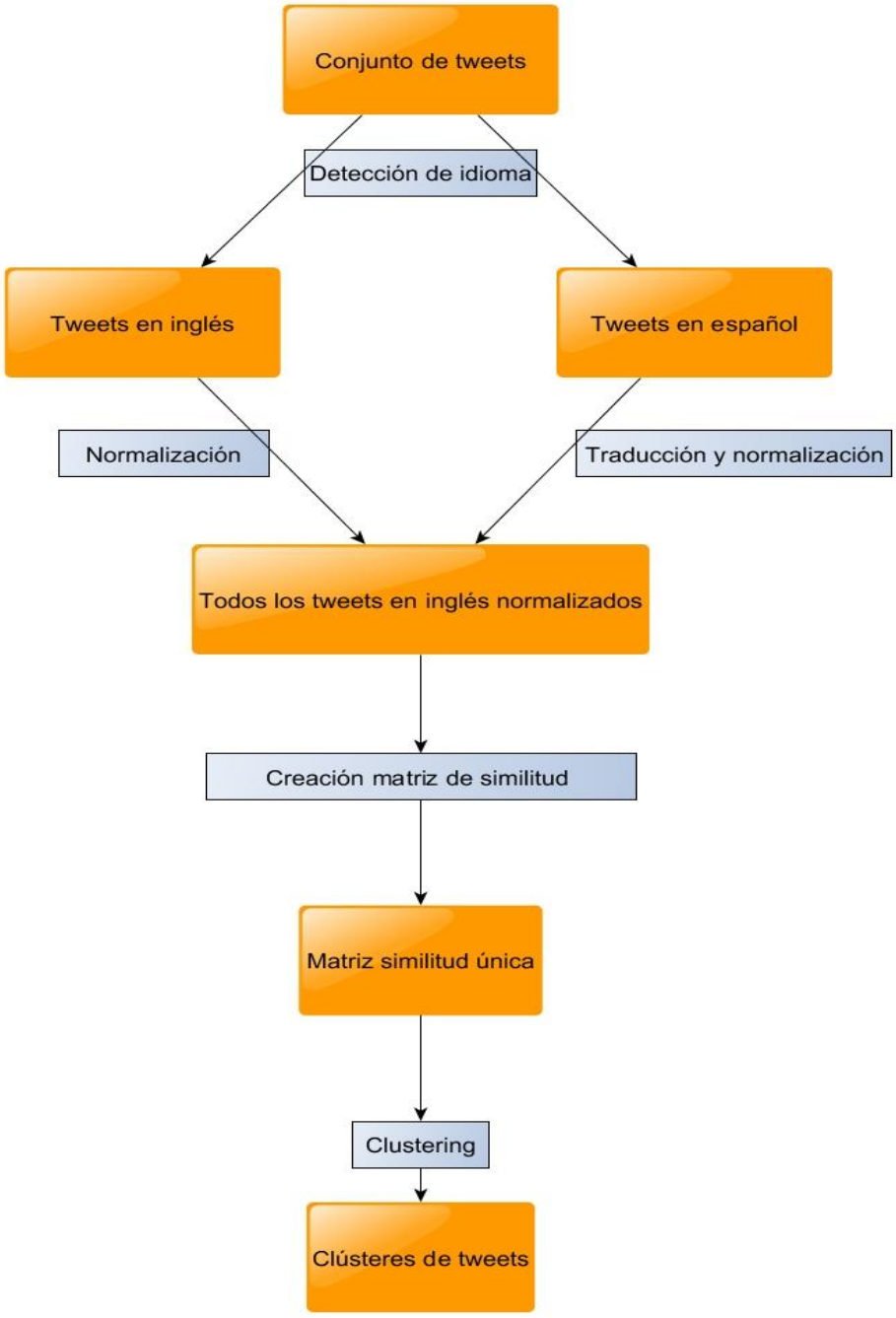

Figura 4.6. Cauce de datos con Traducción Previa y Vector Único.

La Figura 4.7 presenta el flujo de datos cuando se utiliza traducción previa y se representa cada tweet mediante diferentes vectores.

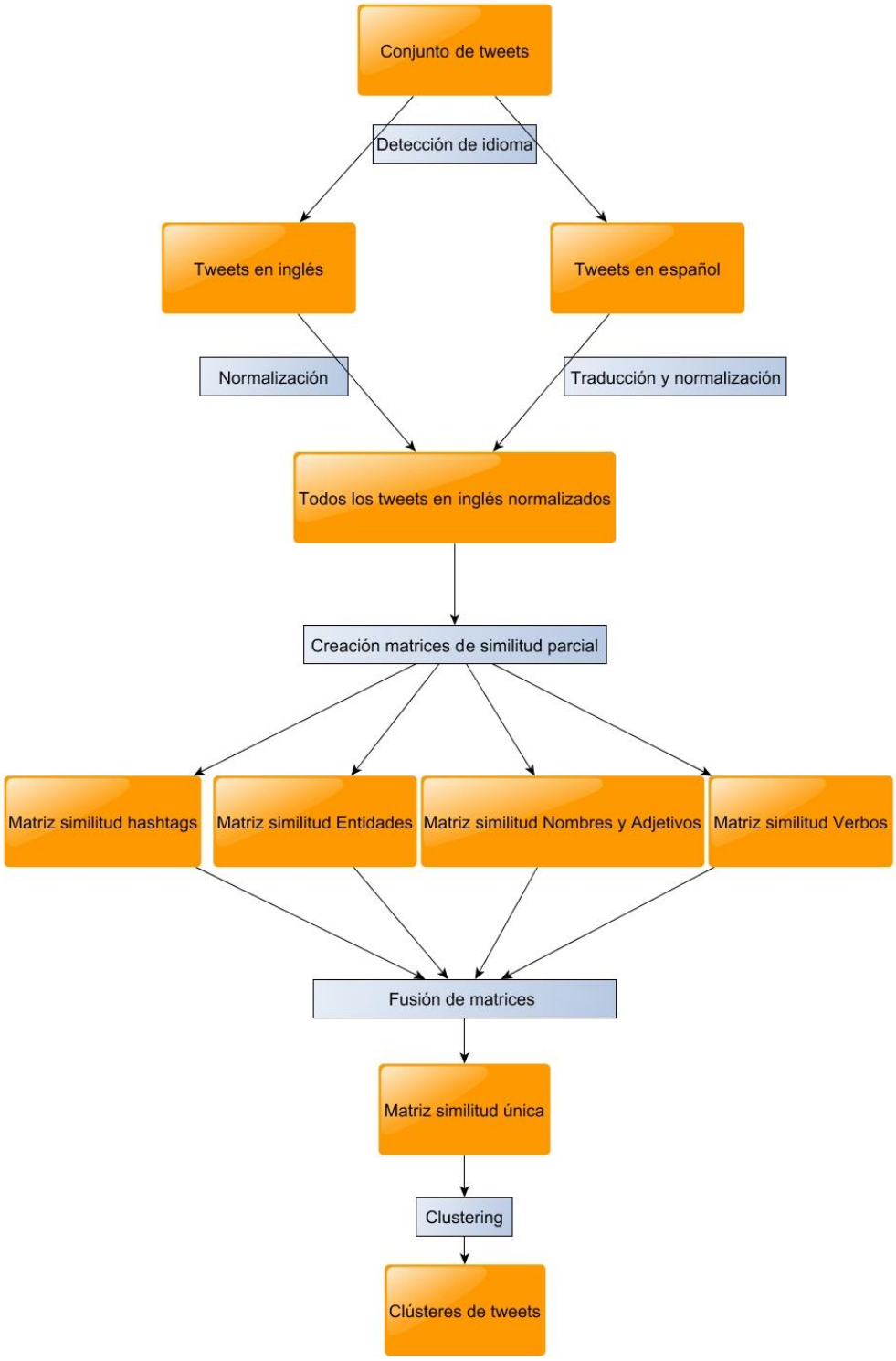

Figura 4.7. Cauce de datos con Traducción Previa y Combinación Lineal/Borrosa.

La Figura 4.8 presenta el flujo de datos cuando se traduce después de haber agrupado de forma independiente los tweets en un mismo idioma y, además, se representa cada tweet con un único vector.

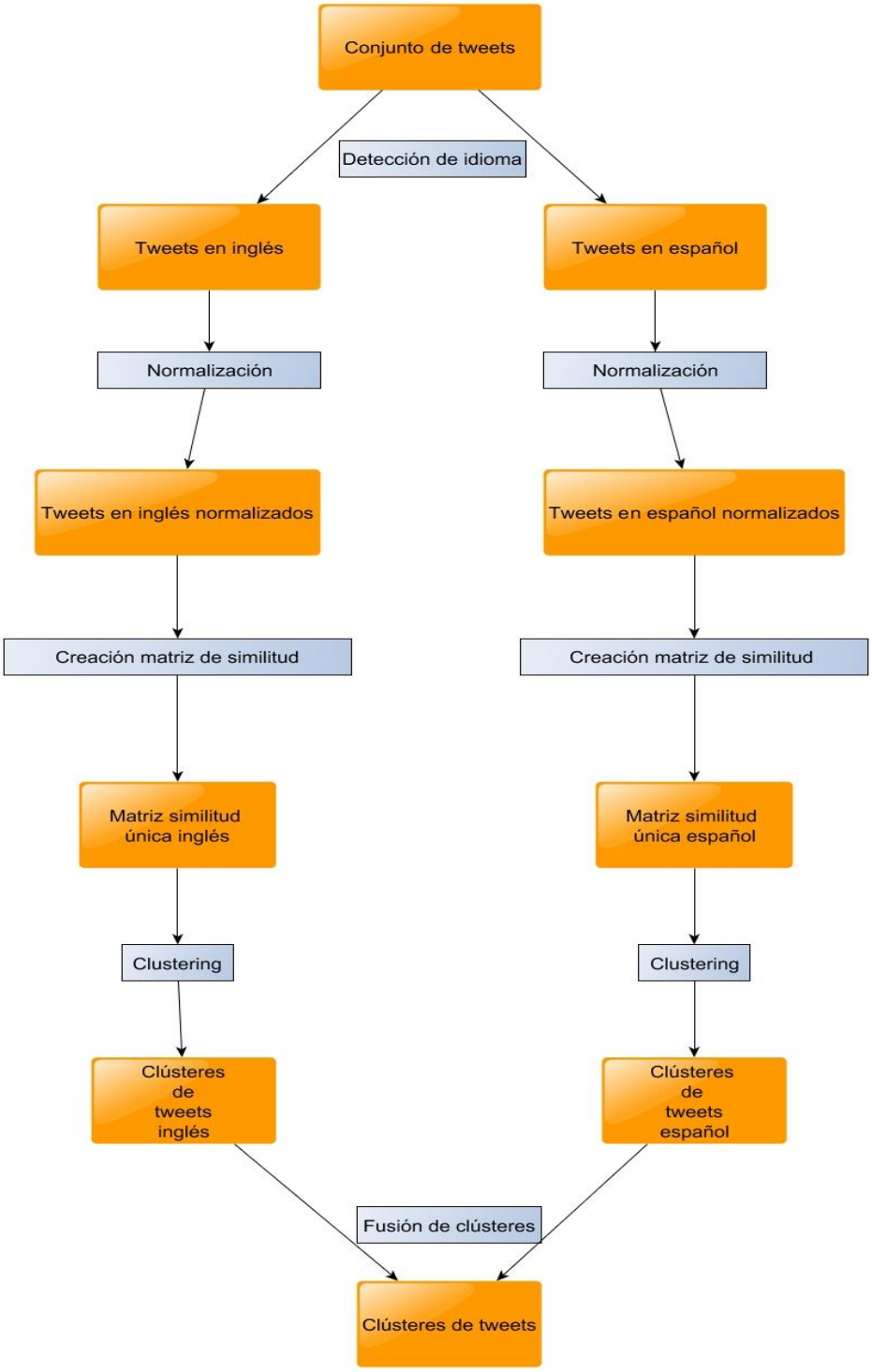

Figura 4.8. Cauce de datos con Traducción Posterior y Vector Único.

Por último, la Figura 4.9 presenta el flujo de datos cuando se traduce después del *clustering* en cada idioma y se representa cada tweet con diferentes vectores.

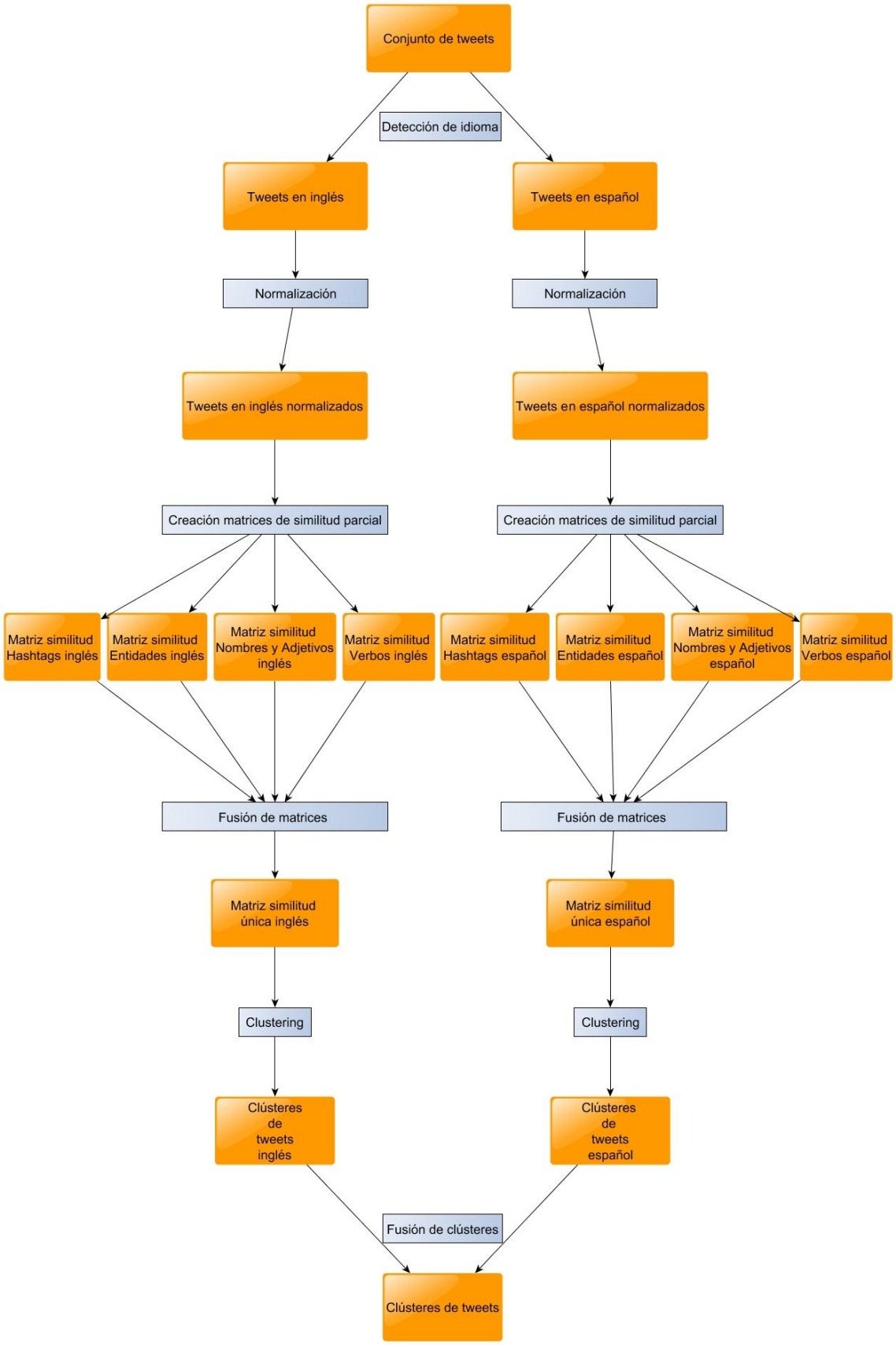

Figura 4.9. Cauce de datos con Traducción Posterior y Combinación Lineal/Borrosa.

En el capítulo 6 se presentan los resultados experimentales de los diferentes flujos presentados.

4.5 Aplicación y Servicio Web

Uno de los requisitos funcionales de la aplicación (se expondrán en el capítulo 5 todos los requisitos) es que debe ser exportado como servicio web. Consideramos que esta idea era adecuada porque no requeriría de demasiado trabajo y nos permitiría compartir nuestra herramienta en internet con cualquiera que necesite crear clusters multilingües de tweets.

Se creó un servicio web basado con la tecnología *RESTful web services* y además una página web para que consumir el servicio web fuera posible tanto desde un navegador como desde un programa.

El servicio web permite al usuario escoger algunos de los parámetros más importantes del proceso de *clustering* para que pueda comparar por sí mismo los resultados de unas configuraciones con otras y ver cuál se amolda mejor a la colección de tweets que quiere agrupar.

Para consumir el servicio desde un navegador, solo hay que acceder a la página web y seguir los sencillos pasos detallados, e internamente se utilizará el servicio web que se ocupará de que los resultados sean enviados por correo electrónico. Sin embargo, si se quiere consumir desde código el servicio web, deberá enviarse un archivo XML al servicio web. Los componentes del archivo XML para consumir el servicio deben ser:

- Un único nodo principal llamado ‘CMT’. Compuesto de los siguientes subnodos:
    - Nodo ‘email’: cadena con el correo electrónico al que se enviarán los resultados una vez estén listos. En ningún momento se almacena la dirección de correo o se otorga a terceros con fines

comerciales, solo se utiliza para enviar los clusters una vez se han calculado.

- Nodo 'traduccion': carácter '0' ó carácter '1'. Traducción indica qué estrategia se empleará para el proceso de *clustering*, si se escoge '0', se estará escogiendo la estrategia '*Traducción Previa*', sin embargo, si se escoge '1', se empleará '*Traducción Posterior*'
- Nodo 'tweets': cadena de texto donde se incluyen los tweets que se quieren agrupar. El formato que debe llevar el texto con los tweets debe ser el mismo que se ha utilizado para los tweets de la aplicación, y éste está especificado en el capítulo 6 (sección "Corpus de trabajo y formato de los tweets"). También se puede encontrar un guía para dar el formato adecuado a los tweets en la página web. En caso de no venir formateados correctamente los tweets, se enviará un correo electrónico notificando el problema.
- Nodo 'algoritmo': carácter '0' ó carácter '1'. Con este parámetro escogemos el algoritmo con el que queremos agrupar. El '0' representa el algoritmo de *"Fast Affinity Propagation"*, y el '1' representa al algoritmo de *"Louvain"*.
- Nodo 'representacion': carácter '0', carácter '1' ó carácter '2'. Con este nodo se pretende elegir entre los métodos de representación de tweets, '*Vector único*', '*Combinación Lineal* y '*Combinación Borrosa*'. El carácter '0' representa *'Vector único'*, el carácter '1' a *'Combinación Lineal'* y el carácter '2' a *'Combinación Borrosa'*.
- Nodo 'pesado': carácter '0', '1', '2' o '3'. Con este nodo se pretende habilitar la selección de la medida de pesado en los vectores de representación de tweets. El carácter '0' representa la opción *'Binario'*, el '1' representa a *'Binario-IDF'*, el '2' a *'TF'* y el '3' a *'TF-IDF'*.
- Nodo 'normalizacion': cadena 'true' o cadena 'false'. Con esta cadena se habilita la activación o desactivación de la normalización

del contenido de los tweets durante el proceso. La cadena 'true' la activa y la cadena 'false' la desactiva.

- Nodo 'ampliacion': cadena 'true' ó cadena 'false'. Con esta cadena se habilita la activación o desactivación del análisis ampliado sobre los datos de los tweets. La cadena 'true' la activa y la cadena 'false' la desactiva.

Un ejemplo de código XML válido para este servicio web sería la siguiente:

```
<CMT>
       <email>ejemplo@gmail.com</email>
       <traduccion>0</traduccion>
       <tweets>cadena con los tweets en el formato de la
               aplicación</tweets >
       <algoritmo>1</algoritmo >
       <representacion>2</representacion>
       <pesado>3</pesado>
       <normalizacion>true</normalizacion>
       <ampliacion>true</ ampliacion>
</CMT>
```

Tanto la página web como el servicio web están disponibles en una aplicación web en un mismo archivo .WAR, listo para ser descargado en la dirección [35] y desplegado en un servidor web.

Capítulo 5

Descripción informática

En este capítulo se hablará sobre los requisitos de la aplicación y se presentarán los diagramas de casos de uso y diagramas de clases correspondientes al proyecto. Posteriormente se describirán algunos detalles importantes de la implementación y se hablará de cuáles han sido los mayores obstáculos encontrados para completar este trabajo.

5.1 Requisitos

Se van a distinguir entre dos tipos de requisitos, los funcionales y los no funcionales. Los requisitos funcionales hablarán del comportamiento interno que va a tener la aplicación, mientras que los no funcionales son aquellos que no describen información a guardar ni funciones a realizar.

5.1.1 Requisitos funcionales

Los requisitos funcionales del trabajo son:

- **RQF1**: Se desarrollará un sistema capaz de crear grupos o *clusterizar* tweets en más de un idioma.
- **RQF2:** La aplicación permitirá escoger si la traducción de los tweets se lleva a cabo antes de agrupar o por el contrario, agrupar en ambos idiomas y después fusionar clusters traduciendo.
- **RQF3:** La aplicación permitirá utilizar diferentes herramientas de PLN para ver cuál se ajusta mejor a las necesidades de análisis de textos tan cortos como los tweets.
- **RQF4:** Será posible en la aplicación escoger qué algoritmo de *clustering* queremos que se ocupe de agrupar los tweets.

- **RQF5:** La herramienta deberá implementar diversas maneras de representar los tweets.
- **RQF6:** En relación al requisito funcional anterior, obligatoriamente, se implementará una forma de representación de tweets donde cada tweet se representará como varios vectores parciales, se hallará la similitud entre los tweets por cada categoría de vectores y se unirán esas similitudes con un motor borroso para determinar la similitud final entre dos tweets.
- **RQF7:** Se habilitarán opciones para configurar la aplicación y hacer posible la comparación de resultados con distintas configuraciones.
- **RQF8:** Se podrá ver una medición de los resultados al final del proceso con las medidas Precision, Recall y medida-F cuando los tweets incluyan la información de a qué cluster realmente pertenecen.
- **RQF9:** Se deberá exportar la aplicación a un servicio web y habilitar una web para tener acceso al servicio desde navegadores y desde código.
- **RQF10:** La aplicación web deberá enviar un correo electrónico al usuario que haya solicitado el servicio con los tweets agrupados y con la medida de la calidad de los grupos si el usuario nos incluye en los tweets el cluster al que pertenecen.

5.1.2 Requisitos no funcionales

Los requisitos no funcionales de este trabajo son:

- **RQNF1:** Todo el sistema deberá estar implementado en Java aunque puede utilizar herramientas escritas en otros lenguajes.
- **RQNF2:** Los esfuerzos de esta investigación deben ser averiguar las mejores estrategias de representación de tweets para el *clustering* multilingüe de los mismos.
- **RQNF3:** El tiempo de procesamiento debe ser razonable.

5.2 Diagramas de casos de uso

Un caso de uso (CU) es una técnica para la captura de requisitos potenciales de un nuevo sistema o una actualización de software. Cada CU proporciona uno o más escenarios que indican cómo debería interactuar el sistema con el usuario o con otro sistema para conseguir un objetivo específico. Los diagramas de CU sirven para especificar la comunicación y el comportamiento de un sistema mediante su interacción con los usuarios.

Dado que hay muchas posibilidades a la hora de configurar el sistema desarrollado, no se van a presentar todos los casos de uso posibles en esta memoria, solo uno muy representativo del sistema. El CU que se mostrará es el de la aplicación empleando las configuraciones con las que mejores resultados se logran. Estas configuraciones son:

- Traducción previa de los tweets
- Normalización del contenido de los tweets activada
- Análisis ampliado activado
- Combinación borrosa con representación multivectorial de los tweets
- Medida de ponderado TF-IDF
- Algoritmo de Louvain para *clusterizar*

Este CU aparece en la Figura 5.1, es escueto debido a la poca interacción entre usuario y aplicación, los procesos los completa internamente el sistema.

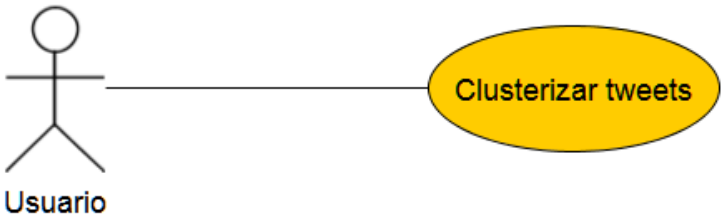

Figura 5.1. Modelo de casos de uso de la aplicación de *clustering.*

A continuación se describe el flujo de eventos para el caso de uso anterior.

Descripción

- Caso de uso: *Clusterizar* tweets
- Actores: Usuario
- Descripción: El caso de uso empieza cuando el usuario crea una instancia del objeto CMT (*Clustering Multilingüe de Tweets*) y comienza el proceso. El flujo de eventos se presenta en la Tabla 5.1.

Flujo principal

Acción del usuario	Respuesta del sistema
1. El usuario crea una instancia del objeto CMT	2. El sistema detecta en qué idioma está escrito cada tweet. 2.1. Para los tweets de ambos idiomas, los normaliza en el idioma correspondiente. 2.2. Traduce los tweets de español a inglés. 3. Identifica qué datos forman parte de cada tweet. 3.1. Identifica los Hashtag de cada tweet. 3.2. Identifica las Entidades Nombradas (ENs) y usuarios de cada tweet. 3.3. Identifica los sustantivos y adjetivos de cada tweet. 3.4. Identifica los verbos de cada tweet. 4. Ejecuta el análisis ampliado sobre los datos de los tweets. 5.1. Crea una lista con todos los Hashtag de todos los tweet. 5.2. Crea una lista con todas las ENs y los usuarios de todos los tweet. 5.3. Crea una lista con todos los sustantivos y adjetivos de todos los tweet. 5.4. Crea una lista con todos los verbos de todos los tweet.

	6. Representa cada tweet con vectores utilizando la medida de ponderación TF-IDF en las 4 categorías definidas. **7.** Crea 4 matrices de similitud parcial con los vectores de cada tweet correspondientes a cada categoría. **8.** Utiliza el motor borroso para fusionar las similitudes parciales de las matrices en una única matriz de similitud. **9.** Ejecuta el algoritmo de *clustering* de *Louvain* con parámetro de granularidad 2.5. **10.** Muestra los clusters por pantalla y los resultados de calidad de éstos.

Tabla 5.1. Flujo de eventos del caso de uso *Clusterizar tweets*.

5.3 Diagramas de clases

La Figura 5.2 muestra un diagrama de clases con las clases principales del sistema desarrollado.

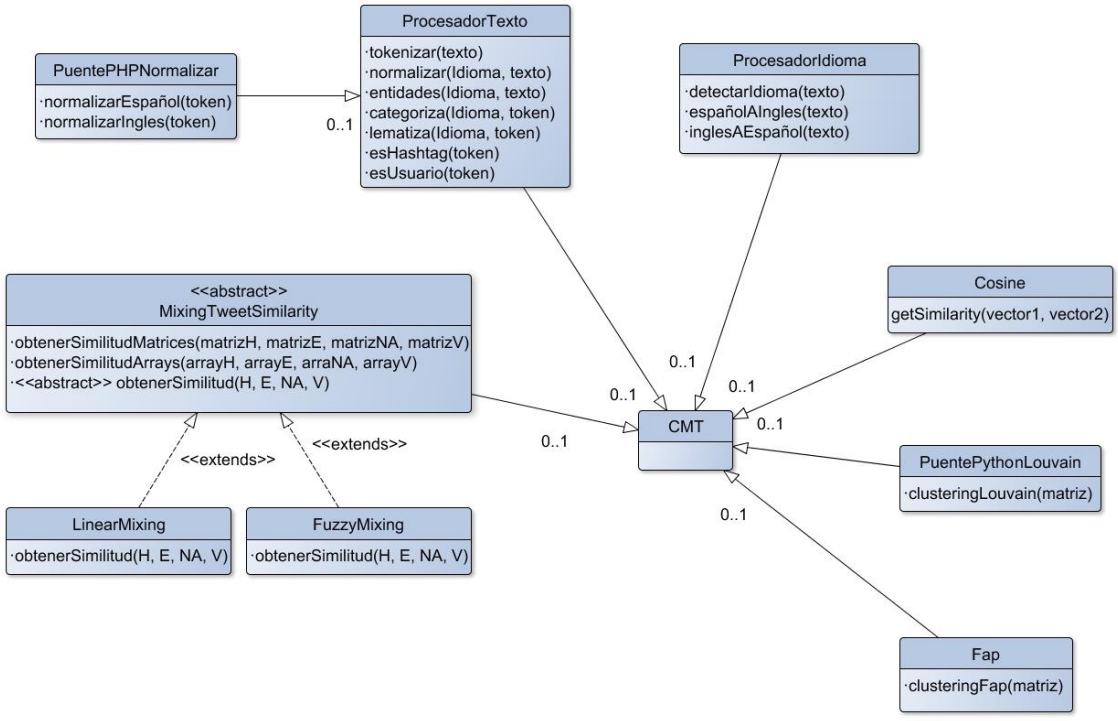

Figura 5.2. Diagrama de clases de la aplicación de *clustering.*

`CMT`: es nuestra clase principal, la que invocamos para obtener los clusters de tweets.

`ProcesadorTexto`: esta clase es utilizada para realizar operaciones sobre textos y obtener información de ellos. Los principales métodos con los que cuenta están especificados en la Figura 5.2 y sus propósitos son los siguientes:

- `tokenizar(texto)`: este método divide un texto en una lista de tokens, y utiliza los espacios para dividir el texto. Esto nos sirve a la hora de analizar el contenido de los tweets.
- `normalizar(idioma,texto)`: con este método, podemos normalizar el texto de un tweet indicando qué texto contiene y en qué idioma está escrito. Utiliza para ellos la clase *`PuentePHPNormalizar`*, que accede al servicio web encargado de la normalización.
- `entidades(idioma,texto)`: de este método podemos valernos para extraer las Entidades Nombradas en el contenido de un tweet. Utiliza para ello la librería StanfordNLP, que nos permite trabajar con herramientas específicas para cada idioma.
- `categoriza(idioma,token)`: este método nos determina cuál es la categoría gramatical de un token, es decir, nos indica si es un sustantivo, un adjetivo, un verbo o cualquier otro tipo de palabra.
- `lematiza(idioma,token)`: La lematización es el proceso con el cual obtenemos el lema de una palabra. La palabra lema tiene muchas definiciones según la RAE, pero en este contexto lema significa entrada de un diccionario o enciclopedia. Utilizamos esa entrada sin género ni número para representar a todas las flexiones de ésta. En el contexto que nos encontramos de investigación de *clustering* de textos, es muy corriente que los conceptos '*stema*' y *'lema'* sean utilizados como sinónimos o interpretados como dos operaciones diferentes, así que se puede decir que no existe una definición exacta de la operación que se emplea aquí, pero por simplicidad, en este trabajo se le ha llamado 'Lematiza' a la operación. Para implementarla, se ha utilizado unas listas de palabras alojadas en la web de Lexiconista [36], que contiene pares de lo que ahí llaman

lemma/token, y que están disponibles tanto para inglés como para español, y nos sirve para agrupar en una única palabra varias referidas al mismo concepto pero con diferentes variaciones. Algunos ejemplos:

- Verbos: 'cantaron'→'cantar', 'cantarán'→'cantar' , 'singing'→'sing' , 'sung' →'sing'
- Sustantivos: 'leona'→'león', 'leones'→'león', 'lions'→'lion'

  Y así con todo tipo de palabras.

- `esHashtag(token)`: un método que rápidamente nos confirma si un token es un Hashtag.
- `esUsuario(token)`: método que nos confirma si un token es nombre de un usuario de Twitter.

`PuentePHPNormalizar`: como ya se me mencionó en el capítulo de *Metodologías y Tecnologías*, para normalizar textos de tweets, lo que hacemos es invocar un servicio web escrito con una tecnología PHP. Este servicio solo permite ser invocado con el cliente PHP de la tecnología con la que fue escrita. Esta clase sirve de puente entre Java y PHP, ya que encapsula la creación de un proceso que utiliza código PHP para ponerse en contacto con la página web y obtener la normalización del texto solicitado, tanto en inglés como en español, dependiendo del método que se elija invocar.

`ProcesadorIdioma`: clase enfocada a tratar los asuntos relacionados con el idioma de los tweets, utilizada tanto para traducir como para detectar el idioma en el que está un texto. Sus métodos sirven para traducir de un idioma a otro y detectar idiomas. Todo tweet que vaya en la entrada y no se detecte como español o inglés será clasificado como idioma "otro" y se le tratará como un tweet en inglés por motivos de simplicidad.

`Cosine`: clase con un único objetivo muy sencillo, dar soporte al cálculo de la similitud coseno entre dos vectores.

`<<abstract>>MixingTweetSimilarity`: esta clase es una clase abstracta, que proporciona dos métodos a las clases que extienden de ella y obliga a que implementen un tercero. Los métodos que implementa esta clase son:

- `obtenerSimilitudMatrices(matrizH,matrizE,matrizNA,matrizV)`: este método sirve para formar la matriz de similitud final a partir de 4 matrices de similitud parciales. Para ello emplea el método `obtenerSimilitud`, que debe implementar la clase que extienda de esta.
- `obtenerSimilitudArrays(arrayH,arrayE,arrayNA,arrayV)`: método empleado para la fusión de clusters. Se utiliza para saber la similitud entre un cluster en español y el resto de clusters en inglés y utiliza, igual que el método anterior, 4 similitudes parciales para obtener la similitud final, invocando al método `obtenerSimilitud` de la clase que extiende.

El método que resta es abstracto y no tiene implementación aquí, se le da en las clases que implementen esta clase que se definirán a continuación.

`LinearMixing`**:** clase Java que se emplea para la fusión de matrices y arrays de similitud parcial en una única matriz, empleando un enfoque lineal. Esta clase extiende de la clase `MixingTweetSimilarity` y, por consiguiente, tiene los dos métodos de obtención de similitudes completas. El método que implementa es `obtenerSimilitud` que es donde se halla el enfoque lineal. Esta clase corresponde a la representación de tweets *"Combinación Lineal"*. En el método `obtenerSimilitud`, lo que recibimos son las similitudes parciales de las 4 categorías entre un par de tweets y multiplicamos cada una de ellas por su peso correspondiente y los resultados los sumamos para obtener la similitud final del par de tweets.

`FuzzyMixing`: es otra clase que extiende de la clase abstracta `MixingTweetSimilarity` y ya trae implementados dos métodos. Solo implementamos un método aquí, `obtenerSimilitud`, que emplea un motor borroso para averiguar la similitud final de las similitudes parciales. Es la clase correspondiente a la representación de tweets *"Combinación Borrosa"*. Recibe 4 valores de similitud parcial entre dos tweets y nos devuelve una similitud absoluta entre éstos empleando el motor borroso.

`Fap`: esta clase utiliza código de la librería APRO [31] para ejecutar el algoritmo de *clustering "Fast Affinity Propagation"*, y su método `clusteringFap` sirve para ese

propósito, recibiendo una matriz de similitud y los tweets que representan, y devolviendo los clusters que se han generado con este algoritmo.

`PuentePythonLouvain`: como ya se mencionó en el capítulo de *Metodologías y Tecnologías*, para utilizar el algoritmo de *Louvain* se ha recurrido a un código escrito en lenguaje Python. Esta clase sirve de puente entre Java y Python. Nos encapsula la invocación de código de este lenguaje distinto y nos permite simplemente invocar al método `clusteringLouvain` y obtener los clusters de tweets como salida a partir de la matriz de similitud, los tweets que representa y el parámetro de granularidad que necesita este algoritmo.

5.4 Detalles de la implementación

Ahora se especificarán algunos detalles de la implementación que destacan sobre el resto y se considera que deben ser detallados por su importancia.

Fusión de clusters

Las implementaciones creadas en este trabajo para resolver el problema de trabajar con tweets en distintos idiomas ya se han detallado pero para no olvidarnos, son dos, *Traducción Previa* y *Traducción Posterior*. Aquí se va a explicar el código que se ha implementado para la fusión de clusters que sucede en la opción *Traducción Posterior.*

Como ya se describió, esta estrategia hace que se trabaje con los tweets independientemente por idioma y cuando ya se han encontrado los clusters monolingües, se fusionan los resultados creando clusters multilingües.

La fusión de clusters españoles e ingleses comienza con la identificación de los centroides de cada cluster. Brevemente, el centroide de un cluster se define como el punto equidistante de los objetos pertenecientes a dicho cluster, y en nuestro caso, el centroide será un tweet que servirá para representar al cluster al que pertenece. El centroide se halla primero sumando las similitudes de cada tweet con todos sus compañeros, y seguidamente entre estos resultados, escogiendo el tweet que reúne mayores similitudes con el resto.

Una vez tenemos los centroides de todos los clusters, comparamos cuánto se parecen los clusters en español con los clusters en inglés, y para ello comparamos el centroide que representa a cada cluster. La manera en la que se halla la similitud entre los centroides no es siempre la misma, se emplea el método de representación de tweets que se haya escogido para esa ejecución de la aplicación.

Cada cluster en español se fusionará con el cluster en inglés con el que más se asemeje, siempre y cuando la similitud sea mayor a un umbral. El objetivo de la inclusión de este umbral es evitar que se fusionen clusters con muy poca similitud entre ellos. Este umbral de similitud es el número 0.033 y se obtuvo mediante una experimentación previa con diferentes valores hasta encontrar el más adecuado.

Código expandible a más idiomas

Como siguiente detalle importante de la implementación de esta aplicación, se hablará de la orientación que se ha dado al código para ser expandido y hacer posible trabajar con más idiomas.

La aplicación funciona con dos idiomas, inglés y español, y para añadir un nuevo idioma al grupo de los aceptados en los tweets que puede agrupar la aplicación es necesario añadir 2 elementos.

1. Traductor para el nuevo idioma.
2. Normalizador para el nuevo idioma.

Con estos dos elementos software estaremos listos para añadir cualquier idioma que nos propongamos. La primera modificación que deberíamos hacer sería cambiar la clase `ProcesadorIdioma` añadiendo un idioma más a las posibilidades que detecta, ya que en esta implementación se ha hecho que cualquier tweet en un idioma que no sea español o inglés sea incluido en la categoría de idioma “otros”. Una vez que ya detectase nuestro idioma deseado la clase correspondiente a los idiomas, deberíamos añadir métodos para traducir de ese idioma a inglés y viceversa. Es posible que no hubiera que buscar herramienta software para este traductor, ya que la tecnología que se usa en este trabajo incluye la posibilidad de traducir entre más de 60 idiomas, los más utilizados en este planeta.

Cuando hubiéramos adaptado la clase a este idioma nuevo, deberemos modificar la clase `ProcesadorTexto` en su método `normaliza(idioma,texto)` y añadir soporte a la normalización en el idioma que estamos agregando.

Una vez listos los dos elementos software necesarios, vendría la parte más ardua, habría que modificar el flujo de datos para que se incluyeran las operaciones del idioma que agregásemos. Habría que incluirlo en todas las opciones, tanto traducción previa (traduciendo a inglés todos los tweets) y traducción posterior (trabajar con los tweets en el idioma original y luego fusionarlos). En esta segunda opción sería también necesario complicar un poco más el método en el que se fusionan los clusters. En la implementación original solo hay que buscar si se fusionan los clusters españoles con los ingleses, pero con más idiomas habría que comparar cada cluster con los clusters del resto de idiomas.

5.5 Obstáculos más relevantes durante el desarrollo

Durante el desarrollo de esta investigación ha habido muchos obstáculos que sobrepasar y problemas de todas las variedades. En esta sección se va a hablar de los problemas que más conflicto han generado, ya sea por el tiempo que han empleado o por lo aparentemente sencillos que parecían al principio. Todos los obstáculos aquí destacados fueron solventados y tuvieron solución adecuada.

Lexemas, raíces léxicas, *lematizadores* o *stemizadores*

Bien entrado en el desarrollo de la aplicación nos encontramos con un problema que requería decisiones importantes, y era el problema de la *lematización*. Este problema no tiene un nombre único, es un tema que se aborda en la comunidad de PLN de diferentes maneras, y es que en los diferentes idiomas, hay varias palabras para referirnos a un mismo concepto, ya sean verbos con sus conjugaciones o sustantivos con su género y número u otras palabras,. Para nosotros los seres humanos es muy sencillo ver que las palabras '*vendedor*' y '*vendedora'* hablan del mismo concepto, pero para una máquina no es igual. Debemos utilizar alguna

herramienta que permita a nuestras aplicaciones entender que hay palabras escritas de manera distinta que se refieren al mismo concepto.

Hay varias opciones en la comunidad informática para resolver este problema, pero cuando se quiso tomar una decisión sobre cuál escoger hubo un grave problema. De las herramientas en español se probaron y estudiaron prácticamente todas las soluciones disponibles en Java y ninguna desempeñaba las funciones esperadas correctamente, sino con multitud de errores. Sea dicho que para la versión inglesa los resultados eran aceptables.

Tras mucho navegar en la Web en busca de una mejor solución, se encontró una página web, Lexiconista [36], que ofrecía algo que nos permitiría crear nuestra propia herramienta de una manera muy rápida. Una lista de pares de palabras en español, lema-palabra para 497.560 palabras, en un único fichero con una línea por cada par lema-palabra, y que separa con un espacio los lemas y las palabras. Con esta lista pudimos crear una tabla hash que por cada palabra escrita en español nos devolviese rápidamente su lema. Finalmente, debido a la sencillez de esta solución y su alta efectividad, se decidió optar por ella tanto en español como en inglés, ya que en este lenguaje también existe una colección de pares lema-token en la misma página web.

Normalización

En las fases más tempranas de este desarrollo, rápidamente se vio que uno de los problemas a la hora de analizar el contenido de los tweets sería la mala ortografía que éstos traen. La tutora de este trabajo era conocedora de un proyecto de la UNED donde se estaba desarrollando un servicio web capaz de corregir tweets tanto en español como inglés. La verdad es que parecía hecho a medida para nuestro problema, ya que lo solucionábamos tan solo invocando un servicio web.

La cosa se puso difícil cuando se empezó a desarrollar un cliente Java para ese servicio web. No fue posible, el servicio web había sido escrito en PHP, lo cual no debería presentar un problema, pero según la guía de ese proyecto, en su desarrollo se utilizó una herramienta software llamada *NUSOAP*, solo disponible en PHP y que hacía aparentemente imposible consumir el servicio web desde Java.

Afortunadamente, el alumno que desarrolló la aplicación para su trabajo de fin de grado hizo una guía de instalación para que cualquiera pudiese instalarlo en su propio ordenador si fuera necesario. Esta fue la siguiente decepción, ya que la instalación no fue posible debido a errores en la guía de instalación.

Se decidió entonces buscar otra manera de utilizar este servicio web dado el gran valor que podía aportar, y finalmente se consiguió invocar desde Java un Script PHP que utilizara el cliente de la aplicación NUSOAP para invocar el servicio web. Esto sí resultó exitoso y pudimos utilizar la misma herramienta de normalización para los dos idiomas con los que trabajamos en este proyecto.

Ampliación del análisis

El desarrollo de la ampliación del análisis no fue nada fácil. La idea surgió una vez ya estaba funcionando la aplicación y se podían observar los primeros resultados del *clustering*. Cuando vimos que los resultados eran algo peores de lo esperado, que los clusters eran imprecisos y que había cosas que claramente se podían mejorar haciendo un post-análisis o ampliación del análisis de los datos de cada tweet, empezamos a ver cómo hacerlo. En los clusters se podían ver ligeros patrones de los errores que había, que permitían hacerse una idea de cómo aumentar la representación de los tweets con más datos que no aparecían directamente en el tweet, pero que eran datos que podían estar relacionados con el contenido, como Hashtags idénticos a ENs por ejemplo.

Todas las operaciones sobre los datos de los tweets se averiguaron tras un largo proceso de prueba y error, probando varias operaciones sobre el flujo de datos y observando el efecto que cada una de ellas tenía individualmente en el resultado final. Los clusters resultantes tras el análisis ampliado son notablemente mejores, como se podrá ver en el capítulo 6 que presenta los resultados experimentales.

Desarrollo de la aplicación web

El desarrollo del servicio y aplicación web parecía sencillo a primera vista, pero finalmente tuvo muchas más complicaciones de las esperadas.

La primera y más importante fue con respecto al servidor web. Tras muchas pruebas del servicio web, durante la carga de la librería Stanford la máquina virtual de java necesitaba emplear mucha memoria y por más que se aumentaba, seguían existiendo problemas. Al final, tras mucho navegar la en la Web en busca de alguien que hubiera tenido problemas similares con GlassFish, o algún tipo de ayuda, resultó que el problema no se resolvía modificando la memoria que consumía el proyecto, sino modificando el parámetro de memoria máximo del servidor web. En resumen, un montón de tiempo malgastado por una nimiedad.

No solo fue eso lo que presentó problemas. Se comprobó que era necesario establecer un control de errores, para distinguir errores durante el proceso de errores en el formato de los tweets. Había que avisar al usuario que invocaba el servicio web mediante correo electrónico de cuál había sido el error, si era un error interno o si era un error del formato de los tweets. En el caso de que fuera un error en el formato, además se debería incluir una descripción del error para que se pudiera solucionar. Esto conllevó mucho trabajo adicional, ya que hubo que probar individualmente cada error y verificar que se notificaba correctamente lo ocurrido, haciendo de la creación del servicio y de la página web una tarea mucho más complicada de lo se suponía a priori.

Capítulo 6

Resultados Experimentales

En este capítulo se va a describir el corpus con el que se ha trabajado para desarrollar y evaluar la aplicación, además de la evaluación del rendimiento del sistema implementado. La evaluación consiste en una serie de experimentos donde se han comparado los resultados obtenidos con las diferentes configuraciones planteadas en el proyecto. Se podrá ver qué opciones de las configuraciones son las más adecuadas para que el proceso de *clustering* de tweets logre los mejores resultados posibles.

6.1 Corpus de trabajo y formato de los tweets

Durante esta investigación, se ha trabajado con 3 corpus distintos, las colecciones BACKGROUND, TRAINING y TEST:

Training: este corpus ha servido como conjunto de entrenamiento. Cuando hacía falta averiguar la granularidad del algoritmo de *Louvain* o ajustar algunos parámetros o reglas del motor borroso, todas estas pruebas se ejecutaban sobre esta colección.

Background: esta colección de tweets también sirvió para realizar pruebas de tipo entrenamiento, pero de manera complementaria, la intención era que la aplicación no se adaptase demasiado a la colección de tweets de training y fuera válida para todo tipo de tweets.

Test: ha sido empleada para evaluar los resultados del sistema propuesto y determinar la calidad de la aplicación.

Estas colecciones con tweets han sido extraídas de RepLab [37], una competición internacional para la evaluación de sistemas de tratamiento de reputación online, organizado por CLEF (*Cross-Language Evaluation Forum*) [38].

Las tres colecciones originales contenían tweets en muchos idiomas e incluían varias anotaciones diferentes por cada tweet: el identificador del tweet, el autor del tweet y una cadena de texto representando el grupo temático al que pertenecía. Gracias a que incluían este grupo temático al que pertenecían no hizo falta anotarlos de nuevo para poder medir la calidad de los resultados.

Al tener las colecciones de tweets de varios idiomas, hubo que extraer los tweets en español e inglés, porque en este trabajo, como primera aproximación, se quería hacer un planteamiento en dos idiomas solamente. De media, habría alrededor de 350.000 tweets en español e inglés en cada colección, así que hubo que reducir la cantidad bruscamente hasta alrededor de 500 por cada colección para que el tiempo de ejecución fuera aceptable (inicialmente, dada la complejidad de la propuesta y los diferentes pasos de procesamiento que necesita, no era un objetivo prioritario en este trabajo el perseguir reducir los costes computacionales al máximo, sino ver la viabilidad de la propuesta).

A la hora de extraer 500 tweets de una colección tan grande, se corría el riesgo de extraer muchos tweets pertenecientes a los mismos clusters. Para evitar esto, el método por el que se optó para reducir las cantidades de tweets en las colecciones fue, calcular qué porcentaje sería 500 tweets en la cantidad específica de tweets de cada colección, y extraer ese mismo porcentaje de cada grupo temático de tweets para reducir el número de tweets pero maximizando el número de clusters en nuestras colecciones de trabajo.

En todo este trabajo se ha utilizado el formato de entrada original que traían los tweets directamente de las colecciones de RepLab, muy adecuado para este proyecto. El formato de texto de entrada se ha mantenido igual para todas las formas de esta aplicación, la propia aplicación en sí y sus extensiones, el servicio y la página web.

El formato tiene las siguientes características:

- Por cada tweet, una línea terminada con salto de línea.
- En cada línea 4 categorías de datos separados por tabuladores y rodeados de dobles comillas (“dato“):

- En la primera categoría, un identificador único del tweet que se utilizará para identificar los tweets en el correo de respuesta en el servicio Web.
- En la segunda categoría, el autor del tweet, sin el carácter @ al principio. Por ejemplo, si el autor de un tweet es “@policia”, el dato que tenemos que poner entre comillas dobles es “policía”.
- En la tercera categoría incluiremos una cadena que represente al cluster al que pertenece, para que el servicio Web nos calcule una estimación de la calidad de los clusters y lo añada en el correo de respuesta.
- En la cuarta categoría, añadimos el texto del tweet.
- Aunque las tres primeras categorías vienen incluidas en las colecciones originales, para el servicio Web se han hecho opcionales. La decisión sobre estas opciones debe ser vinculante para todos los tweets de la colección, es decir, que no es válido añadir datos de estas categorías si no se añade esa categoría a todos.

A continuación se presentan diferentes ejemplos de tweet con el formato válido para la aplicación y servicio Web. En primer lugar, la Figura 6.1 presenta un grupo de tweets con todos los datos incluidos.

```
"278364451512664066"    "valerieeusse"  "RL2013D01E001" "Hay mi amor mañana te recojo en el BMW jaja'"
"278375574819442688"    "xLoveDove" "RL2013D01E001" "My mom wants me to get a BMW , hmm."
"278380977062043648"    "ThugAngel2011" "RL2013D01E001" "And a BMW 5 series"
```

Figura 6.1. Tweets en formato correcto con todos los datos.

Un grupo de tweets con los datos de todas las categorías excepto la primera se presentan en la Figura 6.2.

```
""  "valerieeusse"  "RL2013D01E001" "Hay mi amor mañana te recojo en el BMW jaja'"
""  "xLoveDove" "RL2013D01E001" "My mom wants me to get a BMW , hmm."
""  "ThugAngel2011" "RL2013D01E001" "And a BMW 5 series"
```

Figura 6.2. Tweets en formato correcto con todos los datos menos el identificador.

Por último, la Figura 6.3 presenta un conjunto de tweets con un formato que no es válido (ya que no son consistentes en el número de categorías que especifican).

```
"278364451512664066"    "valerieeusse"  ""  "Hay mi amor mañana te recojjo en el BMW jaja'"
""  "xLoveDove" "RL2013D01E001" "My mom wants me to get a BMW , hmm."
"278380977062043648"    ""  "RL2013D01E001" "And a BMW 5 series"
```

Figura 6.3. Tweets en formato incorrecto.

6.2 Configuraciones evaluadas

En este punto se recuerda que en esta investigación uno de los objetivos era explorar diferentes opciones de configuración. A continuación se especifica una lista de configuraciones que se han creado para explorar diferentes opciones a lo largo del proceso de *clustering* multilingüe de tweets:

- *"Traducción Previa"* de los tweets vs. *"Traducción Posterior"*.
  - En la opción *"Traducción Posterior"*, el valor mínimo de similitud entre clusters para fusión. Comparaciones de valores entre 0 y 0.2.
- Tecnologías PLN: *"StanfordNLP"* vs. *"OpenNLP"*.
- Métodos de creación de las matrices de similitud. *"Vector único"* vs. Varios vectores (*"Combinación Lineal"* vs. *"Combinación Borrosa"*).
- Medida de ponderación en los vectores de representación: *"Binario"* vs. *"Binario-IDF"* vs. *"TF"* vs. *"TF-IDF"*.
- Normalización de los tweets: Activada vs. Desactivada.
- Ampliación del análisis: Activado vs. Desactivado.
- Algoritmo de clustering: "*Louvain"* vs. "*Fast Affinity Propagation*".
  - Valor de granularidad algoritmo de Louvain.

Debido a que la memoria tiene un tamaño limitado, se ha decidido únicamente incluir evaluaciones de las configuraciones que resultan más importantes y aportan más información sobre el proceso de *clustering*. En este caso se ha optado por mostrar los resultados de:

1. *"Traducción Previa"* de los tweets vs. *"Traducción Posterior"*.

2. Métodos de creación de las matrices de similitud. *"Vector único"* vs. Varios vectores (*"Combinación Lineal"* vs. *"Combinación Borrosa"*).
3. Medida de ponderación en los vectores de representación: *"Binario"* vs. *"Binario-IDF"* vs. *"TF"* vs. *"TF-IDF"*.
4. Normalización de los tweets: Activada vs. Desactivada.
5. Ampliación del análisis: Activado vs. Desactivado.
6. Algoritmo de clustering: "*Louvain"* vs. "*Fast Affinity Propagation*".

Para todas las pruebas aquí realizadas se utilizarán las medidas que habitualmente su emplean para analizar el rendimiento de aplicaciones de clasificación automática, éstas son *Recall, Precision* y *F-measure*.

La medida *Recall* evalúa la cobertura de un sistema, es decir, calcula la probabilidad de que el sistema dé con una de las respuestas reales. Su ecuación es:

$$Recall = \frac{nº\ respuestas\ correctas}{nº\ respuestas\ reales}$$

La medida *Precision* calcula la probabilidad de que un sistema dé una respuesta y ésta sea correcta. La ecuación que se utiliza para esta medida es:

$$Precision = \frac{nº\ respuestas\ correctas}{nº\ respuestas\ dadas}$$

En último lugar, *F-measure* es la combinación de las dos medidas anteriores, que consiste en calcular la media ponderada dándole un valor prefijado al coeficiente *β*. En este caso *β* tomará el valor 1 para que tengan el mismo peso *Precision* y *Recall* en el resultado. La ecuación que define *F-measure* es:

$$F = \frac{(\beta^2 + 1) * Precision * Recall}{\beta^2 * Precision + Recall}$$

6.2.1 *"Traducción Previa"* vs. *"Traducción Posterior"*

Como ya se comentó, estas dos estrategias hacen referencia al momento en que se traducen los tweets de español a inglés, en *"Traducción Previa"* se traduce de

español a inglés antes de agrupar, mientras que en *"Traducción Posterior"* se agrupa por idiomas y después se fusionan los clusters de ambos idiomas si existen similitudes entre sus tweets.

En la Tabla 6.1 podemos ver los resultados de comparar ambas estrategias en las diferentes colecciones de datos. El resto de configuraciones se mantuvieron fijas en ambos casos y fueron las siguientes:

- Tecnología PLN: *"StanfordNLP".*
- Métodos de representación y creación de las matrices de similitud: Varios vectores con *"Combinación Borrosa"*.
- Medida de ponderación en los vectores de representación: *"TF-IDF"*.
- Normalización de los tweets: Desactivada.
- Ampliación del análisis: Activado.
- Algoritmo de *clustering*: *"Louvain"*.
    - Valor de granularidad algoritmo *Louvain*: 2.5.

	Medidas	TEST
Traducción Previa	Recall	0.7
	Precision	0.7
	F	0.7
Traducción Posterior	Recall	0.61
	Precision	0.51
	F	0.56

Tabla 6.1. Resultados de *clustering* con las dos estrategias de traducción de tweets.

Como se puede ver en la tabla, los mejores resultados se obtienen cuando se traduce previamente, es decir, cuando se traduce de español a inglés y luego se agrupan los tweets en un único idioma.

6.2.2 *"Vector Único"* vs. *"Combinación Lineal"* vs. *"Combinación Borrosa"*

En la siguiente experimentación se quería ver cuál es la mejor forma de representar los tweets para obtener buenos resultados de *clustering*. La Tabla 6.2 presenta los resultados comparando la representación de vector único con las representaciones con varios vectores. El resto de configuraciones que se han mantenido fijas y que se han utilizado en esta experimentación han sido:

- Traducción de los tweets: *"Traducción Previa".*
- Tecnología PLN: *"StanfordNLP".*
- Medida de ponderación en los vectores de representación: *"TF-IDF"*.
- Normalización de los tweets: Desactivada.
- Ampliación del análisis: Activado.
- Algoritmo de *clustering*: *"Louvain"*.
    - Valor de granularidad algoritmo *Louvain*: 2.5.

	Medidas	TEST
Vector único	Recall	0.78
	Precision	0.51
	F	0.62
Combinación Lineal	Recall	0.72
	Precision	0.5
	F	0.59
Combinación Borrosa	Recall	0.7
	Precision	0.7
	F	0.7

Tabla 6.2. Resultados de *clustering* con las diferente formas de representar los tweets.

Los resultados aquí también son reveladores, si nos fijamos en la medida F, la mejor opción de todas es la *"Combinación Borrosa"*. Sorprendentemente, la opción de *"Vector único"* es mejor que *"Combinación Lineal"*, que desde el principio se podía sospechar que sería la segunda mejor. Con estos resultados se puede deducir que si queremos clusters más precisos, la opción más adecuada es claramente *"Combinación Borrosa"*, pero si no es nuestra preferencia, la opción *"Vector Único"* nos dará igualmente excelentes resultados.

6.2.3 Ampliación del análisis: Activado vs. Desactivado

Otra de las cosas que hemos evaluado es cuánto aporta el análisis de los tweets ampliado que hemos propuesto sobre no hacer dicho análisis. La Tabla 6.3 muestra los resultados obtenidos con la ejecución de la aplicación con el análisis ampliado activado y desactivado. El resto de configuraciones se mantuvieron fijas en ambos casos y fueron las siguientes:

- Traducción de los tweets: *"Traducción Previa"*.
- Tecnología PLN: *"StanfordNLP".*
- Métodos de representación y creación de las matrices de similitud: Varios vectores con *"Combinación Borrosa"*.
- Medida de ponderación en los vectores de representación: *"TF-IDF"*.
- Normalización de los tweets: Desactivada.
- Algoritmo de *clustering*: *"Louvain"*.
    - Valor de granularidad algoritmo *Louvain*: 2.5

	Medidas	TEST
Activado	Recall	0.7
	Precision	0.7
	F	0.7
Desactivado	Recall	0.59
	Precision	0.67
	F	0.63

Tabla 6.3. Resultados de *clustering* con y sin el análisis de los tweets ampliado.

Esta tabla deja muy claro cuál es la opción vencedora en el análisis ampliado. Cuando está activado, en todas las pruebas mejora la calidad general de los resultados, especialmente en la medida *Recall*.

6.2.4 Normalización: Activada vs. Desactivada

Como el proceso de normalización puede añadir ruido también en el procesamiento de los tweets, otra de las cosas que queríamos comprobar en la experimentación era ver cuánto aportaba a los resultados el realizar o no la fase de normalización. La Tabla 6.4 presenta los resultados donde se pueden ver los efectos que tiene activar o desactivar la normalización del contenido de los tweets en el resultado final. El resto de configuraciones fijas utilizadas en esta experimentación han sido:

- Traducción de los tweets: *"Traducción Previa".*
- Tecnología PLN: *"StanfordNLP".*
- Métodos de representación y creación de las matrices de similitud: Varios vectores con *"Combinación Borrosa"*.

- Medida de ponderación en los vectores de representación: *"TF-IDF"*.
- Ampliación del análisis: Activado.
- Algoritmo de *clustering*: *"Louvain"*.
    - Valor de granularidad algoritmo *Louvain*: 2.5.

	Medidas	TEST
Activada	Recall	0.72
	Precision	0.72
	F	0.72
Desactivada	Recall	0.7
	Precision	0.7
	F	0.7

Tabla 6.4. Resultados de *clustering* con y sin la normalización de los tweets.

Estos resultados demuestran que activando la normalización de los tweets, se mejora la calidad de los clusters en todos sus aspectos, aunque pueden ser mejoras muy bajas. La normalización aumenta el tiempo de computación total notablemente, y si merece la pena la espera esta pequeña pero real mejora, es decisión de quien ejecute la aplicación.

6.2.5 Algoritmo *"Louvain"* vs. *"FAP"*

Por último, dado que hemos utilizado dos algoritmos de clustering diferentes, hemos experimentado para ver cuál de los dos ha obtenido los mejores resultados agrupando tweets. La Tabla 6.5 muestra los resultados. El resto de configuraciones que se han mantenido fijas en esta experimentación han sido:

- Traducción de los tweets: *"Traducción Previa"*.
- Tecnología PLN: *"StanfordNLP".*
- Métodos de representación y creación de las matrices de similitud: Varios vectores con *"Combinación Borrosa"*.
- Medida de ponderación en los vectores de representación: *"TF-IDF"*.
- Normalización de los tweets: Desactivada.
- Ampliación del análisis: Activado.

	Medidas	TEST
Louvain	Recall	0.7
	Precision	0.7
	F	0.7
Fast Affinity Propagation	Recall	0.66
	Precision	0.55
	F	0.6

Tabla 6.5. Resultados de la aplicación con los dos algoritmo de *clustering.*

Estos resultados no dan lugar a duda, los resultados mejoran cuantitativamente si el algoritmo de *clustering* que escogemos es el algoritmo de *"Louvain"*. Uno de los motivos de que esto ocurra es debido al parámetro relacionado con la granularidad del agrupamiento que necesita este algoritmo y que se ha determinado en una experimentación previa con las colecciones de training.

Capítulo 7

Conclusiones

En este capítulo se presentan las conclusiones obtenidas de los desarrollos del trabajo, tanto personales como generales.

7.1 Aportaciones y conclusiones alcanzadas

El objetivo general de este trabajo ha sido crear un sistema capaz de agrupar temáticamente tweets en distintos idiomas, y esto se ha obtenido de diferentes maneras, lo que ha significado mejoras en el sistema y en los resultados.

Desde el punto de vista de la propuesta creada es necesario analizar los objetivos que se marcaron al principio para determinar si éstos se han cumplido o no:

- *"Se plantearán diferentes estrategias en el manejo de la información multilingüe, valorando la posibilidad o no de traducir y, en caso de hacerlo, valorando cuándo y qué se traduce"*. Se han planteado dos propuestas al problema de trabajar con tweets en diferentes idiomas y se ha obtenido una clara conclusión acerca de la conveniencia de ambas.
- *"Se plantearán diferentes estrategias a la hora de representar los tweets. Se analizará qué puede ser más o menos importante dentro del contenido de los tweets y si es mejor una representación tradicional para el clustering de tweets o representaciones novedosas, basadas en las características propias de los tweets. Se evaluará qué representación logra mejores resultados de clustering"*. Se han propuesto diferentes representaciones para los tweets a todos los niveles, tanto en el número de vectores con el que se representa cada tweet como la medida de ponderación utilizada, hasta como se fusionan las similitudes con representaciones de más de un vector.

- *"Se utilizarán todas las tecnologías y herramientas de software de código abierto que resulten más adecuadas al problema que queremos resolver"*. Se han comparado las librerías OpenNLP y StanfordNLP como software de PLN con conclusiones muy esclarecedoras, además de utilizar otras librerías de código abierto para diferentes tareas dentro del proyecto.
- *"Se procurará que el coste en tiempo de la solución propuesta sea razonable, teniendo en cuenta que serán necesarios diferentes procesamientos sobre los tweets"*. Se ha conseguido una aplicación de *clustering* multilingüe de tweets que funciona en un tiempo razonable cuando se trabaja con no más de 500 tweets.
- *"El diseño y la propuesta final deben ser fácilmente adaptables para funcionar con colecciones de tweets con más idiomas o con idiomas diferentes a los presentes en las colecciones que se van a utilizar en este trabajo"*. Se ha enfocado la implementación para que sea posible añadir más idiomas a los que acepta la aplicación actual y, además, se han añadido unas indicaciones generales en el capítulo 5 que ayudan en el caso de querer añadir más idiomas.

7.2 Conclusiones personales

A nivel personal la experiencia de desarrollar un trabajo como el llevado a cabo a través de la realización de este TFG me ha aportado lo siguiente:

- He puesto en práctica conocimientos con los que estaba bastante familiarizado, como la programación con lenguajes Java y Python, pero también he aprendido cosas nuevas como crear un servicio web con *RESTful web services* o cómo implementar la Lógica borrosa en un problema concreto.
- Para mí era completamente desconocido todo lo que tiene que ver con Procesamiento del Lenguaje Natural y cómo procesar información para un objetivo común, como puede ser el *clustering*.

- He trabajado con tweets como documentos, lo que me ha aportado también para saber que no todos los documentos se deben procesar de la misma forma aunque se utilicen para un mismo problema.
- El desarrollo de este me ha permitido ver cuáles son mis carencias y cuáles mis puntos fuertes como desarrollador de software y responsable de un proyecto.
- He sido único responsable de diseñar una herramienta de gran envergadura y he podido experimentar qué conlleva algo así.

7.3 Trabajos futuros

A lo largo del desarrollo del trabajo realizado ha sido posible recabar información e ideas que podrían ser de interés para la mejora de las técnicas estudiadas en este sistema, o que podrían servir como base para futuros proyectos:

- Explorar representaciones de tweets diferentes a las vectoriales.
- Explorar otras medidas de pesado en los vectores.
- Aumentar el número de categorías de datos en las que se separa el contenido de los tweets para representarlos *multivectorialmente*, haciendo más compleja la obtención de similitud final entre dos tweets pero permitiendo similitudes más adaptadas a la realidad.
- Expandir el contenido de los tweets con recursos externos, como extrayendo información de las URLS que contienen, mapeando términos a entradas en Wikipedia y/o Wordnet, etc…
- Revisión del sistema borroso para ver si es mejorable a nivel de los conjuntos de las variables lingüísticas o de las reglas de la base de conocimiento.
- Empleo de otros algoritmos de *clustering*.
- Estudio más en profundidad del método óptimo de fusión de clusters monolingües.
- Optimización del sistema para acelerar el proceso lo máximo posible.

Bibliografía

[1] ExpandedRamblings, estadísticas de Twitter http://expandedramblings.com/index.php/march-2013-by-the-numbers-a-few-amazing-twitter-stats/

[2] Apoorv A., Boyi X., Illia V., Owen R. and Rebecca P. *Sentiment Analysis of Twitter Data*. Proceedings of the Workshop on Languages in Social Media, pp. 30-38, 2011

[3] Fabian D., Eduard S. and Patrick E. *Building a Social Dictionary based on Twitter Data.* Tagung der Computerlinguistikstudierenden, 2014.

[4] Alan R., Sam C., Mausam and Oren E. *Named Entity Recognition in Tweets*. Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, pp. 1524–1534, 2011.

[5] Kathy L., Diana P., Ramanathan N., Md. Mosotfa, Ali P., Ankit A. and Alok C. *Twitter Trending topic Classification*. Proceedings of the 11th IEEE International Conference on Data Mining Workshops, pp. 251-258, 2011.

[6] Carlos V. and Antonio M. *Unsupervised topic Discovery in microblogging networks.* http://www.sciencedirect.com/science/article/pii/S0957417415002444

[7] Tushar Khot. *Clustering Twitter Feeds using Word Co-occurrence. http://pages.cs.wisc.edu/~tushar/projects/cs784.pdf*

[8] Antennuci D., Handy G., Modi A. and Tinkerhess M. *Classification of tweets via clustering of Hashtags.* Expert Systems with Applications, 42(17-18): 6472-6485, 2015.

[9] KD. Rosa, Shah R., Lin B., Gershman A. and Frederking R. *Topical clustering of tweets.* Proceedings of the SWSM'10.

[10] Ifrim G., Shi B. and Brigadir I. *Event detection in Twitter using Aggressive Filtering and Hierarchical Tweet Clustering.* Proceedings of the WWW Workshop SNOW 2014.

[11] SNOW Data Challenge 2014. http://www.snow-workshop.org/2016/challenge/

[12] Tsur O., Littman A. and Rappoport A. *Efficient Clustering of Short Messages into General Domains.* Proceedings of the 7th International Conference on Weblogs and Social Media, 2013.

[13] Jordi Colomer. *New Clustering proposals for Twitter based on the Louvain algorithm.* Trabajo Fin de Máster. UNED. 2014.

[14] Zubiaga A., Spina D., Martinez R. and Fresno V. *Real-Time Classification of Twitter Trends.* Journal of the Association for Information Science and Technology, 66(3): 462-473, 2015.

[15] Perez-Tellez F. and Pinto D. *On the Difficulty of Clustering Company Tweets* Proceedings of the 2nd International Workshop on Search and mining user-generated contents, pp. 95-102, 2010.

[16] Fresno V., Martinez R and Montalvo S. *Improving Web Page Clustering Through Selecting Appropiate Term Weighting Functions.* Proceedings of the 1st International Conference on Digital Information Management, pp. 511-518, 2006.

[17] Ken Beck, creador de XP https://es.wikipedia.org/wiki/Kent_Beck

[18] Java. http://www.oracle.com/technetwork/java/index.html

[19] Python. https://www.python.org/

[20] PHP. http://php.net/

[21] JSP. http://www.oracle.com/technetwork/java/javaee/jsp/index.html

[22] HTML. http://www.w3.org/html/

[23] XML. http://www.w3.org/XML/

[24] Java RESTful Services. https://docs.oracle.com/javaee/6/tutorial/doc/gijqy.html

[25] NUSOAP. http://sourceforge.net/projects/nusoap/

[26] StanfordNLP. http://nlp.stanford.edu/

[27] OpenNLP. https://opennlp.apache.org/

[28] Normalizador de tweets. http://alderamin.lsi.uned.es/normalizador/normalizar/normalizar.php

[29] Language Detection Library for Java. https://github.com/shuyo/language-detection

[30] Translator API for Yandex. https://github.com/rmtheis/yandex-translator-java-api

[31] APRO. http://www.apro.u-psud.fr/

[32] NetworkX. https://networkx.github.io/documentation/latest/index.html

[33] Distancia coseno. https://es.wikipedia.org/wiki/Similitud_coseno

[34] Enlace a Louvain, 100 millones de elementos. https://perso.uclouvain.be/vincent.blondel/research/louvain.html

[35] Enlace a mega con aplicación web CMT: http://bit.ly/CMTURJC

[36] Lexiconista. http://www.lexiconista.com/datasets/lemmatization/

[37] RepLab. http://nlp.uned.es/replab2013/

[38] CLEF. http://www.clef-initiative.eu/