Análisis de sentimientos en Twitter: El bueno, el malo y el >:(

Martín Becerra
Facultad de Matemática, Astronomía, Física y Computación
Universidad Nacional de Córdoba
Córdoba, Argentina
Email: mrtnbcrr@gmail.com

Resumen

Analizando los grandes volúmenes de datos generados en redes sociales sobre la opinión pública acerca de diferentes temáticas puede resultar en valiosos descubrimientos. Estas actividades son costosas de realizar manualmente, requieren de muchos recursos humanos y tiempo. Los sistemas de análisis de sentimientos y algoritmos de minería de datos han resultado ser de gran utilidad para poder obtener una percepción general de los temas de interés y la opinión sobre los mismos. En este trabajo proponemos analizar un conjunto de datos usando un clasificador de sentimientos para etiquetar publicaciones realizadas por usuarios de redes sociales en conjunto con algoritmos de clustering para poder detectar cuales son las temáticas sobre las cuales se expresan opiniones. Proponemos utilizar una base de 2000 reseñas de películas etiquetadas como positivas y negativas para luego entrenar un clasificador SVM de sentimientos. Luego utilizamos Twitter para recolectar y clasificar los sentimiento de 122 mil publicaciones relacionadas a un acontecimiento televisivo de gran convocatoria. También utilizamos el algoritmo de clustering K-Means para obtener un vistazo general sobre los temas y un aproximación del sentimiento asociado a estos. Finalmente, a través de una visualización interactiva, vemos los resultados de este procesamiento.

1. Introducción

Con el surgimiento de la internet 2.0, los usuarios tienen la posibilidad de generar su propio contenido y compartirlo públicamente con mayor facilidad. En este auge, las redes sociales han cobrado gran popularidad, en particular la plataforma de microblogging Twitter que permite a sus usuarios compartir mensajes de texto en 140 caracteres con sus familiares, amigos y seguidores. Diariamente se publican más de 500 millones de mensajes, comúnmente llamados tweets. Debido a que la principal razón de estas publicaciones es expresar el punto de vista y la opinión de los usuarios, resultan ser de gran interés para ser analizados.

Para la explotación de todos estos datos que circulan públicamente en la web podemos realizar diversas tareas que nos permitan extraer información útil de las opiniones. Esta área de trabajo, denominada como minería de opiniones, se enfoca en el tratamiento automático de textos en los cuales se ven reflejados la opinión, los sentimientos, las emociones y las actitudes de las personas hacia ciertos temas y sus aspectos.

Conocer lo que sus usuarios opinan puede tener diversas aplicaciones como, por ejemplo, recomendar productos ó determinar a qué candidato político se votará en las próximas elecciones.

Un objetivo especialmente interesante desde el punto de vista de extracción de información y representación del conocimiento es clasificar positiva o negativamente, según la opinión de distintos usuarios, los diferentes aspectos de una entidad y conocer qué motiva tales opiniones. Poder generar una solución a este problema tan complejo requiere gran conocimiento del dominio y que pueda ser trasladado a la solución de software propuesta, con un gran esfuerzo de desarrollo.

Sin embargo, para poder realizar este tipo de análisis nos encontramos con características del lenguaje natural que hacen que sea un área de investigación llamativa. Se presentan con problemas de ambigüedad o vaguedad semántica que se ven reflejados, por ejemplo, al momento de detectar el sarcasmo y las intenciones del autor de una oración. [1]

También, entre los algoritmos más comunes están aquellos que utilizan un conjunto de palabras del lenguaje, llamado lexicón [2], compuesto por palabras positivas (bueno, genial, excelente) y palabras negativas (feo, desagradable, horrible) para poder etiquetar el sentimiento de una oración. Dependiendo del contexto en el cual se empleen estas palabras pueden tener distinto significado, incluso pueden no expresar un sentimiento en particular. Esto hace que sea necesario el uso del sentido común para poder reconocer el sentimiento que transmite la oración.

A veces, no solo las palabras sino también las mismas expresiones pueden tener sentimientos asociados en contextos diferentes. Por ejemplo el comentario “No emite sonido alguno” podría considerarse positivo cuando se habla de un auto nuevo, pero también podría indicar negativo si se hablase de un equipo de música. Esto hace necesario saber que estamos analizando y las implicaciones particulares del tema. [3], [4]

Lo mismo ocurre con oraciones que solo presentan hechos: el conocimiento específico del dominio permite la extracción de opiniones por más de que no se transmita un sentimiento positivo o negativo. [5].

Para poder obtener una estimación de forma rápida de cómo es la opinión pública acerca de un tema podemos agrupar en clusters opiniones similares que hablen de un mismo aspecto. Si bien es una solución menos costosa para aproximarse al análisis de sentimientos basado en aspectos, esta solución no será tan adecuada comparada con un sistema donde se hayan inyectado grandes cantidades de conocimiento sobre el dominio, pero es una alternativa para hacer una primera aproximación exploratoria sobre los datos.

Finalmente, monitorear grandes volúmenes de datos e identificar las porciones más relevantes para extraer descubrimientos de forma resumida es un procedimiento costoso para ser realizado manualmente. Para esto es común realizar un análisis exploratorio de los datos mediante visualizaciones.

Las visualizaciones nos permiten resumir grandes volúmenes de datos en representaciones gráficas. Posteriormente, un experto puede interpretarlos rápidamente y realizar conclusiones que engloben la totalidad de los datos. Luego, puede tomar una decisión basada en la información recolectada.

En este trabajo proponemos estudiar las opiniones sobre un acontecimiento televisivo de gran convocatoria llamado “Los Oscars”. Realizaremos un proceso de adquisición de datos bajo la etiqueta “#oscars” de las publicaciones que se compartieron. Para poder aplicar los distintos algoritmos que son necesarios para poder analizar las publicaciones en primer lugar debemos transformarlos, mediante un proceso comprendido por varias

fases, en una matriz que represente la información relevante de los datos. Posteriormente, utilizando el algoritmo K-Means de clustering agrupamos en temas aquellas opiniones similares entre sí. Para complementar este resultado, entrenamos un clasificador de sentimientos para etiquetar cada publicación con el sentimiento que expresa y, así, poder tener una valoración general de los usuarios hacia el tema. Finalmente, navegamos los datos recolectados y los resultados de una manera rápida y efectiva utilizando diferentes visualizaciones.

Este trabajo se organiza de la siguiente manera: primero, en la sección 2 describimos cómo se recolectaron y procesaron los datos. Luego, en la sección 3, definimos el análisis de sentimientos, los distintos tipos de análisis que se pueden realizar y sus problemáticas más comunes. Continuamos enfocándonos en los algoritmos de clustering y cómo utilizarlos para realizar un análisis exploratorio de los datos en la sección 4. Finalmente, presentamos las conclusiones a las cuales llegamos en la sección 5.

2. Adquisición y procesamiento de los datos

Para este trabajo, utilizamos la red social *Twitter* que permite enviar mensajes cortos de hasta 140 caracteres, llamados *tweets*. Comúnmente utilizado por sus usuarios para compartir puntos de vista y opiniones con sus familiares, conocidos y seguidores. Al día de la fecha, Twitter cuenta con más de 332 millones de usuarios quienes generan más de 500 millones de tweets al día. Es uno de los 10 sitios web más visitados del mundo y fue descrito como "el *SMS* de la internet".

Es importante mencionar que los usuarios pueden agrupar *tweets* por tema utilizando *hashtags*, palabras o frases que utilizan el prefijo *"#"*, o también mencionar usuarios para comenzar una conversación con el símbolo *"@"* acompañado del nombre de un usuario. Esto es lo que nos permitirá realizar la extracción y el posterior análisis de un tema que nos parezca relevante.

2.1. Adquisición de datos

Para poder recolectar los datos fue necesario crear una aplicación de Twitter[1] para, así, poder hacer uso de la Search API (interfaz de programación de aplicaciones de búsqueda). Luego, utilizamos la librería Tweepy[2] para implementar un programa que nos permitiera recolectar los *tweets* sobre el tópico que consideráramos de interés. Para este trabajo decidimos analizar la edición de los premios Óscars 2016, un acontecimiento televisivo popularmente comentado por los televidentes. Bajo la etiqueta "#oscars", logramos recolectar más de 120 mil tweets, caracterizados principalmente por contener opiniones de usuarios, spam y publicidad, entre otras.

Al momento de trabajar con la API de Twitter fue necesario tener en cuenta ciertas limitaciones de la misma. Los límites de frecuencia de la interfaz sólo permitían realizar hasta 180 peticiones cada 15 minutos [6]. También, hay que tener en cuenta que Twitter filtra gran cantidad de los tweets publicados para que los resultados de nuestras búsquedas sean de mayor calidad, por lo que no está disponible la totalidad de publicaciones que se realizaron sino aquellas que Twitter considera que son más relevantes para el usuario.

1. https://apps.twitter.com/ - Twitter Application Manager
2. http://www.tweepy.org/ - Librería de Python para acceder la Twitter API.

2.1.1. Estadísticas del dataset "#oscars".

- Idioma: Inglés
- Tweets: 122.443
- Tweets únicos: 38.055
- Palabras en tweets únicos: 406.311
- Palabras únicas: 72.792
- Palabras que se repiten en más de 100 tweets: 452
- Palabras que se repiten en más de 1000 tweets: 26

2.2. Procesamiento de los datos

El conjunto de datos resultante de la etapa de recolección de datos es una colección de tweets que contienen un mensaje, información sobre el usuario que lo publicó y metadatos asociados como, por ejemplo, la fecha de creación, su identificador único y la cantidad de veces que otros usuarios compartieron la publicación. Es necesario que los datos pasen por un proceso comprendido por varias etapas, también conocido como *pipeline*, en donde la entrada de cada etapa es la salida de la anterior transformando y filtrando los diferentes atributos. Esto nos permitirá representar la información relevante de los datos en una matriz, sobre la cual podremos aplicar técnicas de clustering.

A continuación, explicaremos el pipeline diseñado para este trabajo utilizando un tweet a modo de ejemplo para poder ver las transformaciones.

2.2.1. Transformación de tweets a texto. En primer lugar, la colección de tweets es transformada en nuestro corpus extrayendo solo el texto de cada publicación. Un tweet es un objeto complejo con muchas propiedades pero lo que nos interesa es el mensaje que el usuario escribió en forma de téxto. Si bien al realizar este paso perderemos información sobre el autor del tweet y los metadatos propios de la publicación, conservamos una referencia para poder vincular el resultado final de nuestro procesamiento con la publicación original. El producto de esta etapa es un listado de documentos (oraciones) llamado corpus. En la figura 1 podemos ver a modo de ejemplo la transformación de tres publicaciones a un listado que contiene solo el mensaje de las mismas.

```
['Spotlight, Óscar a Mejor Guion Original
http://ow.ly/YUmRu #Oscars', '#DaveGrohl también paso
por los #Oscars tocando #Blackbird de los #Beatles Mira:
http://ow.ly/YUmHr', '#LeonardoDiCaprio #Oscars te lo
merecías! Eres de mis actores favoritos!']
```

Figura 1. Ejemplo de transformar tres *tweets* sobre los *#oscars* a una lista de documentos que contiene los mensajes de cada publicación.

2.2.2. Tokenización. En segundo lugar, cada documento de nuestro corpus es transformado en un listado de términos llamados tokens. Esta representación de datos también es conocida como bolsa de palabras (*bag of words*). Los tokens son cadenas de caracteres entre espacios en blanco o puntuación, pero no siempre es así, como por ejemplo en el caso de las abreviaturas. El conjunto total de palabras utilizadas, distintas y únicas, es el vocabulario del corpus.

En este paso, también se filtran las palabras que no tienen una semántica referencial clara, como artículos, pronombres, preposiciones, etc. *stop words* que son muy frecuentes en el lenguaje. También es necesario eliminar aquellos tokens propias del glosario de Twitter como *"RT"*, *"HO"* y *"HT"* que no agregan valor al análisis que deseamos realizar. En la figura 2 podemos ver el resultado de realizar la tokenización sobre el ejemplo de la subsección anterior.

```
[['spotlight', 'oscar', 'mejor', 'guion', 'original',
'http://ow.ly/YUmRu', '#oscars'], ['#davegrohl', 'paso',
'#oscars', 'tocando', '#blackbird', '#beatles', 'mira',
'http://ow.ly/YUmHr'], ['#leonardodicaprio', '#oscars',
'merecias', 'mis', 'actores', 'favoritos']]
```

Figura 2. Ejemplo de tokenizar los tres *tweets* transformados en la figura anterior

Al momento de trabajar con Twitter nos encontramos con términos como las menciones ("@" + nombre de usuario), los hashtags ("#" + tema) y las direcciones ('http://...'). Estos son considerados tokens especiales, los cuales consideraremos como un único token.

2.2.3. Stemming. Muchas veces diferentes tokens pueden hacer referencia al mismo concepto, ya que pueden ser representados por variantes morfológicas de una misma familia de palabras. Por ejemplo, podemos representar "escribo", "escribíamos" y "escribimos", en su raíz como "escrib" ya que tienen un significado similar y derivan del mismo verbo. Esto nos permite relacionar aquellos tweets que contienen palabras con similitud semántica gracias a su raíz, lo que luego facilitará el proceso de clustering. A este proceso de reducción de tweets lo identificaremos como ***stem*** o ***stemming***.

Si bien puede resultar de utilidad, hay que tener en cuenta que pueden surgir errores a partir de este proceso, dependiendo del algoritmo que utilicemos y nuestro corpus. Un posible error es el caso en que dos palabras con distintos significados sean reducidas a una misma palabra cuando no debería ser así. Como por ejemplo:

casa, casorio, caso $\rightarrow$ cas

Otro tipo de error se da cuando dos palabras deberían ser reducidas a una misma forma normal pero se considera que provienen de diferentes palabras. Como por ejemplo:

alumnus $\rightarrow$ alumnu,
alumni $\rightarrow$ alumni,
alumna/alumnae $\rightarrow$ alumna.

La solución a este problema es un proceso de reducción a forma canónica que incorpore más información lingüística, proceso conocido como *lematización*. En este trabajo optamos por no implementar esta opción porque resultaba costoso instalar la herramienta que realizaba esta tarea, y también debido a que los tweets tienen muchas palabras no estándares que tampoco pueden tratar bien los lematizadores.

2.2.4. Vectorización. El próximo paso consiste en generar una matriz rala (*sparse* matrix), transformando cada lista de palabras en un vector en un espacio Euclídeo, donde cada columna es una característica. Las características son principalmente palabras del vocabulario extraído de la tokenización. De esta forma, podemos considerar cada palabra del vocabulario como una columna, o podemos obtener representaciones más detalladas si consideramos como posibles características secuencias de palabras, llamadas ***n-gramas***, que han ocurrido en el texto. Los n-gramas pueden ser secuencias de una palabra (unigramas), de dos palabras (bigramas), de tres palabras (trigramas), y así sucesivamente. En la figura 3 podemos ver como nuestros tweets de ejemplo generan un vocabulario con unigramas y bigramas, con los cuales luego podremos representar nuestros datos.

```
['#beatles', '#beatles mira', '#blackbird', '#blackbird
#beatles', '#davegrohl', '#davegrohl paso',
'#leonardodicaprio', '#leonardodicaprio #oscars',
'#oscars', ..., 'paso', 'paso #oscars', 'spotlight',
'spotlight oscar', 'tocando', 'tocando #blackbird']
```

Figura 3. Ejemplo del vocabulario generador luego de tokenizar tres *tweets* utilizando unigramas y bigramas

Cada tweet se representa en una fila, y los valores que toma en cada columna están determinados por la ocurrencia de las palabras de cada columna en el tweet. De esa forma, la celda *i,j* de la matriz tendrá el valor de la ocurrencia del n-grama representado en la columna *i* en el tweet *j*.

La forma más simple de asignar valores a las celdas es con valores binarios: 1 si el *n*-grama representado por la columna *i* ocurre en el tweet *j*, y 0 si no ocurre. Teniendo un vocabulario grande, es de esperar que la mayor parte de *n*-gramas no ocurra en un tweet. Por lo tanto las matrices serán ralas, es decir, con gran cantidad de 0. A esta forma de representar los tweets la identificaremos como ***bin*** o ***binarización***. La figura 4 nos muestra como podríamos representar los tweets de la figura 2 utilizando el vocabulario definido en la figura 3.

```
[[0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0],
[1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1],
[0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0]]
```

Figura 4. Representación de los *tweets* de la figura 1 utilizando binarización

Donde el vocabulario asociado a las columnas es:

```
['#beatles', '#blackbird', '#davegrohl',
'#leonardodicaprio', '#oscars', 'actores', 'favoritos',
'guion', 'http://ow.ly/YUmHr', 'http://ow.ly/YUmRu',
'mejor', 'merecias', 'mira', 'mis', 'original', 'oscar',
'paso', 'spotlight', 'tocando']
```

Al momento de vectorizar los documentos, podemos elegir entre diferentes opciones que nos permiten reducir el volumen del corpus y obtener los resultados más relevantes.

Una palabra que aparece en más de la mitad de los tweets es redundante y no brinda mucha información para diferenciar un tweet de otro. Para esto es útil definir un umbral de frecuencia para determinar qué características deben ser incluidas en la matriz, de tal forma que aquellas características que son muy frecuentes sean ignoradas, ya que pueden ser consideradas *stop words* propias del corpus. Tampoco debemos incluir aquellas características que son poco frecuentes ya que no proveen generalizaciones sobre tendencias en los textos. Identificaremos con ***min df*** y ***max df*** a las cotas inferior y superior, respectivamente, del umbral de frecuencia, describiendo así las palabras que descartamos porque ocurren menos veces que ***min df*** o más veces que ***max df***.

2.2.5. Frecuencia. La representación de la co-ocurrencia de tweets y palabras con valores binarios es demasiado gruesa y no captura algunas diferencias importantes, como por ejemplo si una palabra ocurre muchas veces en un tweet. Para eso, podemos asignar valores de valores de frecuencia a las celdas: n si el n-grama representado por la columna i ocurre n veces en el tweet j, y 0 si no ocurre. A esta forma de representar los tweets la identificaremos como ***tf*** o ***frecuencia de términos***. La figura 5 muestra un ejemplo de como se representaría un tweet utilizando frecuencia de terminos resaltando las ocurrencias de cada término del vocabulario.

```
'Esos que dicen México ganó dos #Oscars ayer, no, México no
ganó señores #Inarritu y #Lubezki si'
```

Sería representado como:
```
[1, 1, 1, 1, 1, 1, 2, 2, 1]
```

Con el siguiente vocabulario:
```
['#inarritu', '#lubezki', '#oscars', 'ayer', 'dicen',
'esos', 'gano', 'mexico', 'señores']
```

Figura 5. Ejemplo un *tweet* luego de la etapa de frecuencia de términos

Sumado a esto, para producir resultados aún más relevantes, podemos limitar la máxima cantidad de características(columnas) con las cuales representaremos los tweets. Esto se puede llevar a cabo considerando solo aquellas n primeras ordenadas de mayor a menor según su frecuencia en todo el corpus. De esta manera, representaremos los tweets solo con aquellos n términos que sean más relevantes. A esta forma de reducir el volumen de tweets la identificaremos como ***max features*** o ***límite de características***.

2.2.6. Frecuencia de términos - frecuencia inversa de documento. La representación de la co-ocurrencia de tweets y palabras con valores de frecuencia nos permite conocer el grado de importancia que posee una característica para identificar un tweet según cuántas veces esta ocurre en el mismo.

Sin embargo, aquellas características que son muy frecuentes en el conjunto de todos los tweets lleva a que no sean tan relevantes al momento de identificarlos. Para evitar esto, podemos incorporar un factor de frecuencia inversa de documento que atenúe el peso de las características que ocurren con mayor frecuencia en la colección de tweets e incrementa el peso de aquellas características que ocurren menos seguido. En la figura 6 podemos ver como representaríamos tres tweets distintos luego de distribuir el peso de las características.

```
[['spotlight', 'oscar', 'mejor', 'guion', 'original',
'http://ow.ly/YUmRu', '#oscars'], ['#davegrohl', 'paso',
'#oscars', 'tocando', '#blackbird', '#beatles', 'mira',
'http://ow.ly/YUmHr'], ['#leonardodicaprio', '#oscars',
'merecias', 'mis', 'actores', 'favoritos']]
```

Se representa como:

```
[[0, 0, 0, 0, 0.23, 0, 0, 0.39, 0, 0.39, 0.39, 0, 0, 0,
0.39, 0.39, 0, 0.39, 0.], [0.36, 0.36, 0.36, 0, 0.21, 0, 0,
0, 0.36, 0, 0, 0, 0.36, 0, 0, 0, 0.36, 0, 0.36], [0, 0, 0,
0.43, 0.25, 0.43, 0.43, 0, 0, 0, 0, 0.43, 0, 0.43, 0, 0, 0,
0, 0]]
```

Figura 6. Los 3 *tweets* de ejemplo luego del proceso de *tf-idf*

Esta manera de representar los tweets la identificaremos como ***tf*idf*** o ***frecuencia de términos - frecuencia inversa de documento***. Es importante notar que tanto ***tf*** como ***tf*idf*** implican la definición de un umbral, el cual, dependiendo de nuestro corpus, nos ayudará a conseguir mejores resultados. Por lo tanto, será necesario realizar distintas pruebas con distintos umbrales hasta poder obtener resultados representativos.

3. Análisis de sentimientos

El análisis de sentimientos busca extraer opiniones, sobre una determinada entidad y sus diferentes aspectos desde el lenguaje natural de los textos. Esto se realiza de manera automática utilizando algoritmos para clasificación. Las opiniones son clasificadas según el sentimiento que transmiten, es decir, como positivas, negativas o neutras.

Su importancia está en que nuestra percepción de la realidad, y así también las decisiones que tomamos, es condicionada en cierta forma por cómo otras personas ven y perciben el mundo. Es por esto que, desde un punto de vista de utilidad, queremos conocer las opiniones de otras personas sobre temas de interés, ya que tienen diversas aplicaciones como recomendar productos y servicios, determinar a qué candidato político se votará en las próximas elecciones o incluso medir la opinión pública ante la medida tomada por una empresa o un gobierno [7].

3.1. Tipos de análisis de sentimiento

A la hora de extraer esta información, hay una gran variedad de métodos y algoritmos dependiendo del nivel de granularidad del análisis que queramos llevar a cabo. Se distinguen tres niveles: nivel de documento, de oración o de aspecto. El análisis a nivel de documento determina el sentimiento general expresado en un texto, mientras que el análisis a nivel de frase lo especifica para cada una de las oraciones del texto.

Sin embargo, estos dos tipos de análisis no profundizan en detalle el elemento que a las personas les gusta o no. No especifican sobre qué es la opinión, ya que considerando la opinión general de un objeto como positiva (o negativa) no significa que el autor tenga una opinión positiva (o negativa) de todos los aspectos de dicho objeto.

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

3.1.1. Análisis a nivel de documento. Considerada como una de las tareas más simples de la minería de opiniones, el análisis a nivel de documento apunta a clasificar la opinión de un documento, en este caso un tweet. Esta tarea no considera los detalles en cuanto a entidades o aspectos, sino que considera el documento como un todo, el cual será etiquetado como positivo o negativo.

Esta puede ser considerado como una tarea tradicional de clasificación de texto, donde las clases son las diferentes orientaciones en cuanto a los sentimientos. No obstante, para asegurar que este tipo de análisis tenga sentido asumimos que cada documento expresa una única opinión sobre una única entidad. Si bien esto puede parecer una limitación, porque en un tweet uno podría expresar más de una opinión hacia distintas entidades, en la práctica produce resultados positivos, ya que los usuarios suelen enfocarse en un único aspecto en cada tweet. Seguramente en otros contextos, o si no estuviera limitado el largo de los mensajes, sería una buena idea considerar sistemas de análisis más complejos que permitan realizar un análisis con mayor granularidad.

Considerando esto y tomando como clases *"positivo"* y *"negativo"*, cualquier método de clasificación de texto basado en aprendizaje automático se puede aplicar directamente; la omisión de la clase *"neutro"* ayuda a simplificar el problema. Las palabras que conforman las opiniones son el factor determinante en el análisis de sentimientos también es una buena opción utilizar métodos de aprendizaje basados en el uso de lexicones. Estos son diccionarios que contienen listados de palabras etiquetadas con el sentimiento asociado correspondiente, en algunos casos por un valor que también indica la intensidad del mismo. Luego, dada una opinión, simplemente podemos buscar el valor de todas las palabras que la componen y sumar el valor asociado para finalmente clasificar la opinión como positiva o negativa según corresponda [8], [9]. También se podría considerar extender el sistema para contemplar negaciones y palabras que intensifican el sentimiento [10].

Para este trabajo decidimos utilizar un acercamiento más simple entrenando un clasificador de sentimientos utilizando reseñas de películas. Cuando asociamos la valoración de cada documento a una polaridad positiva o negativa es posible generar un conjunto de datos etiquetado automáticamente. Utilizamos un conjunto de datos públicos ya etiquetados de esta manera, en la sección 3.3 lo detallamos.

3.1.2. Clustering como forma de acercarse a los aspectos. La identificación y extracción de aspectos sobre las cuales hablan las opiniones es una tarea compleja. Por ende, decidimos obtener un acercamiento utilizando algoritmos de clustering sobre el conjunto de tweets. En primer lugar, recolectamos un conjunto de tweets que tratan de un mismo tema (en este caso, los #oscars), y luego tratamos de identificar en este conjunto los diferentes aspectos sobre este tema.

De esta manera, podemos hacer un análisis exploratorio de los datos como si cada uno de estos subtemas fuese un aspecto propio de la entidad #oscars. Luego, podemos aproximarnos a la clasificación de cada aspecto etiquetandolo como positivo o negativo según la opinión de los tweets que contiene cada cluster y, consecuentemente obtener una apreciación general del total del corpus. Es una forma rápida y poco costosa de realizar un análisis exploratorio de un gran conjunto de datos sin requerir interacción humana en el ciclo de procesamiento.

3.2. Selección de un clasificador de sentimientos

El rol de un clasificador de sentimientos es asignar a un tweet un sentimiento positivo o negativo de forma automática. En nuestra prueba de concepto, queremos asignar sentimientos positivos o negativos a los tweets del corpus de #oscars. Para ello, implementamos un clasificador de sentimientos basado en aprendizaje automático, y evaluamos diferentes alternativas.

3.2.1. Metodología de evaluación. Para poder saber el extento de eficiencia que tienen las predicciones de nuestro clasificador evaluamos su rendimiento mediante una validación cruzada de K iteraciones.

Para evaluar el rendimiento de un clasificador, el *accuracy* del mismo, medimos la proporción de casos clasificados correctamente sobre el total de todos los casos a clasificar. La clasificación correcta de los casos es conocida porque los casos de evaluación, así como los de entrenamiento, han sido anotados previamente.

$$Accuracy = \frac{\text{Total de aciertos}}{\text{Total de casos}}$$

En la validación cruzada de *K* iteraciones o *K-fold cross-validation* los datos anotados se dividen en *K* subconjuntos. Uno de los subconjuntos se utiliza como datos de prueba y los (*K-1*) restantes como datos de entrenamiento. El proceso de validación cruzada es repetido durante *K* iteraciones, con cada uno de los posibles subconjuntos de datos de prueba. Finalmente, se realiza la media aritmética de los resultados de cada iteración para obtener un único resultado. Este método es muy preciso puesto que evaluamos a partir de *K* combinaciones de datos de entrenamiento y de prueba. Sin embargo, aun así tiene una desventaja con respecto al bajo nivel de rapidez que posee desde el punto de vista computacional. En la práctica, la elección del número de iteraciones depende de la medida del conjunto de datos. Comunmente, se utiliza la validación cruzada de 10 iteraciones (*10-fold cross-validation*). [11]

3.3. Análisis de resultados de clasificación de sentimientos

Para este trabajo trabajamos con el dataset utilizado en Pang y Lee (2004) [12]. Está compuesto por 2000 reseñas de películas clasificadas con puntajes, la mitad son positivos y la otra mitad son negativos, con un límite de 20 reseñas por autor por clase con un total de 312 autores.

Pasamos las opiniones por un preproceso simple de *tf*idf*. Con este conjunto de datos vectorizados entrenamos diferentes clasificadores provistos por la librería open source de aprendizaje automático *Scikit-Learn* para Python [13], y comparamos su efectividad. Debido a que los clasificadores tienden a tener diferentes parámetros para poder configurarlos, en vez de probar manualmente las alternativas que tenemos para cada parámetro, hicimos una búsqueda exhaustiva probando todas las combinaciones posibles utilizando la herramienta *Grid Search* de Scikit-Learn.

Comparamos tres tipos de clasificadores, una máquina de vectores de soporte (*Support Vector Machine ó SVM*); un árbol de decisión; y un clasificador bayesiano naïve con distribución multinomial. Todos estos clasificadores son provistos por la librería Scikit-Learn. A continuación detallamos los parámetros que obtuvieron un

mejor rendimiento.

Para la *SVM* obtuvimos el mayor rendimiento, de 87.15 %, utilizando una penalidad C de 100 y utilizando la función de pérdida *"hinge"*. Sólo obtuvimos un rendimiento del 66.85 % utilizando árboles de decisión con el criterio *"entropy"* y con un máximo de 6 niveles de profundidad. Finalmente, obtuvimos un rendimiento de 84.60 % con el clasificador bayesiano naïve con distribución multinomial y con un factor *alpha* de 0.1. Por lo tanto, en nuestra arquitectura del sistema decidimos quedarnos con el clasificador SVM para realizar la asignación de sentimiento a los tweets de nuestro corpus.

4. Clustering

Al momento de analizar los tweets que recolectamos, nos encontramos con el problema de que las opiniones de los usuarios involucran un conjunto de temas muy variados. Para facilitar el análisis exploratorio de los datos, utilizamos métodos que nos segmentan el conjunto de tweets por subtemas y, posteriormente, podemos clasificar cada uno de ellos para ver si se expresa un sentimiento positivo o negativo en general hacia el mismo.

Para poder explicar los métodos de agrupamiento que utilizamos, primero debemos introducirnos en el aprendizaje automático supervisado y no supervisado.

4.1. Aprendizaje automático supervisado y no supervisado

En general, el problema del aprendizaje automático es que intenta predecir propiedades desconocidas de los datos a partir de información no estructurada suministrada en forma de ejemplos. Este problema se puede subdividir en dos categorías.

En primer lugar, en un problema de aprendizaje supervisado donde conocemos cuáles son los resultados deseados a partir de los datos de entrenamiento, y nos permite tener una idea de la relación que existe entre éstos. El objetivo del aprendizaje supervisado es crear una función capaz de predecir el valor correspondiente a cualquier objeto de entrada válida después de haber visto los datos de entrenamiento. Para ello, tiene que generalizar a partir de los datos presentados a situaciones no vistas previamente. Esto es realizado por los clasificadores de sentimiento que hemos entrenado y evaluado en el capítulo anterior.

Por otro lado, el aprendizaje no supervisado es otro tipo de aprendizaje automático que nos permite trabajar con un problema sobre el cual no tenemos información acerca de la solución. Se distingue del aprendizaje supervisado, ya que no cuenta con un resultados etiquetados manualmente. Se genera un modelo a partir de las relaciones, descubriendo así una estructura presente entre los datos.

El objetivo de este tipo de aprendizaje podría ser ver una versión resumida de los datos para que un experto pueda interpretarlos rápidamente; o bien descubrir patrones significativos presentes en los datos. Este tipo de uso se llama análisis exploratorio de datos.

En la siguiente sección describimos en detalle la técnica de aprendizaje no supervisado que vamos a estar usando, llamada agrupamiento o *clustering*.

4.2. Fundamentos de clustering

Los cerebros humanos son buenos encontrando regularidades y patrones en los datos. Una forma de hacerlo es agrupar aquellos objetos que son similares entre sí. Por

ejemplo, los biólogos descubrieron que la mayoría de los seres vivos caen en una de dos categorías: seres vivos de distintos colores que pueden moverse y seres vivos que de color verde que no pueden moverse. La primer categoría es llamada animales, la segunda, plantas. Llamamos a esta operación de agrupamiento *clustering*.

Supongamos que un biólogo encuentra un nuevo ser vivo color verde que no se había visto antes. Gracias a su modelo mental de lo que es una planta y que es un animal podrá predecir ciertos atributos del ser verde: que no se moverá, que si lo toca se puede raspar o incluso envenenar, que se podría enfermar si lo come. Todas estas predicciones, si bien no son necesariamente ciertas, son útiles porque permiten al biólogo sacar más partido de sus recursos de análisis.

Los métodos de clustering tratan de establecer grupos de objetos que están en un espacio *n*-dimensional, de forma que los objetos que están más cercanos queden agrupados en un mismo grupo y que los distintos grupos estén lejanos entre sí.

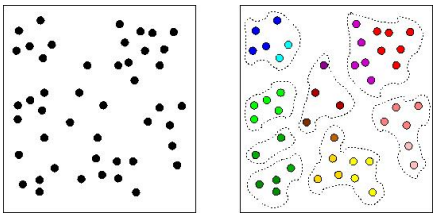


Figura 7. Resultado de realizar clustering en un conjunto de datos utilizando 8 clusters. Los colores se utilizan para diferenciar los clusters entre sí.

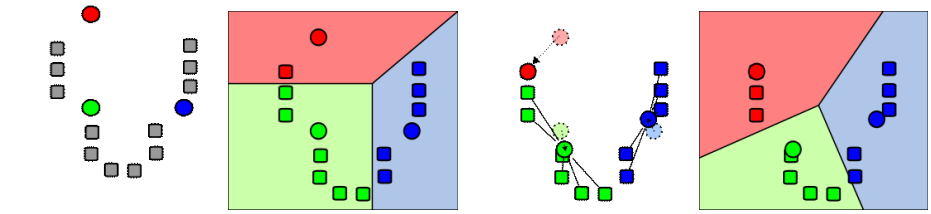
En la figura 7 podemos ver cómo dos o más puntos son agrupados en el mismo cluster ya que están uno cerca del otro. Los puntos son objetos en un espacio *n*-dimensional, representados como vectores. Asimismo, si la definición de distancia que tienen los elementos cambia, entonces también la manera en que están conformados los clusters.

Se pueden aplicar métodos de clustering para obtener grupos de objetos semejantes en la realidad. Para ello, necesitamos representar a los objetos como vectores y trasladar la noción de semejanza a una similaridad en un espacio *n*-dimensional.

El procesamiento de vectorización de datos que mencionamos en la sección 2.2 es esencial para poder representar los tweets como puntos en un espacio *n*-dimensional, es decir, como vectores.

En nuestro problema, tener clusters de tweets de un determinado dominio nos puede resultar útil de la misma forma. Al momento de recolectar nuevos tweets sobre un tema, podemos asignarlos a uno de los clusters que ya hemos identificado previamente en nuestro corpus sobre el mismo tema, asumiendo así que comparte las mismas propiedades que los elementos en ese cluster. Esto es de utilidad para conocer mejor nuestros datos y poder tomar mejores decisiones. [14]

Existen varios tipos de algoritmos de clustering, entre ellos, algoritmos por densidad, por distribución y por agrupamiento basado en centroides. En este trabajo utilizamos el algoritmo de Lloyd de agrupamiento basado en centroides, a menudo referido como *"K-means"*.



(a) Se generan *k* centroides iniciales (en este caso *k*=3) al azar

(b) Se crean *k* clusters asociando cada instancia al centroide más cercano. Las particiones representan un diagrama Voronoi generado por los centroides

(c) El centroide de cada uno de los k clusters se calcula nuevamente

(d) Se repiten los pasos (b) y (c) hasta converger

Figura 8. Ejemplo de clustering utilizando K-Means con *k*=3 clusters. Las figuras circulares representan a los centroides y los cuadrados a las diferentes instancias de los datos.

4.2.1. K-means. La idea principal es dividir *M* puntos de *N* dimensiones definiendo *K* centroides, y luego tomar cada punto de la base de datos y situarlo en la clase de su centroide más cercano, de tal forma para minimizar la suma de las funciones de distancia de cada punto en el cluster con respecto al centroide. Es decir:

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{\mathbf{x} \in S_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2$$

donde u_i es la media de los puntos en S_i.

El próximo paso es calcular nuevamente el centroide de cada grupo y volver a distribuir todos los objetos según el centroide más cercano. El proceso se repite hasta que ya no hay cambio en los grupos de un paso al siguiente.

K-means converge rápidamente, deteniéndose luego de que no haya habido cambios de una iteración a la siguiente, sin embargo no garantiza que siempre llegará a un mínimo global, por lo que se puede definir un máximo de iteraciones para evitar que el algoritmo sea más efectivo. [15]

La figura 8 muestra los pasos que lleva a cabo K-Means para encontrar una solución en 3 clusters en un espacio de 2 dimensiones requiriendo solo 2 iteraciones para converger. Las figuras circulares representan a los centroides, y los cuadrados, a las diferentes instancias de los datos.

Las técnicas basadas en centroides son apropiadas para datos que pueden ser representados en un espacio Euclídeo en donde es posible calcular la distancia entre un centroide y un elemento del conjunto.

4.3. Coeficiente de silhouette

Para identificar si los clusters están bien formados necesitamos saber si hay cohesión entre los elementos que pertenecen a un mismo cluster y qué tan separados están de aquellos elementos en otros clusters. Para esto, utilizamos el coeficiente de silhouette.

Se calcula, en primer lugar, tomando un punto *p* y calculando la distancia promedio al resto de los elementos de su mismo cluster. Llamaremos a este valor *a(p)*. Luego, para

el mismo punto p se calcula la distancia promedio con respecto a todos los elementos de otro cluster que no contengan a p. Repetimos esto buscando el mínimo para todos los clusters. Llamaremos a este valor *b(p)*.

Finalmente, el coeficiente de silhouette para p será

$$silhouette(p) = \frac{b(p) - a(p)}{max(a(p), b(p))}$$

Podemos tomar el promedio de todos los coeficientes de silhouette como una métrica del proceso de clustering que varía entre -1 y 1, donde un valor más alto indica mejor cohesión y separación entre los clusters.

4.4. Problemas habituales de clustering

Si bien K-means plantea buscar los mínimos cuadrados, resultando en un algoritmo bastante simple, tiene ciertos problemas y limitaciones con los cuales nos topamos a la hora de trabajar con los tweets recolectados.

En primer lugar, la elección del número de clusters adecuado. Si bien hay varios estudios acerca de cómo definir la cantidad apropiada de agrupaciones [16], [17], no hay una forma específica para determinar la cantidad de clusters en que deberíamos dividir nuestro corpus.

En segundo lugar, no siempre hay una estructura que se pueda reconocer en los datos. Si los datos no son naturalmente divisibles (o no lo son en las dimensiones en que los hemos representado o con la medida de distancia elegida o en el número de clusters elegido) al clasificarlos vamos a estar generando clusters que no representen una estructura significativa. En la figura 9 se puede ver como K-Means divide los objetos en 2 clusters aunque no exista tal división en los datos.

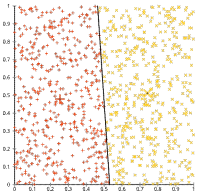


Figura 9. Ejemplo de correr K-Means con *k*=2 en un dataset con datos distribuidos de manera uniforme donde es evidente no existen grupos significativos en los datos.

Asimismo, también puede ocurrir que haya un cluster que tenga un volumen mucho mayor al resto de los clusters, conteniendo así a casi todos los elementos. Este problema es difícil de identificar verificando la cohesión entre clusters, pero lo podemos detectar si es que vemos que un cluster es órdenes de magnitud más grande que el resto. Identificaremos a este problema como catch-all cluster.

Por último, K-means puede converger en un mínimo local. Esto se debe que al inicializar los centroides, estos están distribuidos de tal forma que los resultados contradicen la estructura obvia en los datos.

Si bien esto se debe principalmente a que los centroides iniciales son elegidos al azar, hay diferentes formas de encarar este problema, entre ellas utilizar otros algoritmos como K-means++, que especifica un procedimiento para inicializar los centroides antes de que K-means comience [18] [19], o el método Forgy, que selecciona k elementos

pertenecientes al conjunto como centroides. Otra forma también utilizada es la de ejecutar varias veces K-means y que el resultado final sea aquel que minimice mejor la suma de los cuadrados para cada cluster.

4.5. Descripción de soluciones

Para este trabajo se utilizó *Scikit-Learn* [13]. Esta librería cuenta con una amplia variedad de herramientas para realizar tareas de minería y análisis de datos, que nos permitieron crear nuestro pipeline de procesamiento y clusterización sin tener que enfocarnos en la implementación de los algoritmos.

Para estos experimentos se utilizó un dataset compuesto por más de 122 mil tweets que compartían en común la etiqueta #oscars. En la sección 2.1.1 se pueden ver más estadísticas sobre los datos y también información sobre el procesamiento previo que realizamos para reducir el volumen del corpus.

A modo de referencia, describimos brevemente los diferentes parámetros que nos permitían configurar el pipeline a la hora de realizar la etapa de clustering. Podemos encontrar información más detallada sobre cada uno de ellos en la sección 2.2.

Etapas de pre-procesamiento.

- **Stemming (*stem*)**: Reducir cada token a su raíz, agrupando aquellos que hacen referencia al mismo concepto pero son representados por diferentes variantes morfológicas.
- **N-grams (*n-grams*)**: Representaciones más detalladas de secuencias de a lo sumo n palabras que han ocurrido en el texto como características.
- **Binarización (*bin*)**: Se representa marcando la ocurrencia de características en el tweet. 1 si el n-grama ocurre en el tweet, y 0 si no ocurre.
- **Frecuencia de términos (*tf*)**: Similar a bin, pero enumerando cuantas veces ocurre la caracteristica en el tweet.
- **Frecuencia de términos - frecuencia inversa de documento (*tf*idf*)**: A tf pero atenuando el peso de las características más ocurrentes.
- **Umbral de frecuencia (*max df y min df*)**: Definimos una cota inferior (*min df*) para filtrar aquellas características que son poco frecuentes. También definimos una cota superior (*max df*) para filtrar aquellas características que son más frecuentes en el corpus y pueden ser consideradas *stop words* del dominio.

Podemos agregar a esta configuración un valor ***K*** que será utilizado para que k-means defina el número de conjuntos (o clusters) en los que se dividirán los datos. Para hacer una búsqueda más exhaustiva para identificar la mejor configuración, definiremos un rango de de valores para *K* para cada configuración y finalmente compararemos cual fue la que obtuvo el mayor valor de coeficiente de silhouette.

Luego, llamaremos configuración a una 5-tupla de la forma:

”(min_df=x, max_df=y, K=z, n_grams=a,b), preprocessors=[...])”

Esta tupla sintetiza todos los valores que mencionamos previamente y también los métodos de preprocesamiento que utilizamos en un experimento.

Para definir para cada uno de estos parámetros que componen nuestra configuración, comenzamos por una etapa de exploración manual de los datos probando diferentes cotas para nuestro umbral de frecuencia y cómo éstas reducían el volumen de los datos. Notamos que el 87 % de las características se repetía hasta 5 veces en el corpus, mientras que sólo el 5 % aparecía más de 10 veces en el corpus, lo que reducía drásticamente el volumen de los datos dejando sólo aquellas características más relevantes. A partir de esto probamos diferentes valores para *max df* y *min df* cercanos a estas cotas.

Para el resto de los parametros (*stem*, *n-grams*, *bin*, *tf* y *tf*idf*), probamos diferentes combinaciones arbitrariamente para abarcar exhaustivamente todas las posibilidades para diferentes rangos de *K*.

Los resultados se muestran en la tabla 1 del anexo así como las diferentes soluciones utilizando diferentes configuraciones. Todas fueron realizadas con valores de *k* entre 4 y 9 clusters para K-Means. Cada configuración cuenta con un número de features, un mínimo de frecuencia (*min-df*), un máximo de frecuencia(*max-df*), el rango de n-gramas que se utilizaron y las etapas de pre-procesamiento utilizadas. También encontramos cual fue el mayor valor de la métrica de Silhouette y para qué número de clusters. A modo de conclusión de esta serie de experimentos, podemos ver que en la mayoría de las soluciones la métrica de Silhouette no marca una gran diferencia entre la variedad de resultados. Si bien la configuración ***"(min_df=4, max_df=1.0, K=4, n_grams=(2,3), preprocessors=[])"*** (resaltada en el cuadro 1) es la que puntúa mejor con un valor de 0.344, al explorar manualmente los datos vimos que esta solución, como muchas otras de la tabla, tenía el problema de *catch-all cluster*. En estos casos ocurre que, dependiendo de la configuración, la clase mayoritaria contenía entre un *78.64 %* y *92.49 %* de la población total.

Sin embargo, luego de examinar manualmente los resultados, la forma en la que estaban conformados los cluster y los n-gramas más frecuentes, además de hacer una breve investigación sobre el dominio, pudimos notar una configuración que resultaba ser intuitivamente más útil que las demás para describir los datos. La configuración es: ***"(min_df=4, max_df=1.0, K=8, n_grams=(1,3), preprocessors=[stem, tf*idf])"*** (resaltada con un "*" en la tabla) y se detalla a continuación:

- Features: 25466
- N-grams: (1,3)
- Min df: 4
- Max df: 1.0
- N-Clusters: 8
- Preprocessors: stem + tf*idf

En los 8 clusters de esta solución casi el *76 %* de la población se distribuye equitativamente en dos clases y el *24 %* se distribuye en los otros 6 clusters que restan sin que haya una clase predominante mayor entre ellos.

4.6. Clasificador y validación

Ahora que tenemos los clusters formados queremos contemplar el caso en el que agreguemos un nuevo tweet a nuestro dataset. Para esto podemos entrenar un clasificador utilizando los clusters como etiquetas de clases. Una vez entrenado el clasificador, podremos predecir efectivamente a qué cluster pertenece según la etiqueta de la clase en que fue clasificado. Cada vez que queramos clasificar un tweet que no pertenece

al dataset, tendrá que ser preprocesado anteriormente por el *pipeline* descrito en la sección 2.2. Luego utilizamos una máquina de vectores de soporte (*Support Vector Machines, SVMs*) para predecir a qué clase pertenece. Utilizando una penalidad C de 0.1 y utilizando la función de pérdida "squared hinge", obtuvimos el mayor rendimiento, de *98.08 %*.

Después de ver este resultado y analizando el funcionamiento del clasificador, notamos que las *SVMs* eran propensa a clasificar las instancias como pertenecientes a la clase mayoritaria. Al notar esto, decidimos comparar los resultados con un árbol de decisión como clasificador. De la misma forma que lo hicimos para nuestra *SVM*, utilizamos Grid Search para hacer una búsqueda exhaustiva en cuanto a qué combinación de parametros obtenía un mayor puntaje. Utilizando el criterio de impureza de *Gini* y con un maximo de 9 niveles de profundidad, obtuvimos un *95 %* de efectividad.

4.7. Análisis exploratorio de los datos mediante representaciones gráficas interactivas

Considerando los grandes volúmenes de los datos que manejamos, analizar los resultados manualmente se vuelve una tarea casi imposible. Es en este contexto que el uso de visualizaciones nos permite identificar aquella información relevante de forma condensada. Las visualizaciones nos permiten resumir todas aquellas características de los datos en las cuales estamos interesados en representaciones gráficas, haciendo así que sean fáciles de interpretar para hacer mejores conclusiones y luego poder tomar decisiones basada en la información procesada.

Tomando las principales características de la visualización del NYT, utilizamos una nube de palabras y un listado de tweets para la visualización de cada cluster. Para ajustarlo a nuestras necesidades, agregamos dos características que nos permitían comunicar visualmente el sentimiento asociado a los tweets y, consecuentemente a los términos asociados en la nube de palabras. En el apéndice detallamos el proceso realizado y las características de las visualizaciones que utilizamos para este trabajo.

5. Conclusiones

En este trabajo, hemos estudiado un problema complejo como es el de la minería de opiniones en Twitter y los desafíos asociados al análisis exploratorio de los resultados de dicho proceso. Si bien Twitter tiene ciertas limitaciones a la hora de proveer información, es más que suficiente para llevar a cabo un análisis exploratorio con la finalidad de comprender mejor un mercado o un acontecimiento. Para esto, nos hemos centrado en diseñar un pipeline en el cual adquirimos un conjunto de tweets sobre un determinado tópico, los pre-procesamos y realizamos hacemos clustering con ellos para finalmente verlos representados de forma visual.

En cada etapa de este pipeline identificamos conjuntos de problemas y diferentes soluciones que se pueden llevar adelante para cada uno de ellos. Si bien este trabajo propone una implementación concreta para este flujo, existe la posibilidad de continuar investigando en mayor profundidad de manera independiente cada etapa.

6. Trabajo futuro

A partir del estado actual de la implementación resultante de este trabajo se desprenden diferentes líneas de trabajo sobre las cuales se podrían continuar.

Entre ellas, la que sin duda podría tener mayor contribución al resultado final es la de realizar un preprocesamiento de los mensajes más agresivo para que las características resultantes permitan una forma de implementar los métodos de clustering con mayor efectividad.

Asimismo, si bien abordamos la minería de opiniones como un problema de clasificación de textos, se podría implementar un sistema más complejo de múltiples etapas que contemplase diferentes particularidades del lenguaje natural como la ironía o las negaciones.

Finalmente, en cuanto a la visualización de los resultados del pipeline, encontrar un esquema gráfico que permita procesar visualmente de forma rápida y efectiva es una tarea desafiante por el gran volumen y la complejidad de los datos que manejamos. Uno de los problemas a destacar que resulta atractivo para continuar es la de la representación de un espacio de clusters.

Referencias

[1] R. González-Ibáñez, S. Muresan, and N. Wacholder, "Identifying sarcasm in twitter: A closer look," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2*, ser. HLT '11. Stroudsburg, PA, USA: Association for Computational Linguistics, 2011, pp. 581–586. [Online]. Available: http://dl.acm.org/citation.cfm?id=2002736.2002850

[2] C. Yang, K. H.-Y. Lin, and H.-H. Chen, "Building emotion lexicon from weblog corpora," in *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ser. ACL '07. Stroudsburg, PA, USA: Association for Computational Linguistics, 2007, pp. 133–136. [Online]. Available: http://dl.acm.org/citation.cfm?id=1557769.1557809

[3] M. Gamon, A. Aue, S. Corston-Oliver, and E. Ringger, "Pulse: Mining customer opinions from free text," in *Proceedings of the 6th International Conference on Advances in Intelligent Data Analysis*, ser. IDA'05. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 121–132. [Online]. Available: http://dx.doi.org/10.1007/11552253_12

[4] J. Blitzer, M. Dredze, and F. Pereira, "Biographies, bollywood, boomboxes and blenders: Domain adaptation for sentiment classification," in *In ACL*, 2007, pp. 187–205.

[5] L. Zhang and B. Liu, "Identifying noun product features that imply opinions." in *ACL (Short Papers)*. The Association for Computer Linguistics, 2011, pp. 575–580. [Online]. Available: http://dblp.uni-trier.de/db/conf/acl/acl2011s.html#ZhangL11

[6] i. Twitter, "Api rate limits." [Online]. Available: https://dev.twitter.com/rest/public/rate-limiting

[7] B. Liu, *Sentiment Analysis - Mining Opinions, Sentiments, and Emotions*. Cambridge University Press, 2015. [Online]. Available: http://www.cambridge.org/us/academic/subjects/computer-science/knowledge-management-databases-and-data-mining/sentiment-analysis-mining-opinions-sentiments-and-emotions

[8] M. Hu and B. Liu, "Mining and summarizing customer reviews," in *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '04. New York, NY, USA: ACM, 2004, pp. 168–177. [Online]. Available: http://doi.acm.org/10.1145/1014052.1014073

[9] S.-M. Kim and E. Hovy, "Determining the sentiment of opinions," in *Proceedings of the 20th International Conference on Computational Linguistics*, ser. COLING '04. Stroudsburg, PA, USA: Association for Computational Linguistics, 2004. [Online]. Available: http://dx.doi.org/10.3115/1220355.1220555

[10] A. Kennedy and D. Inkpen, "Sentiment classification of movie reviews using contextual valence shifters," *Computational Intelligence*, vol. 22, no. 2, pp. 110–125, 2006.

[11] P. Refaeilzadeh, L. Tang, and H. Liu, "Cross-validation," in *Encyclopedia of Database Systems*, 2009, pp. 532–538. [Online]. Available: http://dx.doi.org/10.1007/978-0-387-39940-9_565

[12] B. Pang and L. Lee, "A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts," in *Proceedings of the 42Nd Annual Meeting on Association for Computational Linguistics*, ser. ACL '04. Stroudsburg, PA, USA: Association for Computational Linguistics, 2004. [Online]. Available: http://dx.doi.org/10.3115/1218955.1218990

[13] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[14] D. J. C. MacKay, *Information Theory, Inference & Learning Algorithms*. New York, NY, USA: Cambridge University Press, 2002.

[15] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1988.

[16] C. Fraley and A. E. Raftery, "How many clusters? which clustering method? answers via model-based cluster analysis," *The Computer Journal*, vol. 41, no. 8, pp. 578–588, 1998. [Online]. Available: http://www3.oup.co.uk/computer_journal/hdb/Volume_41/Issue_08/Fraley.pdf

[17] B. Mirkin, "Choosing the number of clusters," *WIREs Data Mining Knowl Discov*, vol. 1, no. 3, pp. 252–260, May 2011. [Online]. Available: http://dx.doi.org/10.1002/widm.15

[18] D. Arthur and S. Vassilvitskii, "K-means++: The advantages of careful seeding," in *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA '07. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035. [Online]. Available: http://dl.acm.org/citation.cfm?id=1283383.1283494

[19] M. E. Celebi, H. A. Kingravi, and P. A. Vela, "A comparative study of efficient initialization methods for the k-means clustering algorithm." *Expert Syst. Appl.*, vol. 40, no. 1, pp. 200–210, 2013. [Online]. Available: http://dblp.uni-trier.de/db/journals/eswa/eswa40.html#CelebiKV13

Apéndice

A. Visualización de los resultados

Una vez finalizadas las etapas de análisis de sentimientos y de clustering, contamos con dos tipos de resultados diferentes que sería de utilidad poder ver conjuntamente. Por un lado tenemos a cada tweet etiquetado como "positivo" o "negativo" según el sentimiento expresa el mensaje; por el otro, una división del corpus en diferentes subgrupos que consideramos ser los diferentes aspectos de nuestro tema a analizar.

Ya que queremos hacer un análisis exploratorio de los datos, carece de sentido hacer una evaluación rigurosa de los resultados de etapas anteriores. Es por esto que toleramos un margen de error en cuanto a la clasificación de sentimientos y las clasificaciones de aspectos. Estamos interesados en tener una idea general de cuáles son las diferentes tendencias con respecto a determinado contenido.

También considerando los grandes volúmenes de los datos que manejamos, poder analizar los resultados manualmente se vuelve una tarea casi imposible. Es en este contexto que el uso de visualizaciones nos permite identificar aquella información relevante de forma condensada. Las visualizaciones nos permiten resumir todas aquellas características de los datos en las cuales estamos interesados en representaciones gráficas, haciendo así que sean fáciles de interpretar para hacer mejores conclusiones y luego poder tomar decisiones basada en la información procesada.

En el marco de este trabajo buscamos diferentes tipos de visualizaciones que nos permiten detectar qué sentimientos hay con respecto a los diferentes aspectos de un tema. Las características y la justificación de cada una de ellas se detalla a continuación:

- Cada cluster está compuesto por tweets, por lo que nos gustaría poder tener un pantallazo viendo cuales son las términos que los componen y que tan frecuente son entre los tweets del mismo cluster.
- Muchas veces una palabra no tiene una polaridad positiva o negativa directamente asociada, como por ejemplo "#oscars" o "academy". Entonces, buscamos poder ver visualmente una aproximación que nos indique si en este corpus esta palabra aparece principalmente en contextos negativos o positivos.
- Si bien el punto anterior nos permite rápidamente identificar el sentimiento asociado a un término, da lugar a un margen grande de error y deja de lado la información del contexto en el que ocurre el tweet, por lo que también queremos ver un listado de tweets en los cuales este término ocurre.

Entre las diferentes visualizaciones que analizamos decidimos realizar la nuestra basada en el trabajo que realizó el periódico The New York Times para analizar cuáles eran las palabras más frecuentes que se utilizaban en discursos políticos. En la figura 10 se muestra una imagen de esta visualización[3].

Esta visualización está compuesta por una combinación de una visualización de nube de palabras, representadas con burbujas, y también un listado de recortes de discursos.

En la nube de palabras, cada burbuja contiene un término frecuente y el número de veces que esta se puede encontrar en diferentes discursos, el tamaño de cada burbuja es relativo a la cantidad de menciones del término que representa.

3. http://www.nytimes.com/interactive/2012/09/04/us/politics/democratic-convention-words.html

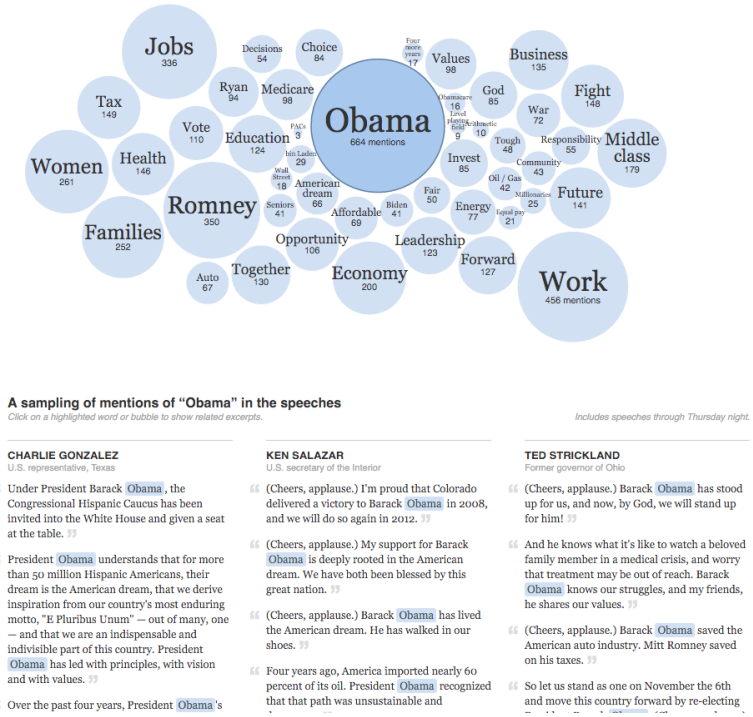


Figura 10. Visualización de The New York Times (NYT) analizando palabras frecuentes en discursos politicos

Asimismo, el listado de recortes está vinculado con la nube de palabras de tal forma que cuando seleccionamos una burbuja solo veremos aquellos fragmentos de discurso en los cuales aparece resaltado el término asociado a la burbuja.

A.1. Nuestra visualización. Tomando las principales características de la visualización del NYT, utilizamos una nube de palabras y un listado de tweets para la visualización de cada cluster. Para ajustarlo a nuestras necesidades agregamos dos características que nos permitían comunicar visualmente el sentimiento asociado a los tweets y consecuentemente a los términos asociados en la nube de palabras.

También como ampliación a esta visualización buscamos una forma de poder visualizar la solución globalmente integrando todas las nubes de palabras con un tamaño proporcional a la población y representando la semejanza relativa entre clusters. Sin embargo, debido a la complejidad de esta tarea y a las limitaciones de la librería que utilizamos para graficar tuvimos que dejarlo a fuera del alcance de este proyecto. Por lo que agregamos un selector que nos permite poder seleccionar entre los diferentes clusters. La figura 11 muestra el resultado de esta etapa.

A.2. La nube de palabras. Como muestra la figura 12, para la implementación de la nube de palabras, utilizamos la librería D3[4] que nos permitió llevar los datos recolectados

4. https://d3js.org/

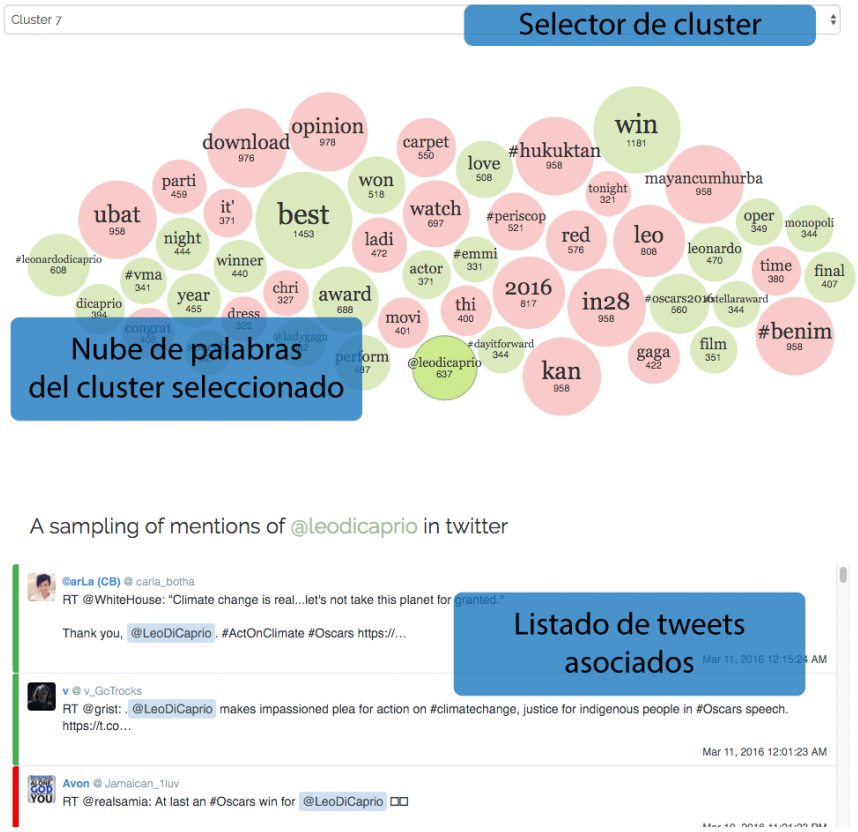


Figura 11. Visualización de la solución diseñada para ver gráficamente el resultado de nuestro pipeline

Figura 12. Visualización de nube de palabras utilizando burbujas

en las etapas anteriores a una implementación visual para representar los clusters de forma resumida y destacar aquellos aspectos que consideramos más importantes para explorar los datos.

En cuanto a la descripción de las burbujas es similar a la del NYT pero decidimos colorear cada burbuja según el sentimiento asociado a los tweets en los cuales aparece el término asociado. Si aparece más veces en tweets que fueron clasificados como positivos y no tantas veces en tweets negativos, lo identificaremos de manera visual con un color verde. En caso contrario, si ocurrió más veces en tweets negativos que positivos, lo coloreamos con un color rojo.

Así mismo, otra característica de este gráfico es que se puede interactuar directamente con las burbujas arrastrándolas y acomodándolas en el espacio. Esto es útil para el caso de uso de que un analista explore el mercado y desee realizar una presentación de sus conclusiones.

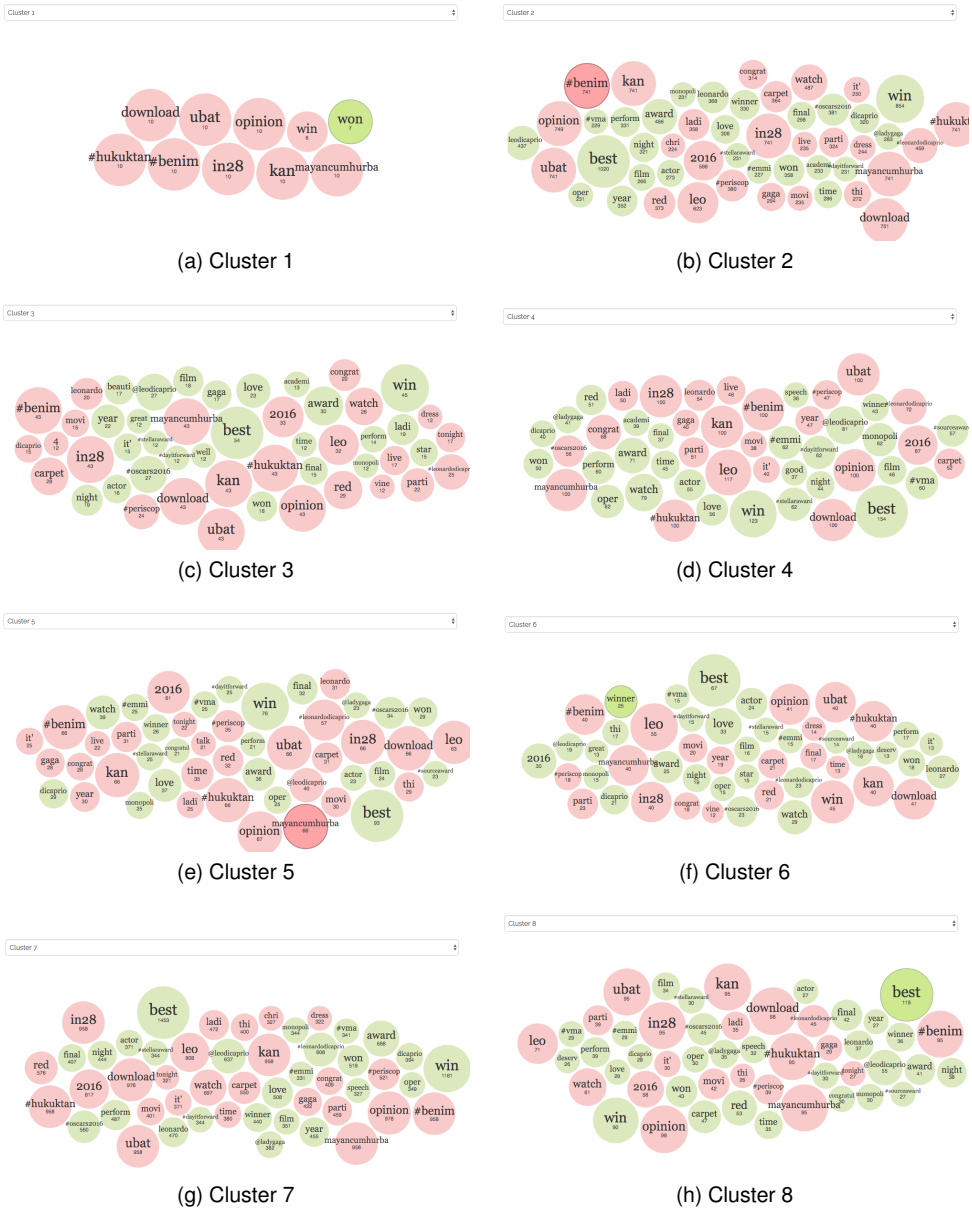
Figura 13. Visualización del listado de tweets etiquetados según su sentimiento

A.3. Listado de tweets. Cuando seleccionamos uno de los nodos del gráfico que se encuentra en la parte superior, automáticamente se despliega un listado filtrado de tal forma que solo aparecen tweets que contienen resaltado el término asociado al nodo seleccionado como muestra la figura 13.

A la hora de implementar este listado de tweets, decidimos hacer una implementación similar al news feed de Twitter. Tenemos un listado donde los tweets se muestran uno arriba del otro y de cada uno de ellos podemos ver el mensaje, la información del usuario y la fecha de publicación. Sumado a esto cada tweet tiene un señalador que nos dice si el sentimiento asociado al mismo es positivo o negativo con un color verde o rojo respectivamente.

Luego de haber realizado la etapa de clasificación de sentimientos y de clustering, este proceso culmina con la posibilidad de ver visualmente sobre qué temas los usuarios de Twitter comentaron durante la gala de los Óscars. Esta visualización nos facilita explorar la opinión de los usuarios sobre cada tema haciendo uso de las palabras más frecuentes y cual es el sentimiento asociado (positivo o negativo) a cada una de ellas según la clasificación de sentimiento de los tweets, propios de cada tema, donde ocurren.

Luego de ver graficado cada cluster en la figura 14 podemos realizar las siguientes serie de observaciones. En primer lugar, a excepción del Cluster 1 donde predominan



(a) Cluster 1 (b) Cluster 2

(c) Cluster 3 (d) Cluster 4

(e) Cluster 5 (f) Cluster 6

(g) Cluster 7 (h) Cluster 8

Figura 14. Visualización de sentimientos etiquetados en los diferentes clusters

los términos negativos, en ningún otro cluster es notable la diferencia entre la cantidad de términos positivos y negativos. Podríamos decir que está casi balanceada.

También podemos observar que algunas palabras, tales como *"leonardo"* (cluster 4

y cluster 6) y *"gaga"* (cluster 3 y cluster 4), ocurren en diferentes clusters pero con diferente frecuencia e incluso con un sentimiento asociado diferente. Esto último se debe en parte a que los tweets propios de cada cluster son diferentes, por lo tanto la frecuencia con la que aparecen ciertos términos puede ser utilizada para caracterizar las particularidades de cada tema.

Asimismo, podemos notar que términos como *"@leodicaprio"* y *"@ladygaga"* ocurren siempre con un sentimiento positivo mientras que *"leo"* y *"opinion"* lo hacen con un sentimiento negativo. En esto podemos ver la particularidad de que cuando se hace referencia a ciertas entidades, como al actor *Leonardo DiCaprio*, de forma positiva se utilizan ciertos términos en el mensaje (*"@leodicaprio"*) distintos a cuando se lo hace de forma negativa (*"leo"*).

Por otro lado, nos encontramos con una de las particularidades de Twitter en cuanto qué tan frecuente es compartir las publicaciones de otros usuarios (*retweets*). En el cluster 1 podemos notar que la nube de palabras por sí sola nos indica que está compuesto por pocos tweets repetidos múltiples veces. En clusters caracterizados por más términos solo se podría detectar que esto ocurre examinando exhaustivamente el listado de tweets asociado a cada término, dándonos así una primera impresión errónea en cuanto a las características de cada tema.

Por último, hay ciertos términos como *"#benim"*, *"download"* y *"in28"* que no aportaron ningún valor para describir los temas e incluso dificultaron realizar un análisis semántico de tal forma que lo podríamos considerar como *spam*. Podríamos abordar este problema con un preproceso más agresivo, eliminando términos que ocurran de forma indistinta en tweets muy diferentes entre sí, por ejemplo, usando test de hipótesis o información mutua como métricas más sofisticadas con el mismo espíritu que el tf*idf, o bien enfocando nuestro análisis utilizando aquellos términos que son más característicos del tema.

B. Tabla de resultados de la etapa de clustering

Features	Min-df	Max-df	N-gramas	Pre-procesamiento	Mejor Silhouette	Mejor K
3448	10	0.01	(1-1)	stem-tf*idf-bin	0.037	6
3448	10	0.01	(1-1)	stem-tf*idf	0.036	7
6353	10	0.01	(1-2)	stem-tf*idf	0.023	7
2459	15	0.01	(1-1)	stem-tf*idf-bin	0.040	6
2459	15	0.01	(1-1)	stem-tf*idf	0.051	5
2413	15	0.01	(2-3)	stem-tf*idf-bin	0.135	8
2317	15	0.01	(2-3)		0.179	4
18878	2	1.0	(1-1)	stem-tf*idf-bin	0.067	8
27183	3	0.01	(2-3)	stem-tf*idf-bin	0.046	2
27183	3	0.01	(2-3)	stem-tf*idf	0.044	5
9991	3	1.0	(1-1)	stem-tf*idf-bin	0.085	7
9991	3	1.0	(1-1)	stem-tf*idf	0.079	7
26380	3	1.0	(1-2)	stem-tf*idf-bin	0.078	4
26380	3	1.0	(1-2)	stem-tf*idf	0.076	7
37222	3	1.0	(1-3)	stem-tf*idf-bin	0.074	5
37222	3	1.0	(1-3)	stem-tf*idf	0.069	5
7735	4	1.0	(1-1)	stem-tf*idf-bin	0.074	7
18617	4	1.0	(1-2)	stem-tf*idf-bin	0.073	4
18617	4	1.0	(1-2)	stem-tf*idf	0.066	8
25466	4	1.0	(1-3)	stem-tf*idf-bin	0.068	4
***25466**	**4**	**1.0**	**(1-3)**	**stem-tf*idf**	**0.072**	**4**
25988	4	1.0	(1-3)		0.208	4
17257	**4**	**1.0**	**(2-3)**		**0.344**	**4**
6155	5	0.01	(1-1)	stem-tf*idf-bin	0.044	5
6262	5	1.0	(1-1)	stem-tf*idf-bin	0.077	7
6262	5	1.0	(1-1)	stem-tf*idf	0.071	7
13772	5	1.0	(1-2)	stem-tf*idf	0.079	5
14353	5	1.0	(1-2)		0.093	4
18042	5	1.0	(1-3)	stem-tf*idf-bin	0.078	6
18042	5	1.0	(1-3)	stem-tf*idf	0.075	8
11780	5	1.0	(2-3)	stem-tf*idf-bin	0.094	5
11780	5	1.0	(2-3)	stem-tf*idf	0.092	8

Cuadro 1. TABLA DE SOLUCIONES CON DIFERENTES CONFIGURACIONES. TODAS FUERON REALIZADAS CON VALORES DE *k* ENTRE 4 Y 9 CLUSTERS PARA K-MEANS. CADA CONFIGURACIÓN CUENTA CON UN NÚMERO DE FEATURES, UN MÍNIMO DE FRECUENCIA (*min-df*), UN MÁXIMO DE FRECUENCIA(*max-df*), EL RANGO DE N-GRAMAS QUE SE UTILIZARON Y CUALES FUERON LAS ETAPAS DE PRE-PROCESAMIENTO UTILIZADAS. TAMBIÉN ENCONTRAMOS CUAL FUE EL MAYOR VALOR DE LA MÉTRICA DE SILHOUETTE Y PARA QUE NÚMERO DE CLUSTERS.