Reconocimiento de Patrones y Aprendizaje Automatizado

Análisis de la actividad en Twitter sobre las elecciones nacionales México-2018 usando agrupamiento por clustering y minería de opinión

M. CONCHA VÁZQUEZ
L. HERNÁNDEZ CANO
M. X. LEZAMA HERNÁNDEZ
E. E. VÁZQUEZ SALCEDO

Facultad de Ciencias, UNAM

Facultad de Ciencias
Universidad Nacional Autónoma de México
http://www.fciencias.unam.mx/

Título del proyecto:
Análisis de la actividad en Twitter sobre las elecciones nacionales México-2018 usando agrupamiento por clustering y mineria de opinión.

Tema del proyecto:
Aprendizaje no supervisado; Clustering, Minería de Opinión, Análisis de Sentimientos.

Periodo:
Mayo-junio, 2018.

Número de Grupo:
7085

Participantes:
Concha Vázquez Miguel.
Hernández Cano Leonardo.
Lezama Hernández María Ximena.
Vázquez Salcedo Eduardo Eder.

Supervisores:
Profesor Gustavo De la Cruz Martínez.
Rafael Robles Ríos.
Estefania Prieto Larios.

Fecha en que se completó el proyecto:
7 de junio del 2018.

Resumen:

Aplicamos técnicas de aprendizaje automatizado no supervisado para el análisis de *tuits* sobre las elecciones nacionales México-2018. En particular empleamos *clustering* K-MEANS en una implementación que directamente ocupa funciones provistas por el lenguaje de programación de propósito estadístico R, haciendo de igual forma un análisis con minería de texto para identificar los términos más comunes y llevar a cabo un análisis de sentimientos.

Índice general

Capítulo 1

Planteamiento

1.1. Introducción

En años recientes se ha visto un incremento en el interés circundante a la potencial aplicación de la minería de opinión o análisis de sentimientos sobre temas electorales alrededor de todo el globo (Andrea Ceron, s.f.) (Carvalho, Nagano, y Barros, s.f.) (Joyce y Deng, s.f.). La línea de investigación y publicaciones en la literatura sobre el tema han ido en auge, sugiriendo que la posibilidad de analizar datos de redes sociales puede fungir de buena forma como contrapartida a los métodos estadísticos de inferencia tradicionales (Ramteke, Shah, Godhia, y Shaikh, s.f.) (Hamling y Agrawal, s.f.).

Por mucho tiempo, México ha sido objeto de múltiples controversias y el foco de muchas polémicas de índole electoral, poniendo inclusive en tela de juicio a los mismos organismos encargados de velar por la integridad del proceso de votación. Las casas encuestadores no están tampoco exentas de varias críticas, propiciando que los mexicanos hayan perdido su credibilidad hacia éstas. Aunado a lo anterior, el crecimiento y proliferación en el número de usuarios en redes sociales y la actividad política que puede presenciarse en *Twitter* podría servir para analizar la actividad de los posibles votantes en torno a temas electorales y poder generar reportes estadísticos basados en la propia actividad de los mexicanos.

Además de poder ayudar a presentar un análisis más imparcial sobre las preferencias electorales de los mexicanos, este tipo de análisis podría también contribuir a que los propios coordinadores de campaña y encargados de imagen de los candidatos pudieran atender las principales inquietudes vistas a través de los reportes. Finalmente, el hecho de llevar a cabo análisis con datos generados directamente por individuos en su día a día puede favorecer la apreciación de la actividad política como una esfera fundamental de la vida actual y la identificación propia como entes políticos.

1.2. Definición del problema

1.2.1. Análisis de sentimientos

Dificultades y comparación con la clasificación por tema

La clasificación de texto mediante análisis de sentimientos generalmente buscará hacer trabajar con unas pocas categorías. Además de los desafíos de la clasificación por tema, se tienen características como la intensidad del sentimiento y el grado de "positividad" que no aparecen en otros tipos de clasificaciones de texto.

Se presentan varios desafíos para lograr la clasificación por análisis de sentimientos. Imaginemos que deseamos clasificar el siguiente fragmento de texto como positivo o negativo (entiendo dichas etiquetas de acuerdo a una interpretación intuitiva):

"Aprender a usar Netbeans me hace sentir alegría, es la mejor forma de pasar el tiempo"

La presencia de las palabras "alegría" y "mejor" hacen pensar que la oración es positiva y podría haber un conjunto de palabras que indicaran esto. En efecto, es un acercamiento que se ha presentado y en particular la técnica que emplearemos se basa en esta noción. Más adelante adentraremos en esto, sin embargo hay que notar por el momento que la construcción de tal conjunto –el *léxico*, como es llamado en este contexto– presenta un desafío también.

Continuemos con el ejemplo, imaginemos que el texto es precedido por una oración de la siguiente forma:

"¡Uy, sí! Aprender a usar Netbeans me hace sentir alegría, es la mejor forma de pasar el tiempo"

La primera frase, para hablantes del castellano en México, es un claro indicador de sarcasmo, lo cuál cambia por completo el significado del texto, negando la interpretación que se le había dado. Esto ejemplifica que la clasificación por análisis de sentimientos es altamente dependiente del contexto, lo cuál no sucede en clasificación por tema –o al menos, no en la misma medida– y es parte de los desafíos que presenta la clasificación por análisis de sentimientos.

Otros desafíos incluyen, en el castellano la *doble negación*, que la forma de expresar los sentimientos varía por tema y contexto (aunque en general se considera que permanece consistente en varios dominios) y en particular, en el caso de los medio de comunicación en internet como Twitter, que los sentimientos se expresan de formas muy particulares al medio en cuestión. Todo ello sin contar la mera definición de etiquetas como *positivo* o *negativo* o *sentimiento*, que no resulta apropiado discutir en este trabajo.

Conceptos básicos

Presentamos algunos de los conceptos y terminología del campo de clasificación de texto por análisis de sentimientos. Cuando se busca una clasificación binaria que agrupe en *positivo* y *negativo* estamos hablando de una *clasificación de polaridad de sentimientos* o simplemente *clasificación de polaridad*. Cuando discutimos una clasificación en más categorías hablamos de *clasificación de sentimientos*.

En Bo Pang (2008) se identifican tres factores útiles en el problema de clasificación de sentimientos:

- En la clasificación de polaridad de sentimientos en donde los autores del texto mencionan explícitamente los sentimientos (por ejemplo, reseñas de una película), los fragmentos de texto en dónde se dan hechos objetivos son útiles.

- Determinar si un texto objetivo (es decir, un texto que hable de hechos y no sea de opinión) corresponde a un texto *bueno* o *malo* sigue sin ser el mismo problema que clasificar por tema, y hereda los desafíos del problema de la clasificación por análisis de sentimientos.
- La diferencia entre información objetiva y subjetiva puede ser sutil (consideremos por ejemplo, "la batería dura dos horas" y "la batería *sólo* dura dos horas")

El problema de la *inferencia de evaluación* consiste en determinar la evaluación del autor de acuerdo a una escala multi-punto, por ejemplo de cero a cinco estrellas. Este problema puede ser abordado como un problema de regresión.

El problema de la *identificación de subjetividad* y la *identificación de opinión* consiste en clasificar la información como una opinión o como un hecho. Se reportan porcentajes de acierto tan altos como 97 %, como en el estudio de Yu y Hatzivassiloglou (s.f.) en el que se usa un clasificador ingenuo de Bayes para clasificar artículos de las secciones de noticias y las de opinión en el *Wall Street Journal*.

Otro problema consiste en analizar textos que no comparten tema pero son relevantes en una discusión política, en particular se analizan la perspectiva y los puntos de vista de los autores para clasificarlos de acuerdo a su orientación política, ideológica, teológica, etcétera.

Características

La tarea de obtener una representación útil a partir de un texto es de suma importancia (Bo Pang, 2008). A continuación discutimos algunas de las características relevantes que se pueden considerar para conseguir una representación útil.

Es común representar un texto como un vector cuyas entradas corresponden a distintas palabras. Se tienen dos acercamientos clásicos, en el primero los valores de las entradas corresponden al número de veces que la palabra aparece en el texto, y en el otro las entradas son binarias y únicamente representan si la palabra aparece o no en el texto.

La posición de un símbolo en el texto (al inicio, por la mitad o al final) puede influir en la medida en la que la palabra o frase afecta el sentimiento general o la subjetividad del texto. La consideración de este efecto es reportada como controversial por Bo Pang (2008).

La presencia de ciertas partes del habla (en particular se habla de ciertos adjetivos) puede influir en mayor medida a la clasificación que otras, sin embargo las otras partes del habla también aportan información.

La representación de la negación es compleja, y se puede tratar en varias etapas con representaciones distintas, algunas que incluyan a la negación y otras no. La presencia de la negación en alguna parte del texto puede afectar la clasificación (como se ejemplifico arriba) y la presencia puede ser muy sutil como en el sarcasmo o la ironía.

Existe interacción entre tema y sentimiento y se puede considerar una representación que considere el efecto que pueden tener tales interacciones.

Clasificación no supervisada y construcción de léxico

Una parte significativa de los acercamientos por clasificación no supervisada se encargan de construir primero un *léxico* –un conjunto de palabras o características– sin supervisión y determinar después el grado en el que las palabras son positivas, negativas, o pertenecientes a algún otro sentimiento (las categorías dependen de las clases en las que se quiera clasificar). Una vez que se construye el léxico se le aplica al texto a clasificar una función basada en los indicadores del léxico, por ejemplo, haciendo una suma de los valores de los indicadores para cada clase.

Este acercamiento asume que el sentimiento está relacionado con la presencia de ciertas palabras o frases en el texto (Łukasz Augustyniak, Szymański, Kajdanowicz, y Tuligłowicz, 2016). Un léxico es un conjunto de palabras o características que tienen asignado un valor de sentimiento

Existen varias propuestas para obtener un léxico. Un acercamiento comienza con la selección de un conjunto de palabras realizado por un experto, quién luego lo expande utilizando herramientas avanzadas de lenguaje (Łukasz Augustyniak y cols., 2016). Otro, por ejemplo, se encarga de partir de un conjunto pequeño de *semillas* y propagar el grado en el que cada una es positiva o negativa a otras palabras siguiendo ciertas restricciones inferidas. Otro acercamiento parte de una representación semántica abstracta para generar estructuras verbales a partir de un conjunto de datos.

La construcción del léxico trae sus propios desafíos. Por ejemplo, Pang y Lee (2002) reportan que los léxicos construidos por un par de estudiantes de posgrado de Ciencias de la Computación tuvieron un 54 % y un 64 % de precisión. Uno puede imaginar que un especialista en letras podría generar un mejor conjunto, sin embargo, el experimento invita a pensar que los seres humanos no somos naturalmente hábiles en esa tarea.

Dado que este es el acercamiento que exploramos, detallamos algunas ventajas y algunas desventajas.

Ventajas:

- Puede tener buen desempeño
- Son eficientes en cuanto a tiempo de ejecución
- No requieren un conjunto etiquetado

Desventajas:

- En casos de textos de opinión sofisticados tienen mal desempeño
- No escalan bien para textos especializados, pues el léxico es específico para un dominio

Clasificación supervisada

Otro acercamiento propone aprender de datos ya etiquetados para hacer predicciones sobre los nuevos casos. Para emplear esta técnica uno debe de definir un método para extraer características del texto y aplicarla a los objetos.

El conjunto de datos etiquetado puede ser producido manualmente por humanos o puede obtenerse infiriendo etiquetas para un conjunto de datos no etiquetado, por ejemplo utilizando las “estrellas” que el autor dió a una reseña.

1.2.2. Elementos de análisis

Definiremos los elementos básicos del análisis. En particular tratamos con *tuits*, que son publicaciones cortas hechas en la plataforma social *Twitter*.

Definición 1 (Tuit) *Un tuit es un vector* $[a_1, ..., a_n] \in R^n$ *de palabras, junto con información del autor, fecha de publicación y otros metadatos. Decimos que las entradas del vector son los atributos de texto del tuit.*

En particular esta representación está de acuerdo con la descrita en la sección de Características.

Definición 2 (Clasificación de un conjunto de tuits) *Dado un conjunto* S *de tuits, una clasificación de* S *es una partición indexada* $C_1, ..., C_m$ *de* S*. Decimos que la clasificación de* $s \in S$ *es i si* $s \in C_i$*.*

Nos interesa encontrar un proceso que para cualquier conjunto de tuits nos devuelva una clasificación que agrupe los tuits en conjuntos en los que predomine un sentimiento.

Debido al gran número de ejemplares necesarios para obtener una clasificación es deseable diseñar un proceso que realice dicha clasificación, pues no es razonable esperar que un ser humano termine la tarea en un tiempo apropiado.

Definición 3 (Clasificación no supervisada) *Cuando se infiere una clasificación a partir de los atributos de los ejemplares sin una clasificación a priori se realiza una clasificación no supervisada.*

Además, se empleará un léxico para realizar el análisis de sentimientos de un conjunto de tuits. Más adelante detallaremos las técnicas y las herramientas que se utilizaron sobre el análisis para representar los resultados.

Capítulo 2

Metodología

2.1. Clustering

Clustering es el problema de encontrar una estructura en una colección de datos sin etiqueta. Una definición amplia de clustering podría ser ël proceso de organizar objetos en grupos cuyos miembros son similares de alguna manera". Un clúster es, por lo tanto, una colección de objetos que son *"similares"* entre ellos y son *"diferentes"* a los objetos que pertenecen a otros grupos.

2.1.1. Descripción del clustering

Clustering es una técnica de minería de datos (*data mining*) dentro de la disciplina de Inteligencia Artificial que identifica de forma automática agrupaciones o *clústeres* de elementos de acuerdo a una medida de similitud entre ellos. El objetivo fundamental de las técnicas de clustering consiste en identificar grupos o clústeres de elementos tal que:

- La similitud media entre elementos del mismo clúster sea alta. **Similitud intra-clúster alta**.
- La similitud media entre elementos de distintos clústeres sea baja. **Similitud inter-clúster baja**.

Agrupamiento mediante *k-means*

K-means clustering es un tipo de aprendizaje no supervisado, que se utiliza cuando se tiene datos no etiquetados (es decir, datos sin categorías o grupos definidos). El objetivo de este algoritmo es encontrar grupos en los datos, con el número de grupos representados por la variable **K**. El algoritmo funciona iterativamente para asignar cada punto de datos a uno de *los grupos K* en función de las características que se proporcionan. Los puntos de datos se agrupan según la similitud de características. Los resultados del algoritmo de agrupamiento K-means son:

- Los centroides de los clústeres K, que se pueden usar para etiquetar nuevos datos.
- Etiquetas para los datos de entrenamiento (cada punto de datos se asigna a un solo grupo)

En lugar de definir grupos antes de mirar los datos, la agrupación le permite buscar y analizar los grupos que se han formado orgánicamente.

Cada centroide de un clúster es una colección de valores de características que definen los grupos resultantes. Examinando los pesos de las características del centroide se puede interpretar cualitativamente el tipo de grupo que representa cada clúster.

El algoritmo de agrupamiento de K-means usa refinamiento iterativo para producir un resultado final. Las entradas de algoritmo son el número de clústeres K y el conjunto de datos. El conjunto de datos es una colección de características para cada punto de datos. Los algoritmos comienzan con estimaciones iniciales para los K centroides, que pueden generarse aleatoriamente o seleccionarse aleatoriamente del conjunto de datos. El algoritmo luego itera entre dos pasos:

1. Paso de asignación de datos:

   Cada centroide define uno de los clusters. En este paso, cada punto de datos se asigna a su centroide más cercano, en función de la distancia euclidiana al cuadrado. Más formalmente, si c_i es la colección de centroides en el conjunto C, entonces cada punto de datos x se asigna a un clúster basado en:

   $$\text{argmin } dist(c_i, x)^2 \text{ tal que } c_i \in C$$

   donde $dist()$ es la distancia euclidiana estándar y permite que el conjunto de asignaciones de puntos de datos para cada i-ésimo centroide de grupo sea S_i.

2. Paso de actualización del centroide:

   En este paso, los centroides se vuelven a calcular. Esto se hace tomando la media de todos los puntos de datos asignados al clúster de ese centroide.

   $$c_i = \frac{1}{|S_i|} \sum x_i \in S_i$$

   El algoritmo itera entre los pasos uno y dos hasta que se cumple un criterio de detención (es decir, ningún punto de datos cambia los clústeres, la suma de las distancias se reduce al mínimo o se alcanza un número máximo de iteraciones).

   Se garantiza que este algoritmo converge a un resultado. El resultado puede ser un óptimo local (es decir, no necesariamente el mejor resultado posible), lo que significa que evaluar más de una ejecución del algoritmo con centroides iniciales aleatorizados puede dar un mejor resultado.

El algoritmo encuentra los clústeres y las etiquetas de los conjuntos de datos para una K particular elegida previamente. Para encontrar la cantidad de conglomerados en los datos, necesitamos ejecutar el algoritmo de agrupamiento K-means para un rango de valores K y comparar los resultados . En general, no existe un método para determinar el valor exacto de K, pero se puede obtener una estimación precisa usando las siguientes técnicas.

Una de las métricas que se usa comúnmente para comparar los resultados en diferentes valores de K es la distancia media entre los puntos de datos y su centroide de grupo. Dado que aumentar el número de clústeres siempre *reducirá* la distancia a los puntos de datos, al aumentar K siempre disminuirá esta métrica, hasta el extremo de llegar a cero cuando K es igual que la cantidad de puntos de datos.

Por lo tanto, esta métrica no se puede usar como el único objetivo. En cambio, si se traza la distancia media al centroide como una función de K y el *"punto de inflexión"*, donde la velocidad de disminución cambia bruscamente, se puede usar para determinar aproximadamente K.

Agrupamiento mediante *clusters jerárquicos*

La técnica de clustering jerárquico construye un dendograma o árbol que representa las relaciones de similitud entre los distintos elementos. La exploración de todos los posibles árboles es un problema NP. Por lo tanto, suelen seguirse algoritmos aproximados guiados por determinadas heurísticas.

La agrupación jerárquica, como su nombre lo sugiere, funciona en base un algoritmo que construye una jerarquía de clusters. Este algoritmo comienza con todos los puntos de datos asignados a un clúster propio. Luego, dos clústeres más cercanos se fusionan en el mismo clúster. Al final, este algoritmo termina cuando solo queda un solo grupo y los resultados del agrupamiento jerárquico se pueden mostrar con dendrograma.

En un dendrograma, en la parte inferior, comenzamos con n puntos de datos, cada uno asignado a clusters separados. Dos clusters más cercanos se fusionan hasta que tenemos un solo grupo en la parte superior. La altura en el dendrograma en el que se fusionan dos clústeres representa la distancia entre dos clústeres en el espacio de datos.

La decisión del número de clusters que pueden representar mejor a los diferentes grupos se pueden elegir al observar el dendrograma. La mejor elección del número de clusters es el número de líneas verticales en el dendrograma cortadas por una línea horizontal que puede atravesar la distancia máxima verticalmente sin intersectar un grupo.

El algoritmo se puede dar con un enfoque de abajo hacia arriba, pero también es posible seguir un enfoque descendente que comience con todos los puntos de datos asignados en el mismo clúster y realizar divisiones recursivas hasta que a cada punto de datos se le asigne un clúster separado.

La decisión de fusionar dos clusters se toma sobre la base de la cercanía de estos clusters. Hay múltiples métricas para decidir la cercanía de dos clusters, así como técnicas de similitud para el aglomeramiento jerárquico que discutimos en la implementación(R., s.f.)(Murtagh y Legendre, s.f.). Dos de las métricas más comunes para hacer el cálculo de cercanía son la distancia euclidiana,

$$\|a - b\|_2 = \sqrt{\sum a_i - b_i}$$

y la distancia de Manhattan

$$\|a - b\|_1 = \sum |a_i - b_i|$$

2.1.2. Los clusters como herramienta para el análisis de conjuntos de datos

De acuerdo con la descripción anterior del agrupamiento por *clustering*, esta forma de inferir una clasificación a partir del conjunto de datos nos permite realizar una exploración de los datos sin conocimiento previo sobre ellos, pues su resultado es una partición de elementos cuyos elementos tienen una representación vectorial cercana.

De esta forma utilizamos al agrupamiento por *clustering* como una forma de explorar el conjunto de datos sin indicar al proceso la clasificación que esperamos obtener.

2.1.3. Ventajas del análisis mediante agrupamiento

- Si se analiza un conjunto de datos sobre el que no se tiene ninguna etiqueta, este análisis nos permitiría encontrar patrones estadísticos en el conjunto de datos.
- No hace falta tener una muestra clasificada *a priori* por la naturaleza no supervisada del procedimiento de agrupación.

- Es una técnica sencilla de utilizar, pues a diferencia de otras no hace ninguna suposición sobre los datos.
- Sólo hace falta representar a los ejemplares como vectores para poder aplicar esta técnica.

2.1.4. Desventajas del análisis mediante agrupamiento

- La clasificación resultante trata únicamente de cercanía entre la representación vectorial de los ejemplares. La interpretación de los resultados se puede volver complicada, pues su valor semántico no es necesariamente evidente ni significativo para el objetivo del análisis.
- La falta de suposiciones sobre el conjunto de datos limita de cierta forma el tratamiento formal de los resultados, esto hace difícil la interpretación de la clasificación.
- La representación vectorial de los ejemplares en este caso fue trivial, sin embargo esta situación no se da en todos los problemas. Esto limita la aplicabilidad de la técnica.
- Si la cercanía vectorial entre las representaciones de dos ejemplares no tiene análogo semántico para los ejemplares originales, entonces la técnica no tiene sentido.
- La complejidad del proceso para agrupar por *k-means* es $O(n^{dk+1}logn)$, donde k es el número de grupos esperados y d la dimensión del espacio de vectores de los ejemplares. Al igual que en el caso del clasificador Bayesiano-no ingenuo, esta cota limita su aplicabilidad.

Notemos que la última desventaja listada es importante: si la cercanía de la representación de dos ejemplares no implica una cercanía semántica de algún tipo de los ejemplares originales, no hay ninguna razón para suponer que los patrones estadísticos de las presentaciones tendrán significado para los datos originales. Esto pone en evidencia la importancia de la representación vectorial de los ejemplares.

2.2. Nubes de palabras y palabras comunes

Las nubes de palabras son una representación visual basada en texto de etiquetas o palabras, que típicamente usadas para mostrar la frecuencia relativa, importancia o popularidad de las etiquetas. También pueden ser usadas como un resumen visual del contenido de un documento o mostrar la evolución de la frecuencia de las etiquetas a lo largo del tiempo (Lee, Riche, Karlson, y Carpendale, 2010)

Las nubes de palabras presentan las siguientes ventajas:

1. Permiten la comparación de la importancia de varias palabras.
2. Hacen uso de un espacio compacto que puede ser reorganizado de forma flexible sin tener un impacto negativo en la facilidad de lectura de la nube.
3. Hay facilidad de lectura en la importancia de las palabras, pues está codificada directamente en el tamaño de la palabra.

Sin embargo tienen la desventaja de que no es fácil definir una escala que sea fácil de leer.

Hay dos aspectos a cuidar para elaborar una nube de palabras. Se debe decidir el tamaño de cada palabra. Típicamente esto se hace como una función lineal de la palabra o como una transformación de raíz cuadrada.

Se puede tomar especial cuidado para decidir el tamaño si hay consideraciones adicionales, como el interés en representar un parámetro adicional, como la evolución de la importancia a lo largo del tiempo.

También se debe definir la posición de cada palabra en la nube. Hay investigación sobre este problema que, como típicamente en problemas de posicionamiento, se trata de un problema NP-duro y se han propuesto muchas heurísticas (Kaser y Lemire, 2007). Por ejemplo, al igual que para el dibujado de gráficas mediante fuerzas, se puede definir una medida de atracción que modele el grado de interconexión entre etiquetas. También se pueden aplicar metaheurísticas tradicionales como recocido simulado.

Otra representación del número de repeticiones de las palabras se puede hacer mediante una gráfica de barras, que consiste de barras rectangulares con longitudes proporcionales a los valores que representan.

Al igual que las nubes de palabras, estas son utilizadas para comparar varios valores de forma gráfica. Tienen la ventaja de que permiten definir una escala que es más fácil de leer que en las nubes de palabras. En particular haremos uso de todas las herramientas expuestas anteriormente.

Capítulo 3

Descripción de la propuesta e implementación

3.1. Bibliotecas

Para nuestro proyecto decidimos apoyarnos de bibliotecas para facilitarnos tanto el manejo de datos, como su interpretación. Vale la pena analizar cada una de ellas para una mayor comprensión de las mismas y su papel a lo largo del proyecto(Tatman, s.f.)(Murrell, s.f.)(Rickert, s.f.).

- **twitteR**
  Twitter es un servicio popular que permite a los usuarios transmitir mensajes cortos ('tweets') para que otros los lean. Con el paso de los años, esto se ha convertido en una herramienta valiosa no solo para fines de medios sociales estándar sino también para experimentos de minería de datos, como el análisis de sentimientos.

  El paquete **twitteR** está destinado a proporcionar acceso a la API de Twitter dentro de R, lo que permite a los usuarios obtener interesantes subconjuntos de datos de Twitter para sus análisis.

- **tm**
  tm es un marco para aplicaciones de minería de textos dentro de R. La estructura principal para gestionar documentos en **tm** es llamado Corpus, que representa una colección de documentos de texto.

  Un corpus es un concepto abstracto, y puede haber varias implementaciones en paralelo.

  Dentro del constructor del corpus, *x* debe ser un objeto fuente que abstraiga la ubicación de entrada. **tm** proporciona un conjunto de fuentes predefinidas, por ejemplo, *DirSource*, *VectorSource* o *DataframeSource*, que manejan un directorio, un vector que interpreta cada componente como documento o estructuras similares a marcos de datos (como archivos CSV), respectivamente. Excepto *DirSource*, que está diseñado únicamente para directorios en un sistema de archivos, y *VectorSource*, que solo acepta vectores (caracteres), la mayoría de las otras fuentes implementadas pueden tomar conexiones como entrada (una cadena de caracteres se interpreta como ruta de archivo).

- **syuzhet**
  Es un paquete viene con cuatro diccionarios de opinión y proporciona un método para acceder a la robusta

herramienta de extracción de sentimientos desarrollada en el grupo PNL en Stanford. Después de cargar el paquete (library (syuzhet)), comienzas por analizar un texto en un vector de oraciones. Para esto, utilizará la función *get_sentences()* que implementa el tokenizer de sentencia openNLP.

- **wordcloud2**
  Este paquete proporciona una interfaz HTML5 para wordcloud para la visualización de datos. Timdream's wordcloud2.js se usa en este paquete, para desarrollar una nube de palabras. Una nube de palabras (o nube de etiquetas) es una representación visual de datos de texto. Las etiquetas suelen ser palabras sueltas, y la importancia de cada etiqueta se muestra con el tamaño o el color de la fuente. Este modo de representación es útil para percibir rápidamente los términos más destacados en una lista y determinar sus prominencias relativas.

  Para hacerlo, antes de poner cada palabra en el lienzo, se dibuja en un lienzo separado para volver a leer los píxeles para registrar los espacios dibujados. Con la información, wordcloud.js intentará encontrar un lugar que se adapte a la palabra más cercana al punto de inicio.

- **NLP**
  El procesamiento del lenguaje natural (generalmente abreviado como "NLP") es la tarea de extraer y resumir automáticamente la información de los datos de texto, y este paquete te lo facilita.

  La tarea de NLP es identificar automáticamente los temas principales en un texto, por lo general mediante la identificación de palabras informativas.

- **graphics**
  Este paquete proporciona las funciones para comenzar a producir gráficos en R. Los gráficos R siguen un "modelo de pintores", lo que significa que la salida de gráficos se produce en pasos, y que la salida posterior oscurece cualquier salida anterior que se solape.

- **Tidyverse**
  Tidyverse es un sistema coherente de paquetes para la manipulación, exploración y visualización de datos que comparten una filosofía de diseño común. Estos fueron en su mayoría desarrollados por el mismo Hadley Wickham, pero ahora están siendo expandidos por varios contribuyentes. Los paquetes Tidyverse están destinados a hacer que los estadísticos y los científicos de datos sean más productivos al guiarlos a través de flujos de trabajo que facilitan la comunicación y dan como resultado productos de trabajo reproducibles. Fundamentalmente, el tidyverse trata de las conexiones entre las herramientas que hacen posible el flujo de trabajo.

3.2. Implementación del sistema

Previo a analizar propiamente el código y las estructuras que fueron necesarias definir para llevar a cabo el *clustering* de los datos con base en las observaciones cualitativas y cuantitativas de los *tuits*, así como el análisis de minería de textos y de análisis de sentimientos que se llevó a cabo, describamos adecuadamente a los ejemplares u observaciones sobre las cuales trabajamos para estudiar su estructura.

3.2.1. Obtención de ejemplares

Parte de la motivación del proyecto surgió en torno a la posibilidad de desarrollar una aplicación *web* que presentara una facilidad de uso para el usuario final y le permitiera llevar a cabo análisis posteriores sin la necesidad de haber cursado ningún curso como el que representa *Reconocimiento de Patrones y Aprendizaje Automatizado*. Por esta razón, desarrollamos una interfaz de usuario que permitiera la obtención de los *tuits*[1] con la ayuda de la biblioteca de `twitteR` provista por `R`.

Usamos nosotros también esta interfaz creada para la obtención de los datos de estudio. Primeramente tuvimos para ésto que registrar nuestra aplicación en la página de Twitter Apps para tener acceso entonces a las facilidades de la *API* (interfaz de desarrollo de aplicaciones) como la de la obtención masiva de *tuits*. Al registrarnos obtuvimos:

- api_key: TI8NRQd65Ijuc9m5v6VFTxZUk
- api_secret: gdNFt7ht4YF10hNXEbHq9mrhTLQTbXCr3gb7iXtL3WSg73kUFU
- access_token: 1003785782228934656-UO4b26k2LCQ4Wt0xDWiSpiEKNhwWQS
- access_token_secret: Ux7a6HWEjjnD48mh7wEM2UaOTE2dJMeicTEjoC69IUSGO

Con estos datos resultó suficiente para poder autenticar la cuenta desde `R Studio`, para así poder usar la función `searchTwitter` con los parámetros dados por el usuario para buscar *tuits* a partir de un *hashtag* en particular, tantos como fueran indicados y a partir de la fecha provista desde la interfaz como puede verse en el código:

```
setup_twitter_oauth(api_key, api_secret, access_token, access_token_secret)
tweets <- searchTwitter(hashtag, n=as.numeric(input$numTweets), since=as.character(input$fecha)
tweets_df <- twListToDF(tweets)
write.csv(tweets_df, file=file.choose(), row.names=F)
```

También convertimos los *tuits* recopilados a un *dataframe* (formato tabular) para después poderlos escribir fácilmente en la ruta especificada en formato de *Comma-Separated Values* (`CSV`).

Descripción del conjunto de datos

Los datos que se presentan por omisión en la aplicación desarrollada fueron pensados para poder tratar a cada uno de los cuatro candidatos presidenciales para las elecciones de México-2018 usando los hashtags más importantes para referirse a ellos[2]:

- Andrés Manuel López Obrador, candidato por la coalición *Juntos Haremos Historia* de los partidos políticos *Morena*, *PT* y *PES*. Se usó el *hashtag* **#AMLO**.
- Ricardo Anaya Cortés, candidato por la coalición *Por México al Frente* de los partidos políticos *PAN*, *PRD* y *Movimiento Ciudadano*. Se usó el *hashtag* **#RicardoAnaya**.
- José Antonio Meade Kuribreña, candidato por la coalición *Todos por México* de los partidos políticos *PRI*, *PVEM* y *PANAL*. Se usó el *hashtag* **#Meade**.

[1] Los llamamos indistintamente *tuits* o *tweets*.
[2] No consideramos *tuits* en torno a Margarita Zavala luego de que abandonó la campaña electoral.

- Jaime Rodríguez Calderón (*El Bronco*), candidato independiente. Se usó el *hashtag* **#ElBronco**.

De igual manera recopilamos *tuits* de los *hashtags* **#INE** y *#Elecciones2018* con tal de tener un análisis de las opiniones que se tienen acerca del órgano electoral y de las elecciones en el país en general.

Cabe mencionar que el número de *tuits* recabados por cada tema fueron mil (1000) para poder hacer un análisis de tamaño considerable. También, se recopilaron solamente aquellos a partir de las fechas de inicios de campaña, es decir, el 14 de diciembre del 2017.

La información de cada *tweet* que nos proporciona la función de búsqueda en `R` antes descrita nos aporta:

- text: El contenido del *tuit*.
- Información adicional como número de *re-tuits*, si ha sido declarado como *favorito*, longitud y latitud en donde se escribió, etcétera:
  - favorited
  - favoriteCount
  - replyToSN
  - created
  - truncated
  - replyToSID
  - id
  - replyToUID
  - statusSource
  - screenName
  - retweetCount
  - isRetweet
  - retweeted
  - longitude
  - latitude

Como se verá en la siguiente subsección de implementación, en nuestro caso después llevamos a cabo un pre-procesamiento al leer los `CSVs` para considerar solamente el contenido mismo de los *tuits*, esto es, el texto.

Cabe recalcar que como hemos mencionado antes, la idea es que el usuario pueda recabar datos por su propia cuenta, así que estos datos representan solamente los que fueron analizados de forma exhaustiva para el reporte.

3.2.2. Implementación en R

En caso de querer revisar el código en nuestro repositorio de `GitHub`, seguir el siguiente *link*:
Repositorio de GitHub

A raíz de la motivación de presentar una aplicación *web* de fácil uso para el usuario final, integramos la biblioteca de `Shiny`, enfocada a comunicarse con las funciones de `R`. Este paquete permite, por ejemplo, crear interfaces de usuario con algunos componentes predefinidos; en nuestro caso creamos un `pageWithSideBar` en el que se logran identificar tres componentes principales:

- headerPanel: Su única función es tener el título de la aplicación *web*.
- sidebarPanel: El componente aparece en la parte lateral izquierda. Aquí colocamos varios páneles condicionales cuya aparición está condicionada directamente por el tipo de análisis quiera llevar a cabo el usuario: clustering, análisis de minería de texto acerca de al frecuencia de términos con gráficas de barras o nube de palabras, o bien, un análisis de sentimientos. Cada uno de los componentes anidados puede servir para recibir la entrada del usuario y posteriormente ser usada para parametrizar las funciones de `R` desde el lado del servidor que atiende las peticiones de la aplicación *web*.
- mainPanel: Aquí se despliegan (lateral derecho) las gráficas resultantes y gráficos que sean creados por el servidor; son renderizados desde el servidor.

Luego de recibir la entrada del usuario desde la UI (*User Interface*), es importante mencionar la forma en que se logró la implementación de la funcionalidad que es atendida desde el servidor.[3]

- Bajar tweets: Lo que hacemos es captar la entrada del usuario en caso de que quiera realizar esta acción y recibir también un tema no contemplado por defecto para poder pasarlo como parámetro a la función `searchTwitter()` descrita en la subsección de obtención de los datos. Parámetros que también son usados y que son recibidos desde la interfaz del usuario son el número de *tuits* a recabar y la fecha de inicio desde la que deberán ser considerados[4]. El procedimiento ya fue indicado a detalle también en la subsección anterior, desde la autenticación de la cuenta con los datos proporcionados por la versión de desarrollador hasta la escritura de los archivos `CSV`.
- Analizar sentimientos (minería de opinión): Luego de importar las bibliotecas para la minería de textos (`tm`) y la que tiene los diccionarios de palabras con sentimientos (`syuzhet`), lo que hacemos inicialmente es leer el archivo `CSV` de interés y quedarnos únicamente con la sección del texto:

```
tweets <- read.csv(ruta_archivo, header = T)
sentimientos_terminos <- iconv(tweets$text)
```

  Posteriormente, basta con llamar al diccionario de NRC de sentimientos incluido en el paquete de `syuzhet` para poder identificar los seis sentimientos y dos emociones y sus valencias que fueron recabadas por personas. El lexicón de sentimientos y emociones que supone también tiene versiones en otros idiomas, por lo que no es obstáculo para la naturaleza de nuestros *tuits*. Las ocho emociones analizadas son:

[3] Se describe el proceso en general, aunque como tal en la aplicación se presentan dos alternativas para cada escenario: usando rutas a los archivos ya existentes o permitiendo al usuario seleccionar archivos propios o recabados para nuevos análisis; el procedimiento en el fondo es el mismo en ambos casos.

[4] Cabe mencionar que en ocasiones la `API` de `Twitter` podrá no regresar el número de *tuits* solicitados dado que no hay suficientes con el *hashtag* indicado o bien por las condiciones en sus políticas.

- anger (ira)
- fear (miedo)
- anticipation (anticipación)
- trust (confianza)
- surprise (sorpresa)
- sadness (tristeza)
- joy (felicidad)
- disgust (repulsión)

Además de los dos sentimientos principales: positive (positivo) y negative (negativo). El código fue el siguiente, en donde al final basta con mostrar una gráfica de barras con la puntuación por sentimiento de cada *tuit*:

```
sentimientos <- get_nrc_sentiment(sentimientos_terminos)
output$grafica <- renderPlot({barplot(colSums(sentimientos),
las = 2, col = rainbow(10), ylab = 'Cuenta', main = 'Sentimientos')})
```

- Análisis de frecuencia de términos y nube de palabras: En estos escenarios, luego de leer el archivo seleccionado o bien dado por la ruta predefinida, primeramente se crea un *corpus* o colección de los documentos, en donde cada *tuit* por sí mismo es tratado como un documento a partir de la representación vectorial de los términos o palabras que exhibe en su sección de texto:

```
tweets <- read.csv(ruta_archivo, header = T)
corpus <- iconv(tweets$text)
corpus <- Corpus(VectorSource(corpus))
```

Posteriormente se lleva a cabo un preprocesamiento de los términos en los documentos (*tuits*) del corpus con tal de limpiarlos y quitarles términos y símbolos innecesarios que pueden opacar el estudio en cuestión. Lo que hacemos es:

- Convertir todos los caracteres a minúsculas.
- Quitar los signos de puntuación.
- Quitar los caracteres numéricos.
- Definir una función sencilla que por medio de una expresión regular simple nos ayude a quitar los *URLs* de los *tuits*.
- Cargar una lista de *stopwords* que construimos para poder ignorar las palabras más frecuentes en el idioma que no contribuyan semánticamente. Esto lo logramos al añadir las palabras del archivo en formato `CSV` a la lista original de *stopwords* y finalmente llamando a la función `removeWords` al realizar un *mapeo* sobre el vector de términos del *tuit*.
- Opcionalmente, en caso de que se trate de un tema particular identificado, es posible aplicar la función antes mencionada para además poder descartar algunas palabras o términos que evidentemente serán los que más aparezcan en el *tuit* por tratarse del propio *hashtag* o de palabras muy parecidas y que nuevemente, pueden opacar el estudio.

El código para este parte es:

```
corpus <- tm_map(corpus, tolower)
corpus <- tm_map(corpus, removePunctuation)
corpus <- tm_map(corpus, removeNumbers)
removeURL <- function(x) gsub('http[[:alnum:]]*', '', x)
cleanset <- tm_map(corpus, content_transformer(removeURL))
stopwords <- read.csv("../Data/stopwords.csv", header = FALSE)
stopwords <- as.character(stopwords$V1)
stopwords <- c(stopwords, stopwords())
cleanset <- tm_map(corpus, removeWords, stopwords)
cleanset <- tm_map(cleanset, stripWhitespace)
```

Después basta con crear una matriz de términos, sumar la frecuencia de cada uno en todos los documentos de la colección o corpus y finalmente quedarnos con aquellos que aparecen más de un número especificado por el usuario para proceder a graficarlos:

```
tdm <- TermDocumentMatrix(cleanset)
tdm <- as.matrix(tdm)
apariciones <- rowSums(tdm)
apariciones <- subset(apariciones, apariciones >=
as.numeric(input$min_apariciones))
output$grafica <- renderPlot({barplot(apariciones,
las = 2,
                    col = rainbow(50),
                    ylab = 'Num._Apariciones',
                    xlab = 'Palabra',
                   cex.names = 0.5)})
```

En el caso de querer formar una nube de palabras, basta con crear una estructura tabular (*dataframe*) de los términos con sus frecuencias correspondientes de aquellos que aparecieron más del número especificado de veces. Luego, renombramos las columnas de la tabla y finalmente construimos la nube con la función de `wordcloud2` con los parámetros especificados por el usuario en la interfaz:

```
apariciones <- data.frame(names(apariciones), apariciones)
colnames(apariciones) <- c('word', 'freq')
output$grafica <- renderPlot({wordcloud2(apariciones,
                    shape = input$forma, size = as.numeric(input$tamanio),
                    minSize = as.numeric(input$tam_chico))})
```

- Clustering por *k-means*: Siguiendo el mismo patrón que el antes descrito y la técnica ya explorada en el tercer proyecto, luego del apropiado preprocesamiento del archivo luego de su lectura como se hizo en los casos anteriores, lo que hacemos ahora es crear la matriz de distancias entre los *tuis* existentes que son los que conforman el corpus de documentos.

```
dtm = DocumentTermMatrix(cleanset)
distancias = dist(dtm)
```

Dependiendo de la sub-acción especificada por el usuario, es posible graficar las inercias entre los distintos clusters para darse una idea de cuántos conviene utilizar. Para esto, consideramos el vector de las inercias e

iteramos los centroides para ir almacenando en el vector las inercias obtenidas en cada caso; luego, podemos graficar el vector:

```
vector <- kmeans(distancias, centers = 1)$betweens
for(i in 2:15) vector[i] <- kmeans(distancias, centers = i)$betweenss
output$grafica <- renderPlot({plot(1:15, vector,
                    type = "b", xlab = "num._clusters",
                    ylab = "inercia_inter-cluster")})
```

En caso de ya querer aplicar el *clustering*, simplementa basta con almacenar los resultados de aplicar el algoritmo con los parámetros dados por el usuario a través de la interfaz y luego graficar cómo fueron agrupados los ejemplares (*tuits*) en los disferentes clusters, mandando este render a la salida:

```
resultados <- kmeans(distancias,
                centers = as.numeric(input$num_clusters),
                algorithm = input$algoritmo)
output$grafica <- renderPlot({plot(resultados$cluster,
                 xlab="Tweet", ylab ="cluster",
                 col = resultados$cluster)})
```

3.2.3. Guía de uso

Con tal de poder —tanto para replicar como parametrizar la aplicación *web*— usar el programa, bastará con abrir ambos archivos `ui.R` y `server.R` en el directorio indicado por el sistema operativo en donde vaya a ser probado con `R Studio`. Posteriormente, en la parte superior se deberá dar *click* en `Run App` o bien escribir en la consola de `R Studio` `runApp()` luego de haber especificado con `setwd()` el directorio de trabajo en donde se encuentran los archivos en este caso.

Al llevar a cabo la acción previa, se abrirá la aplicación *web* en una ventana emergente. También se indicará la dirección en que está trabajando el servidor y el puerto específico, en caso de querer abrirlo con algún navegador *web*. Finalmente, bastará con indicar lo que se desea efectuar en la interfaz de la aplicación y tener en cuenta que algunos de los análisis pueden tomar alrededor de veinte segundos (dependiendo de la capacidad de cómputo) antes de poder desplegar los resultados[5].

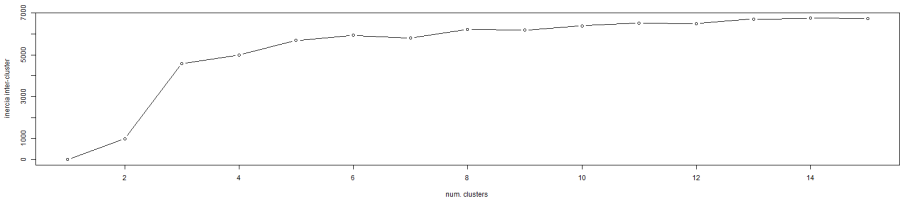
3.2.4. Resultados

Analicemos los resultados obtenidos en los distintos rubros del análisis ofrecido por nuestra aplicación.

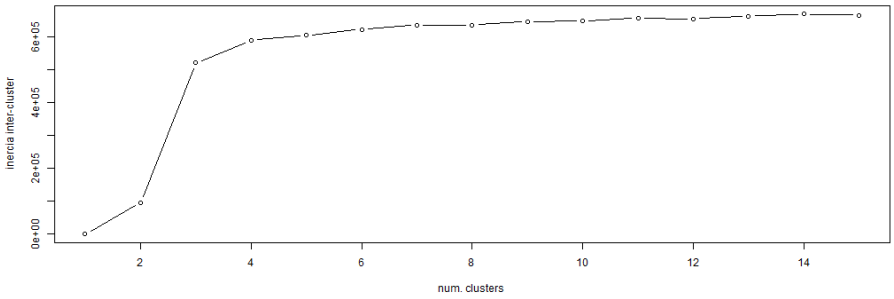
- Clustering por *k-means*:

  Al graficar las inercias de los clusters por cada tema predefinido obtuvimos los siguientes resultados:

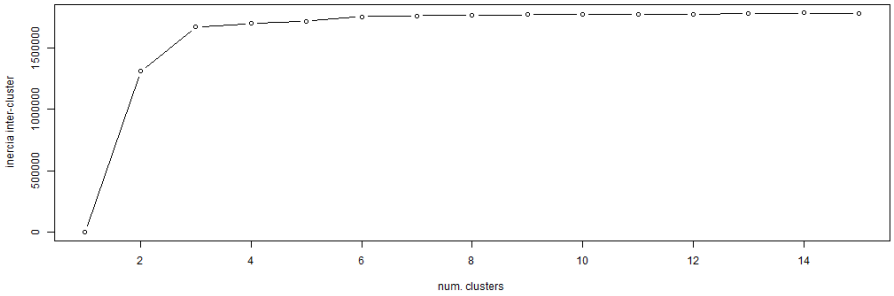
[5]En el caso de las nubes de palabras, éstas serán desplegadas en el visualizador integrado de `R Studio` en la sección de `plot` dado que la función de `renderWordcloud2` no funcionó y tuvimos que tratar con la función de renderPlot tradicional.



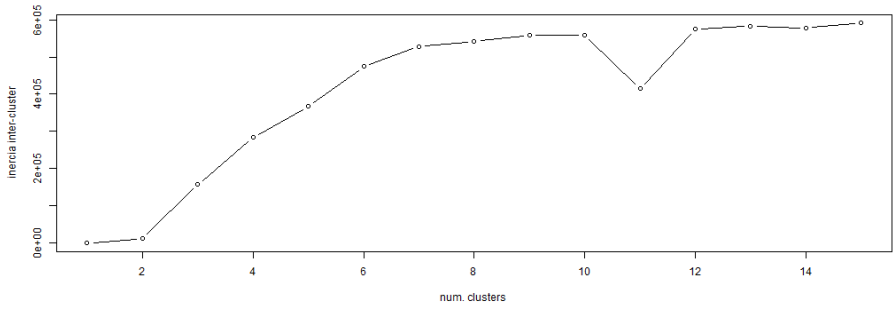
#AMLO.



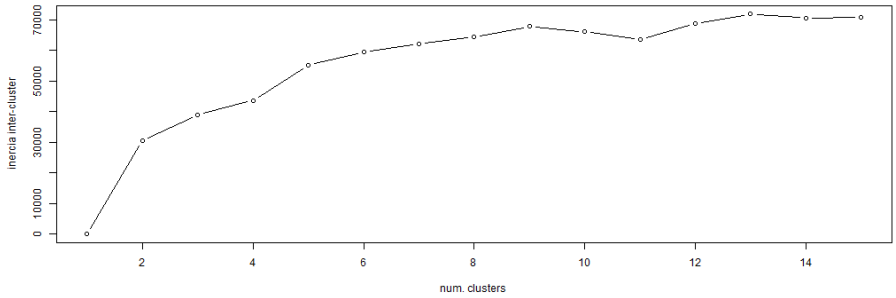
#RicardoAnaya.



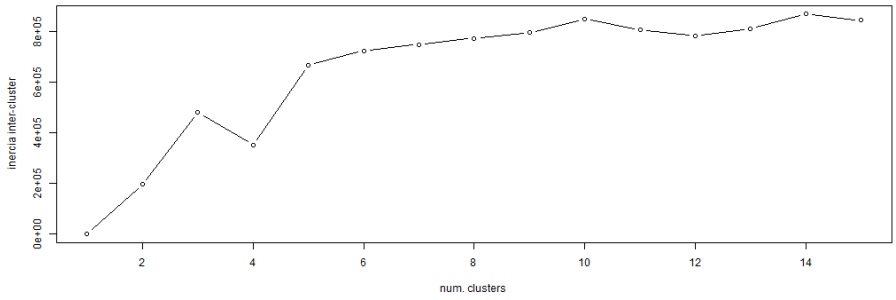
#Meade.



#ElBronco.

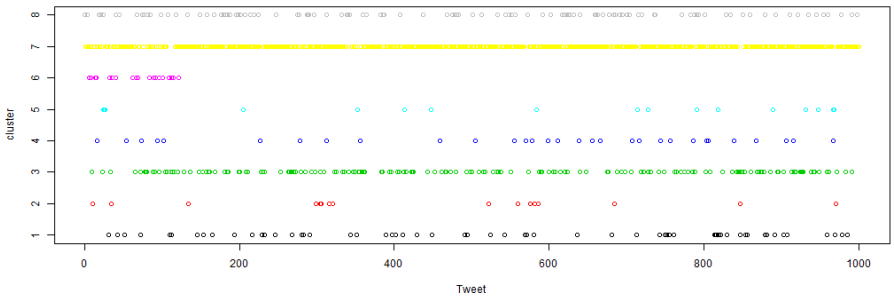


#Elecciones2018.

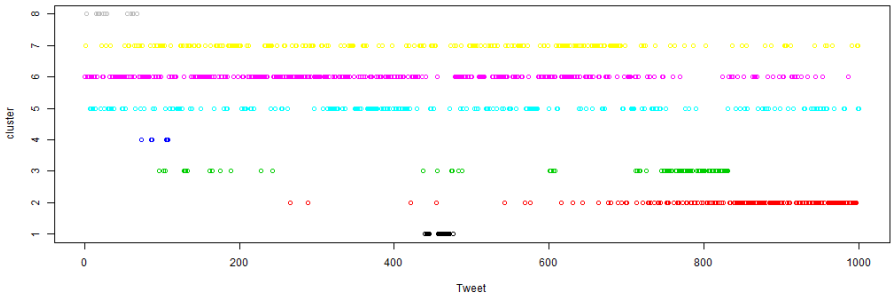


#INE.

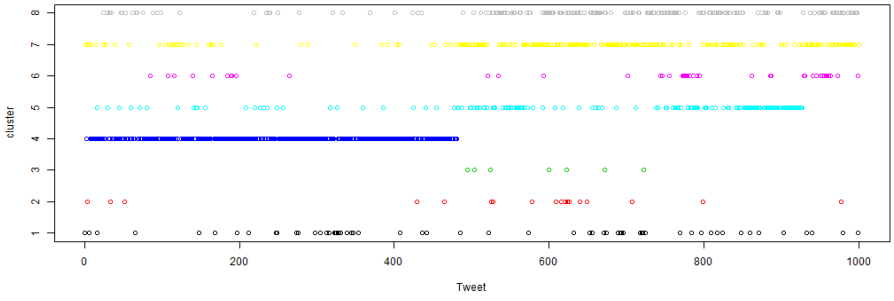
Es posible ver que los clusters sugeridos a partir de estas gráficas son aproximadamente ocho en el caso general. Aplicando *clustering* con ocho centroides obtenemos:



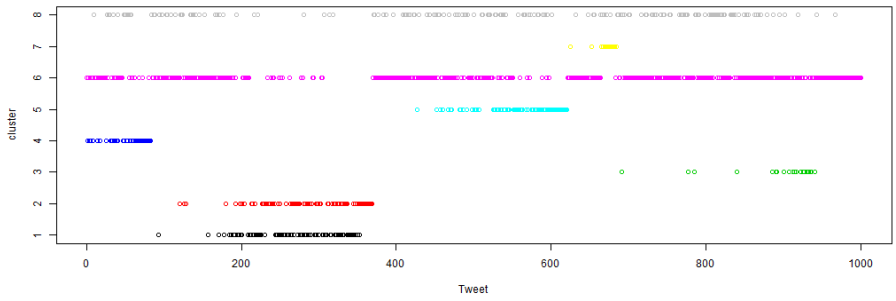
#AMLO.



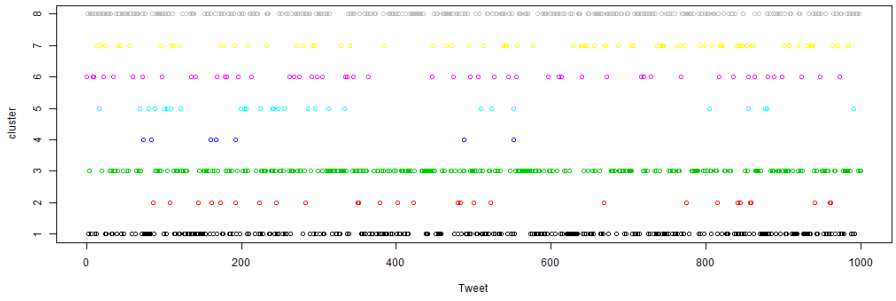
#RicardoAnaya.



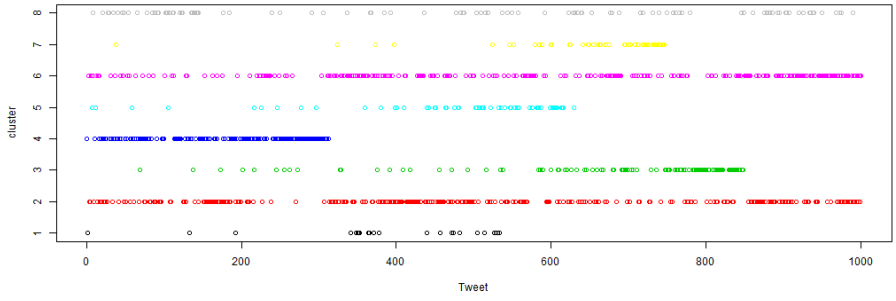
#Meade.



#ElBronco.



#Elecciones2018.



#INE.

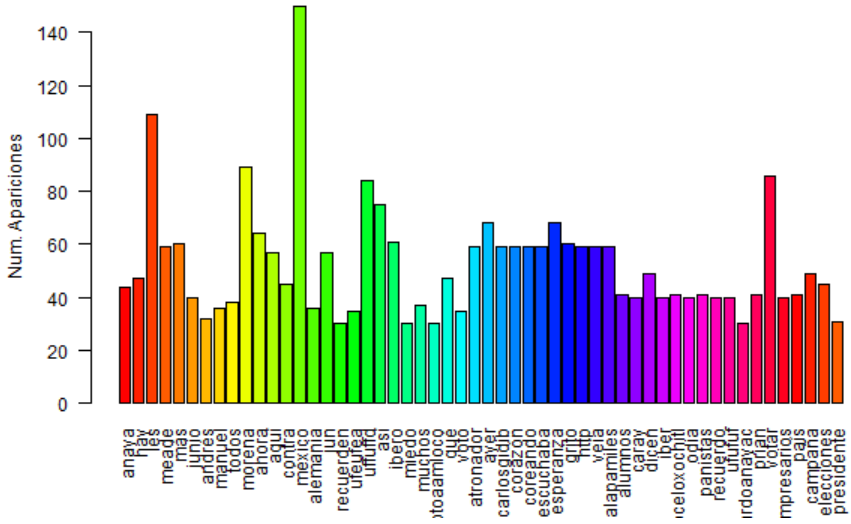
En donde vemos cómo cada miembro del corpus (colección de documentos) que son los *tuits* consideramos son agrupados en los clusters. Resulta interesante cómo el número de clusters recomendados y la clasificación tiene una correlación directa con las emociones y sentimientos considerados en los diccionarios de sentimientos; en particular, está estrechamente relacionado con las emociones del diccionario de la biblioteca

de `syuzhet`.

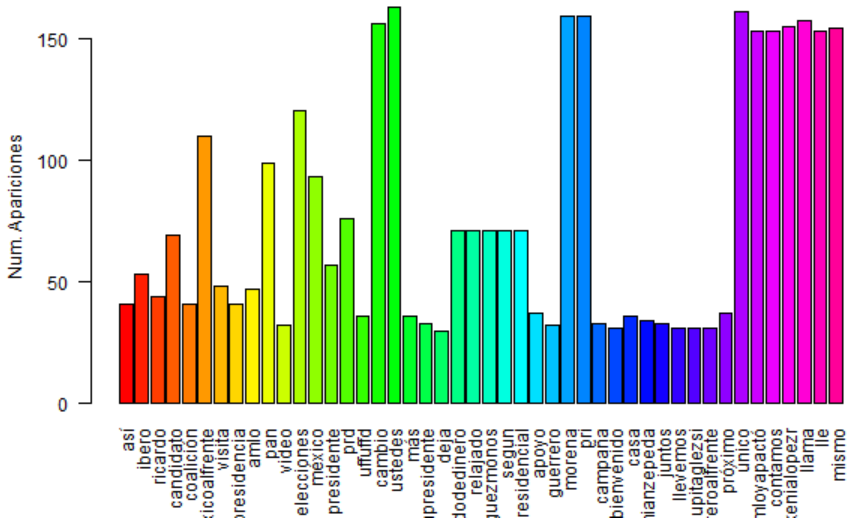
No obstante, también es posible apreciar que aquellos temas o *hashtags* en los que las inercias se establilizan más rápido corresponden a los candidatos o temas electorales que generan menos emociones por parte de los usuarios de *Twitter*, como es el caso de José Antonio Meade, por ejemplo. Las gráficas de sentimientos para poder obtener estas relaciones se presentan más adelante.

- Frecuencia de términos:

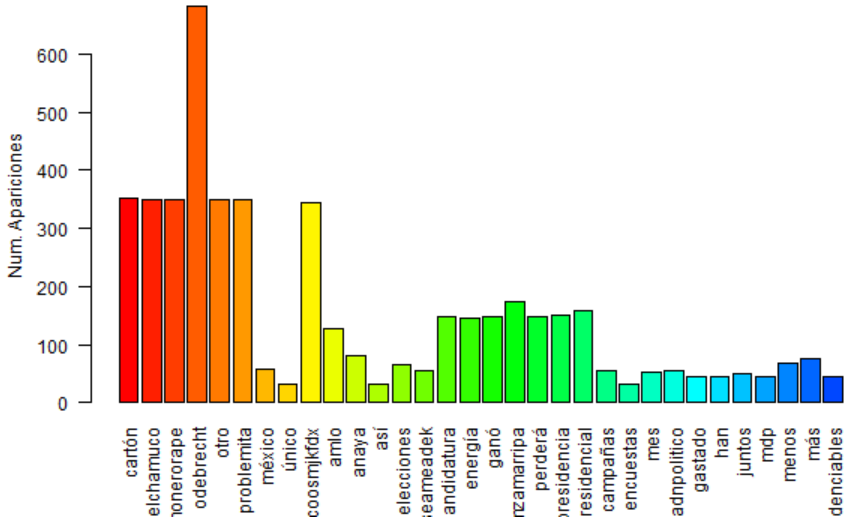
  Concentrándonos ahora en un estudio cuantitativo de las frecuencias de los términos o palabras más usadas dentro de los *hashtags* electorales obtenemos las siguientes gráficas:



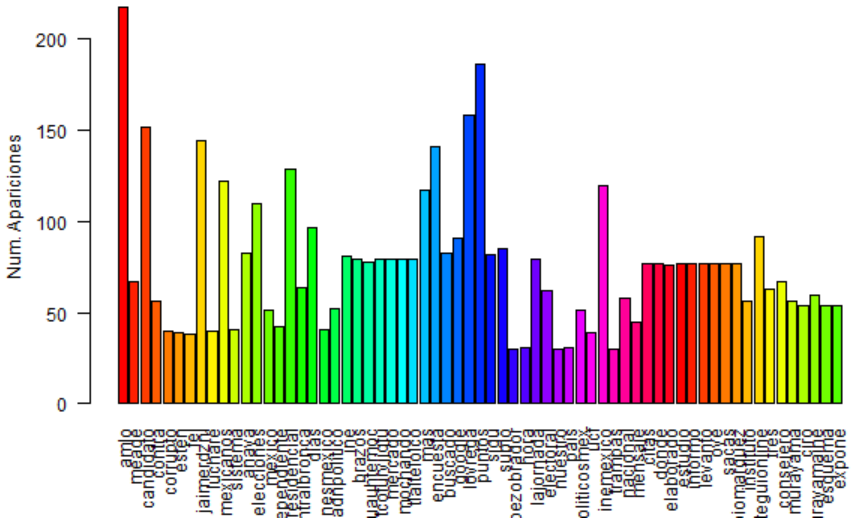
#AMLO.



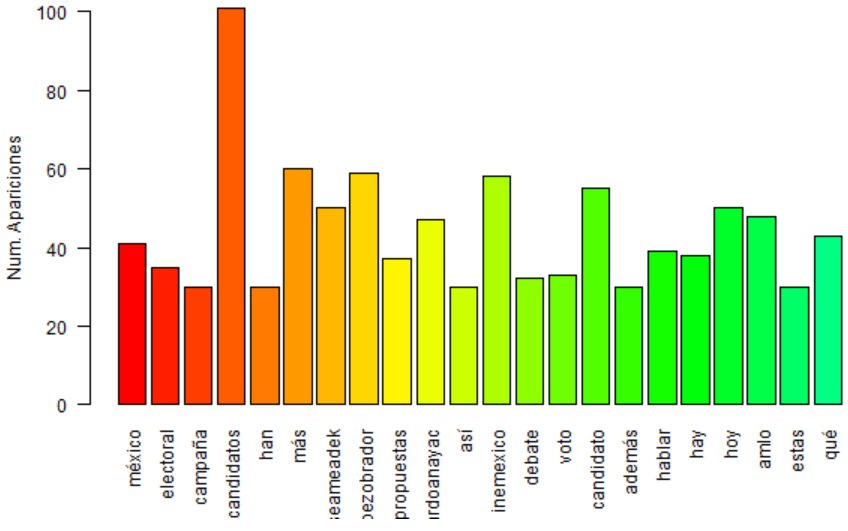
#RicardoAnaya.



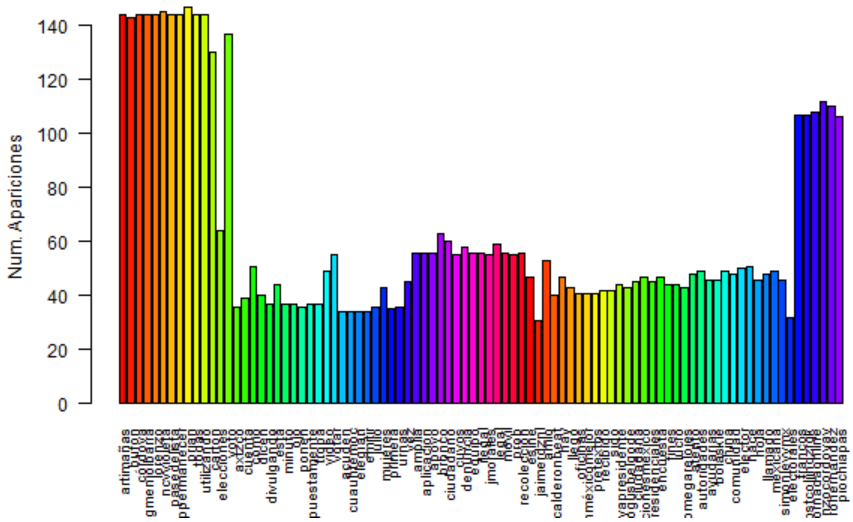
#Meade.



#ElBronco.



#Elecciones2018.



#INE.

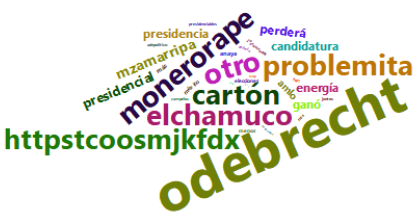
En este caso podemos ver cómo es posible llevar a cabo análisis típicos de minería de textos para poder encontrar estrechos vínculos entre la realidad política que se vive en el país con lo que está hablando la gente en redes sociales. Al restringir el número de apariciones mínimas de cada palabra o término para poder ser considerado en las gráficas, también podemos extraer información acerca de qué temas invitan a las personas a usar un léxico más variado. Vemos —por ejemplo–, que en el caso de #Elecciones2018, los términos no son tan extensos, mientras que con las #Elecciones2018 o #INE sí lo son.

- Nubes de palabras:

  De manera similar a la parte previa, las nubes de palabras nos aporta una forma visual muy entendible de los términos más usados. Nuestros resultados para los temas por defecto considerados fueron los siguientes:

#AMLO.

#RicardoAnaya.



#Meade.

#Elecciones2018.

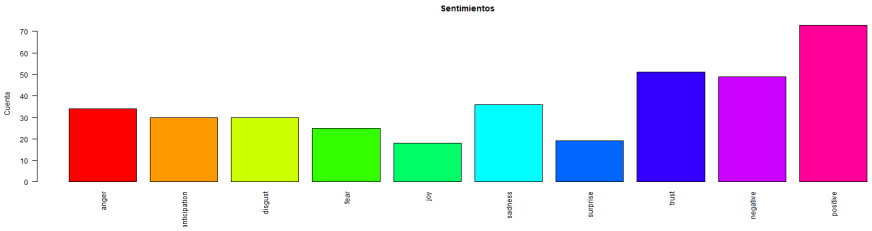
#ElBronco.

#INE.

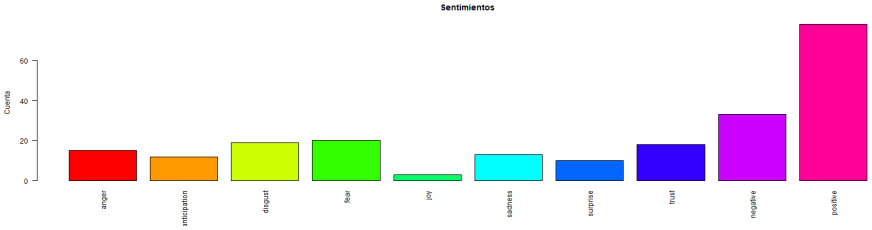
Nuevamente es posible apreciar cómo el tema electoral global se restringe a usar pocos términos. De igual forma, es fácil ver cómo *morena* parece ser uno de los objetos de los que más se habla independientemente del candidato. Finalmente, para cada candidato ocurre que entre los términos más empleados tienen que ver justamente con las controversias en que se ha visto involucrado y demás temas polémicos que lo rodean.

- Análisis de sentimientos:

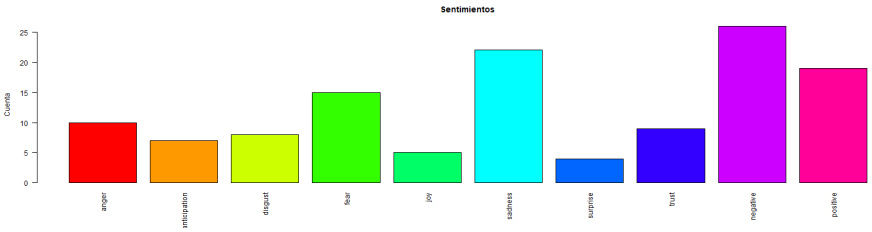
  La parte medular del análisis radica en la minería de opinión, en específico del análisis de sentimientos que generan los presidenciables y demás temas electorales entre los usuarios de *Twitter*. Usando el diccionario de `syuzhet` ya analizado en secciones previas, los resultados obtenidos fueron éstos:



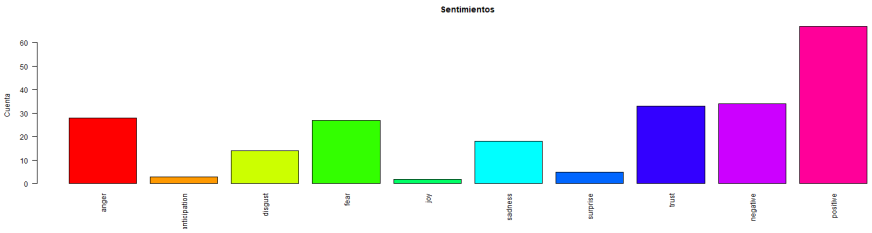
#AMLO.



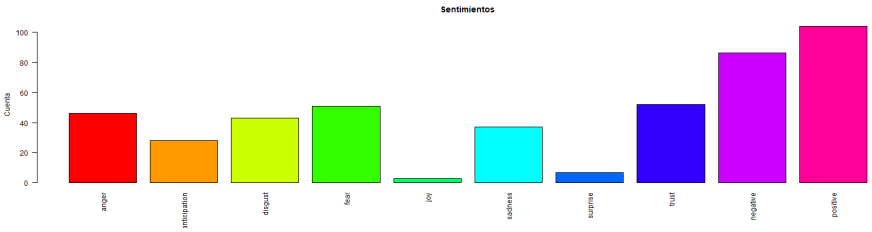
#RicardoAnaya.



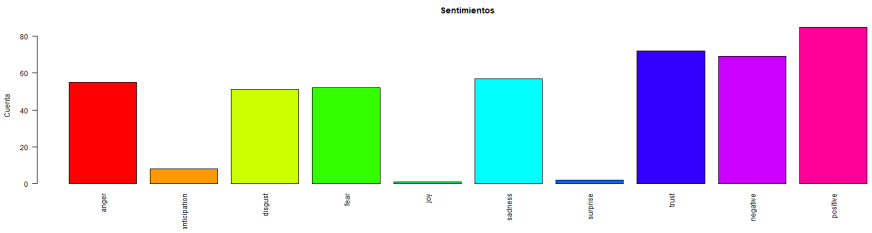
#Meade.



#ElBronco.



#Elecciones2018.



#INE.

Conviene notar la forma en la que los dos punteros actuales en las encuestas —Andrés Manuel López Obrador y Ricardo Anaya Cortés— tienen un desempeño similar en términos de generación de sentimientos en la gente en lo que respecta a los *hashtags* que los identifican. Si bien ambos temas generan muchas emociones positivas en lo que reflejan los *tuits*, en el caso de Ricardo Anaya Cortés es posible destacarle que no genera tantas emociones negativas como su contrincante. Por el otro lado, Andrés Manuel genera más confianza entre los ciudadanos mexicanos que han escrito con el hashtag *#AMLO*. Todas las demás emociones, tanto con una connotación negativa como las positivas, son más fuertes del lado de Obrador.

En caso del candidato del *PRI*, es fácil notar cómo los sentimientos negativos imperan frente a los positivos, y las emociones de tristeza son muy altas en contraste con las demás. Al igual que el candidato Ricardo Anaya, las emociones de sorpresa con muy bajas, pero superior la alegría que genera frente a Cortés.

Llegando al candidato independiente, es interesante notar cómo la confianza y la negatividad generadas en los *hashtags* que lo representan son casi de la misma magnitud, pero la ira que ha generado es relativamente alta. Podemos inferir que se debe a la propuesta de *mocharle la mano a los delincuentes*, pues podemos ver que *mochando* es uno de los términos destacados en la nube de palabras del candidato y que tiene una valencia alta en términos de ira en los diccionarios de emociones y sentimientos como el que aplicamos.

De entre todos los candidatos presidenciabes, podemos ver que Andrés Manuel es el que más emociones positivas genera, luego le siguen empatados Anaya y El Bronco y al final tenemos a Meade, siendo el único con más negatividad que sentimientos positivos. También Andrés Manuel es el que más sentimientos de confianza genera. Haciendo un análisis global, el candidato que genera más sentimientos de ira, tristeza y desconfianza es José Antonio Meade.

Analizando ahora al tema electoral de forma general es fácil ver que en su mayoría genera emociones positivas, pero también muchas negativas, casi a la par. Es sorpresivo ver cómo genera tanto miedo como confianza, pero la anticipación y la sorpresa son muy bajos. En cuanto al INE, la situación es similar: genera poca anticipación, alegría y sorpresa y los sentimientos positivos son casi tan altos como los negativos. Se hace notar la ira, desconfianza, miedo y repulsión por parte de los mexicanos que crean *tuits* usando el *hashtag #INE*.

Capítulo 4

Conclusiones

Resulta imperativo llevar a cabo este tipo de análisis aprovechando la teoría existente en materia de minería de datos y la línea de investigación de aprendizaje automatizado que existe hoy en día como complemento a las técnicas habituales que existen en periodos electorales con tal de acercar a la gente a la esfera política nacional y hacer ver a las instituciones, órganos gubernamentales, casas encuestadoras, militantes políticos, periodistas y gente involucrada en las campañas que la objetividad es fundamental en estos procesos.

Por medio de una adecuada implementación de una aplicación *web* con una entendible interfaz gráfica es además posible contribuir a que gente no experimentada en la teoría pueda hacer uso de los beneficios que ofrece para llevar a cabo análisis complejos.

Si bien es cierto que las noticias son variadas, las opiniones de la gente variopintas y día con día suceden varios acontecimientos, es posible usar este tipo de herramientas y aplicaciones para estarse actualizando constantemente y poder tener un entendimiento de la sociedad y diversos temas que van más allá o trascienden lo que dejan ver por encima las noticias, logrando identificar modas, sentimientos y preferencias de todo tipo de individuos.

Evitaremos hacer comentario político y nos restringiremos a describir los resultados mediante un análisis cuantitativo de los datos que se obtuvieron con el análisis, y hacer conclusiones sobre los métodos empleados en el análisis.

4.1. Análisis mediante agrupamiento por *clustering*

Estas técnicas resultan útiles para realizar una exploración inicial de un conjunto de datos no etiquetado, pues al no hacer suposiciones del conjunto se puede aplicar a una gran variedad de situaciones siempre que se tenga una representación vectorial apropiada de los ejemplares.

La interpretación de los resultados puede resultar difícil, sin embargo es posible que devuelva un resultado útil. Desafortunadamente no es necesariamente evidente encontrar una representación vectorial de los ejemplares que resulte apropiada.

En resumen, la aplicación directa de las técnicas exploradas resulta útil únicamente cuando la representación vectorial de los ejemplares es tal que la cercanía de la representación de dos ejemplares implica similitud de los

ejemplares originales, y aún cuando se cumple esa situación puede resultar difícil encontrar el criterio de semejanza en los ejemplares originales.

En el conjunto de datos analizado en este proyecto las preocupaciones expuestas están mitigadas por el hecho de que los ejemplares ya eran vectores y la similitud entre los objetos que representaban estaba bien modelada por la distancia entre los vectores. Por estas razones el análisis ofreció los resultados esperados.

4.2. Análisis mediante minería de textos y análisis de sentimientos

La aplicación de las técnicas expuestas en cuanto a la minería de datos y el análisis de sentimientos mostró que se pueden obtener resultados que tienen utilidad y son consistentes en su mayor parte.

Por ejemplo, las nubes de palabras en el caso la etiqueta *#Meade* resaltan uno de sus escándalos, lo cuál indica que los usuarios de la red social tienen presente esta situación con el candidato al que representa la etiqueta. Esto se ve reflejado en el análisis de sentimientos, teniendo una mayor presencia los tuits con una orientación negativa que aquella de los de orientación positiva. Es decir, partiendo de cualquiera de los dos análisis se podría llegar a una conclusion similar (que en este momento el discurso alrededor de la etiqueta no es positivo), lo cuál indica una consistencia en el comportamiento de ambas técnicas.

4.3. Conjunto de datos

Observamos que las propiedades de las semillas son indicadores más o menos apropiados de las semillas a las que representan. Esta conclusión se justifica observando que contábamos con una clasificación del conjunto de datos explorados, lo cuál nos permitió comparar los resultados obtenidos con los resultados esperados.

Esto parece indicar que las técnicas exploradas tienen su mayor potencial de utilidad en aquellos conjuntos de datos cuya representación vectorial sea natural, como fue en el caso de las semillas.

Referencias

Andrea Ceron, S. M. I., Luigi Curini. (s.f.). *Using sentiment analysis to monitor electoral campaigns*. Sage Journals, Social Science Computer Reviews.

Bo Pang, L. L. (2008). *Opinion mining and sentiment analysis*.

Carvalho, C. M., Nagano, H., y Barros, A. K. (s.f.). *A comparative study for sentiment analysis on election brazilian news*. Laboratorio de Processamento da Informacao Biológica, Universidade Federal do Maranhao.

Hamling, T., y Agrawal, A. (s.f.). *Sentiment analysis of tweets to gain insights into the 2016 us election*. Columbia Undergraduate Science Journal.

Joyce, B., y Deng, J. (s.f.). *Sentiment analysis of tweets for the 2016 us presidential election*. IEEE.

Kaser, O., y Lemire, D. (2007). Tag-cloud drawing: Algorithms for cloud visualization.

Lee, B., Riche, N. H., Karlson, A. K., y Carpendale, S. (2010). Sparkclouds: Visualizing trends in tag clouds.

Murrell, P. (s.f.). *An introduction to r graphics*. Auckman University.

Murtagh, F., y Legendre, P. (s.f.). *Ward's hierarchical clustering method: Clustering criterion and agglomerative algorithm*. Science Foundation Ireland.

Pang, B., y Lee, L. (2002). Thumbs up? sentiment classification using machine learning techniques.

R., G. (s.f.). *Métodos jerárquicos de análisis de cluster*. Universidad de Granada.

Ramteke, J., Shah, S., Godhia, D., y Shaikh, A. (s.f.). *Election result prediction using twitter sentiment analysis*. IEEE.

Rickert, J. (s.f.). *What is the tidyverse*. RViews.

Tatman, R. (s.f.). *Nlp in r: Topic modelling*. Kaggle.

Yu, H., y Hatzivassiloglou, V. (s.f.). Towards answering opinion questions: Separating facts from opinions and identifying the polarity of opinion sentences.

Łukasz Augustyniak, Szymański, P., Kajdanowicz, T., y Tuligłowicz, W. (2016). Comprehensive study on lexicon-based ensemble classification sentiment analysis.