Universidad Central de Venezuela.
Facultad de Ciencias.
Escuela de Matemática.

Herramientas matemáticas para la minería de texto.

Trabajo presentado ante la **Ilustre Universidad Central de Venezuela**,
por el *Br. Leonardo Toglia*

Caracas, 1 de noviembre de 2016

Nosotros, los abajo firmantes, designados por la Universidad Central de Venezuela como integrantes del Jurado Examinador del Trabajo Especial de Grado titulado "**Herramientas matemáticas para la minería de texto.**", presentado por el **Br. Leonardo Toglia**, titular de la Cédula de Identidad **22671778**, certificamos que este trabajo cumple con los requisitos exigidos por nuestra Magna Casa de Estudios para optar al título de **Licenciado en Matemática**.

Dra. Carenne Ludeña
Tutor

Dra. Mairene Colina
Jurado

MSc. Jesús Lares
Jurado

Índice de Contenido

Agradecimientos

Primeramente le agradezco a Dios por haberme acompañado y guiado durante toda mi carrera.

A mis padres Leonardo y Alba por haberme apoyado en todo momento, por los valores que me han inculcado y por cada uno de sus consejos los cuales me han ayudado para alcanzar cada una de mis metas, a mis hermanos Michele y Giovanna por siempre apoyarme.

Quisiera agradecer enormemente a mi tutora la Dra Carenne Ludeña por su dedicación y apoyo incondicional durante el desarrollo del presente trabajo y además por compartir sus valiosos conocimientos en esta área de la minería de texto. También quiero agradecer a los profesores Jesús Lares y José Sosa por las catedras electivas dictadas referente al área, las cuales fueron fundamentales para la realización de este trabajo, en especial al profesor Jesús Lares por todos sus consejos y comentarios durante la fase final de la redacción y presentación de este trabajo.

Al profesor Expedito Cedeño por guiarme durante la carrera con sus sabios consejos, guías y opiniones y por haberse convertido en gran amigo cuya amistad aprecio mucho.

A mi amigo y compañero de estudio Wilmer González, con el cual comparti largas pero divertidas horas de estudio y trabajo en esta área de estudio.

A la Profesora Mairene Colina por sus oportunas y muy impotantes correciones y sugerencias con la redacción, presentación y defensa del presente trabajo

A cada uno de los profesores que me dieron clases durante la licenciatura tanto a los que me dieron clases previamente en el Liceo José Manuel Nuñez Ponte.

Capítulo 1

Antecedentes y motivación.

1.1 Antecedentes y motivación.

En los últimos 5 años se ha generado más información científica que en toda la historia de la humanidad. Por su parte, los datos generados diariamente pueden ser muy diversos y además se estima que todos los días se crean 2.5 trillones de bytes en datos, de hecho el 90% de los datos del mundo se han generado en los últimos años.

En ese gran volumen de datos generados predomina el **texto** el cual hace referencia a información muy variada. El área de la **minería de textos** ha surgido haciendo uso de herramientas matemáticas y computacionales para encontrar información y relaciones vitales detrás de estos grandes volúmenes de textos.

En la década de los 90, **Dan Sullivan** fue uno de los primeros investigadores en dirigir monografias exclusivamente al tema de minería de textos. En una de ellas define el concepto de minería de textos como "Es cualquier operación realizada para extraer textos procedentes de distintas fuentes externas con el objetivo de obtener inteligencia". Posteriormente **Sullivan** define la minería de textos como "el descubrimiento de información y conocimiento que anteriormente no se conocía a partir de corpus textuales".

La segunda definición que propuso **Sullivan** coincide con la más popular la cual fue formulada por **Marti Hearst** en el articulo de "Untagling text data mining", **Hearst** enfatiza que esta tiene como objetivo descubrir información y conocimiento que previamente se desconocía y que no aparecía en ninguno de los documentos analizados.

Según **Hearst** la minería de textos es un proceso con el que se pretende descubrir nueva información o conocimiento. Posteriormente **Sullivan** señala como minería de textos al proceso de compilar, organizar y analizar grandes colecciones de documentos.

En la actualidad, debido a la emergencia de no saber como manejar grandes volúmenes de datos nace el término Big data el cual marca el inicio a una nueva ciencia: La ciencia de datos. Esta se encarga del manejo grandes volúmenes de información, haciendo uso de las matemáticas y nuevas herramientas computacionales las cuales pueden ser versiones de software libres abiertas a todo

público como lo son Hadoop, R, Python entre otras y versiones comerciales como SAS, Pentaho , Cloudera, MapR entre otros, todas creadas con la finalidad de resolver diversos problemas.

La minería de textos es aplicada para la solución de problemas en áreas muy diversas, empezando por la rama de biomedicina la cual hace uso de diversos algoritmos de minería de textos para procesar datos de biología molecular con la finalidad de reconocer entidades como lo pueden ser proteínas ó genes y posteriormente genera términos para una ontología de genes.

En el área de las ciencias sociales la minería de textos es muy utilizada en el análisis de redes sociales, análisis de sentimientos entre otros. Los medios de comunicación hacen uso de diversos algoritmos para la elaboración de resúmenes de grandes corpus de textos, campañas de marketing basándose en modelos predictivos para el descubrimiento de palabras claves y además de algoritmos de clusterización para identificar la segmentación de clientes.

Por su parte entes gubernamentales encargados en seguridad nacional, hacen uso de la minería de textos para el análisis en las redes sociales para la prevención de ataques terroristas; además no sólo entes en el área de seguridad sino otros entes hacen uso en la toma de decisiones en las administraciones públicas extrayendo conocimientos en redes sociales de participación ciudadana.

1.2 Objetivo general del trabajo.

1. Desarrollar una metodología para la implementación de métodos matemáticos en el análisis de textos.

1.3 Objetivos específicos del trabajo.

1. Definir cada una de las etapas de la metodología.
2. Escoger las herramientas matemáticas para la metodología.
3. Definir caso de estudio para la metodología.
4. Desarrolllar una herramienta sobre el caso de estudio.

Capítulo 2

Metodología con marco aplicativo

El objetivo de este trabajo es proponer una metodología para la implementación de herramientas matemáticas para el análisis de textos. Esta metodología cuenta con las siguientes etapas:

- ***Definición del caso de estudio:*** Plantear que problema se quiere solucionar.
- ***Preprocesamiento:*** Convertir los documentos en textos tratables para poder llevar a cabo los distintos análisis.
- ***Selección de términos informativos:*** Identificar cuales términos son mas frecuentes ó aportan información.
- ***Proceso de clusterización:*** Determinar los grupos que se forman mediante diferentes técnicas de clusterización.
- ***Proceso de clasificación:*** Implementar clasificadores para poder etiquetar los documentos según una clase.
- ***Implementacioón y análisis de resultados:*** Desarrollar aplicaciones haciendo usos de las herramientas estudiadas y analizar los distintos resultados.

En los próximos capítulos se estudiaran cada una de las etapas además de la implementación de uso de la misma.

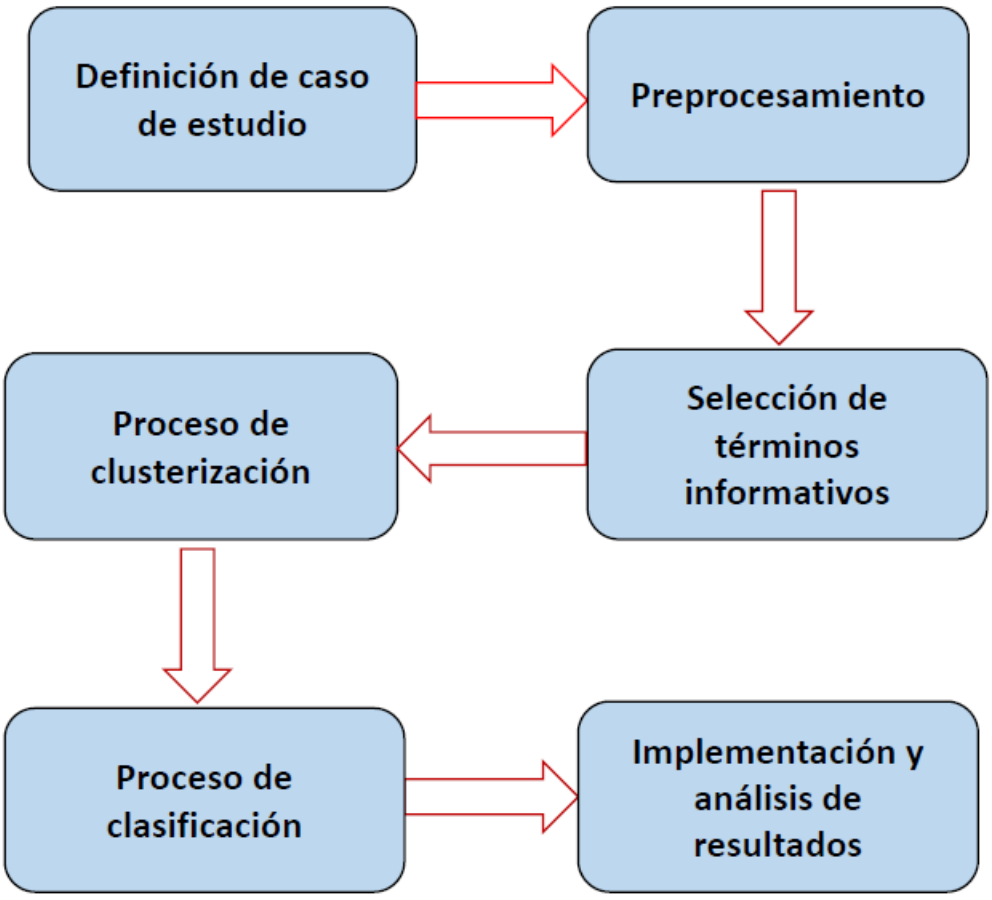


Imagen 2.1: Metodología propuesta.

Capítulo 3

De documentos a textos tratables

Para poder llevar a cabo los diferentes tipos de análisis que permitiran extraer información a las unidades de texto, es necesario el preprocesamiento de los mismos para obtener textos tratables los cuales ya pueden ser procesados por los diferentes métodos de minería de texto para la realización de los distintos tipos de análisis.

3.1 Descriptores de un documento

Cualquier tipo de texto esta compuesto por un conjunto de descriptores los cuales le dan el esquema estándar por el cual está creado cualquier texto.

Los 4 tipos comunmente mas usados son:

1. **Carácteres**: Unidad más básica en general, no permite un análisis muy avanzado, pero puede contener información valiosa (carácteres especiales) muy importante para el preprocesamiento. Por ejemplo el uso de carácteres especiales para separar palabras.

2. **Palabras**: Unidades básicas con contenido (reunión de carácteres), puede ó no incluir palabras complejas separadas por un guión por ejemplo: clave-valor.

3. **Términos**: Colección de palabras con significados precisos por ejemplo: Fuerzas Armadas.

4. **Conceptos**: Información no necesariamente presente o descrita en el documento. Se obtienen de diferentes maneras, métodos estádisticos, métodos estadísticos. Muchos de estos métodos de caracterización usan referencias cruzadas a información externa, por ejemplo: Ontologías e información relacionada con jerarquías.

3.2 Concepto de corpus y su definición

En la lingüística computacional se puede llamar *corpus* a cualquier colección que contenga mas de un texto. Por otra parte en el ámbito de la comunidad científica existe la idea de que el *corpus* no solo ofrece información sobre sí mismo, es decir sobre lo que contiene sino lo que "representa" una sección mas amplia de la lengua seleccionada de acuerdo a una tipología especifica. La noción de "representatividad " aparece en otras definiciones, por ejemplo en [26] un *corpus* se define como una colección de textos escogidos para caracterizar un estado o una variedad de una lengua.

La definición que ofrece [2] añade otro aspecto esencial en la creación de un *corpus*: este debe ser construido de acuerdo a una serie de criterios explícitos; siguiendo la definición de [19] se puede definir un corpus como un conjunto de textos de lenguaje natural, almacenados en un formato electrónico homogéneo, seleccionados y ordenados de acuerdo a criterios explícitos para ser utilizados como módelo de un estado o nivel de lengua determinado, en estudios o aplicaciones relacionados, en mayor o menor medida con el análisis lingüístico.

3.3 Preprocesamiento

Para poder descubrir conocimientos en un conjunto de textos se debe pasar por la etapa más importante llamada **preprocesamiento**, la cual le da al conjunto de textos una forma que le permite ser tratada computacionalmente para aplicar alguna técnica de minería de textos.

El preprocesamiento de los textos cuenta con las siguientes etapas:

1. **Extandarización de los documentos**: Los textos la mayoría de las veces tienen diversos formatos de origen como por ejemplo ASCII, XML entre otros, por lo que todos los documentos a tratar deben estar en el mismo formato.
2. **Tokenización**: La tokenización es la forma de separar el textos en palabras comúnmente llamadas tokens. Este proceso toma en cuenta que las palabras pueden estar interrumpidas por un final de linea, identificar otro carácter para la separación de palabras como lo puede ser un guión ó identificar si las palabras están asociadas a signos de puntuación.
3. **Lexematización (radicalización)**: La lexematización es una parte del procesamiento linguístico que trata de determinar el lema de cada palabra que surge en los documentos. Las palabras son reducidas de género, número, adjetivos y tiempos verbales a su raíz. Las raices se utilizan como términos de indexación en lugar de utilizar palabras, Como un ejemplo de lo explicado anteriormente: estudiante, estudiar, estudiamos tienen como raíz ***studi***.
4. **Sinónimos o significación especial**: Consiste en agrupar las palabras con igual significado y etiquetarlas con una palabra que represente al conjunto. Un ejemplo de uso de esta etapa es el siguiente, las palabras bonita, hermosa, linda pueden ser todas etiquetadas con una misma palabra por ejemplo hermosa.
5. **Remoción de palabras no informativas (stop words)**: Son los términos que se han generalizado y son abundantes en cualquier tipo de texto y no son de carácter informativo acerca del contenido del mismo, por ejemplo artículos, pronombres, preposiciones son los comunes. Se excluyen las palabras vacías para eliminar los términos que no auxilian a generar conocimiento del texto.

Una vez terminada la etapa de preprocesamiento obtenemos lo que se conoce como un conjunto tratable, a partir del conjunto tratable se genera un nuevo concepto el cual es la base para la realización de los diferentes análisis en la minería de textos, este nuevo concepto es conocido como la

matriz de términos-documentos.

La matriz de términos-documentos describe la frecuencia de los términos que ocurren en la colección de documentos y esta formada de la siguiente manera:

Las filas corresponden a los términos y las columnas a los documentos de la colección:

$$\mathbf{m[i,j]} = \begin{cases} 0 & \text{si d(i) no contiene a p(j).} \\ 1 & \text{si lo contiene.} \end{cases}$$

Donde d(i) se refiere al documento "i" y p(j) al término "j".

Otra forma de representar a las entradas de la matriz de términos documentos es la siguiente:

$$m[i, j] = f_{ij}.$$

Donde f_{ij} es la frecuencia de la palabra j en el documento i.

Cabe destacar que también puede hacerse el análisis dual usando la matriz transpuesta obteniendo así una matriz documentos-términos. Un ejemplo de la matriz anterior es el siguiente:

	Docs																			
Terms	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
data	1	1	0	0	2	0	0	0	0	0	1	2	1	1	1	0	1	0	0	0
examples	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
introduction	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
mining	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0
network	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1
package	0	0	0	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

3.4 Selección de términos informativos

Una vez obtenida la representación matricial del conjunto de documentos el siguiente paso para el análisis de los documentos es la selección de términos informativos.

Para ello existen diferentes métodos los cuales serán explicados a continuación.

3.4.1 Probabilidad de una palabra

La probabilidad de ocurrencia de una palabra es la forma más simple de utilizar la frecuencia como indicador de importancia. La probabilidad de una palabra w usando como notación p(w) es calculada como el número de ocurrencias de la palabra w denotada por c(w), dividida por el

número de palabras en el documento denotada por N:

$$p(w) = \frac{c(w)}{N}.$$

Es conocida como la manera más simple, ya que solo consta de observar cuántas veces ocurre la palabra w en un documento.

3.4.2 TFIDF

Este peso es comúnmente usado en las áreas de recuperación de información y la minería de texto. Este peso es una medida estadística usada para evaluar cuan importante es una palabra en un documento, una colección de documentos o un corpus. La importancia crece proporcionalmente según el número de veces que aparece una palabra en el documento pero fuera de la frecuencia de la palabra en el corpus. Típicamente el peso TFIDF esta compuesto por dos términos:

TF (frecuencia del término)

Es una medida que calcula con que frecuencia un término aparece en un documento, sí cada documento es de diferente dimensiones es posible que un término aparezca mas veces en documentos largos que en documentos cortos, si es así la frecuencia del término es usualmente dividida por la longitud del documento, donde la longitud de un documento es el número de términos en el documento, se podría interpretar como un tipo de normalización. Por lo que se obtiene lo siguiente:

$$TF(t) = \frac{\#\ Veces\ que\ el\ \text{término}\ t\ aparece\ en\ un\ documento.}{\#\ Total\ de\ \text{términos}\ en\ el\ documento.}$$

IDF (inverso de frecuencia de documento)

El factor IDF de un término es inversamente proporcional al número de documentos en los que aparece dicho término. Esto significa que cuanto menor sea la cantidad de documentos, así como la frecuencia absoluta de aparación del término, mayor será su factor IDF y a la inversa, cuanto mayor sea la frecuencia absoluta relativa a una alta presencia en todos los documentos de la colección, menor será su factor discriminatorio:

$$IDF(t) = \log_e \frac{\text{Número total de documentos.}}{\text{Número de veces en que el término t aparece en el documento.}}$$

Así de esta manera el peso **TF-IDF** es el siguiente:

$$TFIDF = TF * IDF$$

$$\underbrace{TFIDF(t,d)}_{\text{Peso de un término t en un documento d}} = \underbrace{TF(t,d)}_{\text{frecuencia de aparición de un término t en un documento d}} * \underbrace{IDF(t,d)}_{\text{Factor de idf de un término t}} \tag{3.1}$$

3.4.3 Latent Semantic Indexing (LSI)

LSI es un método utilizado para descubrir conceptos ocultos en un conjunto de documentos. Cada documento y término (palabras) es entonces representado como un vector con elementos correspondientes con esos conceptos.

El propósito es ser capaces de representar los documentos y términos en una forma unificada para exponer similitudes o relaciones semánticas dadas las representaciones matriciales obtenidas a partir de la matriz de términos-documentos:

1. Matriz término-documento.
2. Matriz documento-documento.
3. Matriz término-término.

Denotándolas como A, B y C respectivamente. Para la realización de este algoritmo es necesario usar la descomposición en valores singulares de las matrices enunciadas previamente. Para ello se resumirán resultados de álgebra lineal, los cuales seran útiles para la comprensión del método de LSI.

Resumen de resultados del álgebra lineal

Sea C_{MxN} una matriz con entradas en los reales. El rango de una matriz es el número de filas linealmente independientes así

$$Rango(C) \leq min\{M, N\}. \tag{3.2}$$

Una matriz cuadrada donde todas las entradas fuera de la diagonal son nulas, es llamada matriz diagonal, donde el rango es el número de entradas no nulas en la diagonal. Dada una matriz C_{MxM} cuadrada y los autovalores de C son las soluciones de la ecuación:

$$det((C - I_{MxM})\vec{x}) = 0.$$

Dado un autovalor λ, un autovector asociado a λ es un vector $\vec{x}$ de R^M que satisface la ecuación $(C - \lambda I)\ \vec{x} = 0$. El autovector asociado al mayor autovalor (en valor absoluto) es llamado el autovector principal.

Descomposición de una matriz

Teorema 3.1 *Sea S_{MxM} una matriz cuadrada con entradas en los reales con M autovalores linealmente independientes entonces existe una descomposición:*

$$S = U\Delta U^{-1}, \tag{3.3}$$

donde las columnas de U son los autovectores de S y Δ es una matriz diagonal cuyas entradas son los autovalores de S en orden decreciente:

$$\Delta = \begin{pmatrix} \lambda_1 & & & & \\ & \lambda_2 & & & \\ & & \ddots & & \\ & & & \lambda_{m-1} & \\ & & & & \lambda_m \end{pmatrix} \tag{3.4}$$

Sí los autovalores son distintos la descomposición es única.

Teorema 3.2 *Sea S_{MxM} una matriz cuadrada con entradas en los reales con M autovalores linealmente independientes entonces existe una descomposición diagonal simétrica:*

$$S = Q\Delta Q^T. \tag{3.5}$$

Donde las columnas de Q son los autovectores ortonormalizados y Δ es la matriz diagonal cuyas entradas los autovalores de S. Además como todas las entradas de Q pertenecen a los reales se obtiene que

$$Q^{-1} = Q^T. \tag{3.6}$$

Descomposición en valores singulares

Las descomposiciones anteriormente enunciadas son aplicadas a matrices cuadradas, sin embargo la matriz a estudiar (término-documento) no generalmente es cuadrada, en un caso muy excepcional lo seria.

Una extensión del Teorema 2 conocida como descomposición en valores singulares es usada para los casos donde la matriz no sea cuadrada.

Sea C_{MxN} una matriz no cuadrada, sea U_{MxM} una matriz cuyas columnas son los autovectores ortonormalizados de CC^T y sea V_{NxN} una matriz donde las columnas son los autovectores ortogonalizados de C^TC.

Teorema 3.3 *Sea r el rango de la matriz C_{MxN}, entonces la descomposición en valores singulares de C (SVD) es de la forma:*

$$C = U \sum U^T \tag{3.7}$$

donde:

1. Los autovalores $\lambda_1 \cdots \lambda_r$ de CC^T son los mismos de C^TC. Como CC^T es simétrica sus autovalores son reales y no negativos.

2. Para $1 \leq i \leq r$, sea $\theta_i = \sqrt{\lambda_i}$, con $\lambda_i \leq \lambda_{i+1}$, entonces la matriz $\sum_{MxN}$ es igual a $\sum_{ii}$ para $1 \leq i \leq r$ y cero para las demás entradas. Los valores θ_i son conocidos como los valores singulares de C.

A continuación describimos como emplea el método LSI la descomposición en valores singulares a partir de la matriz de términos-documentos:

Sea A_{MxN} la matriz término-documento de una colección de documentos. Cada columna de A corresponde a un documento. Si el término **i** ocurre a veces en el documento **j** entonces $A[i, j] = a$.

Las dimensiones de A , m y n corresponden al número de palabras y documentos en la colección. Cabe destacar que $B = A^T A$ es la matriz documento-documento, si el documento **i** y **j** tienen b palabras en común entonces $B[i, j] = b$, por otra parte, $C = AA^T$ es la matriz término-término si el término i y j ocurren juntos en c documentos entonces $C[i, j] = c$; evidentemente ambas matrices B y C son cuadradas y simétricas y además B esMxMy C es NxN.
La descomposición en valores singulares de A se realiza usando las matrices B y C generando la siguiente descomposición de A:

$$A = S \sum U^T. \tag{3.8}$$

Donde S es la matriz de autovectores de B, U es la matriz de autovectores de C y $\sum$ es la matriz diagonal de los valores singulares obtenidos como la raíz cuadrada de los autovalores de B. Algunas veces los valores singulares son "muy pequeños para considerase un valor singular" (esta elección se realiza empíricamente).

En LSI se ignoran estos valores singulares y se reemplazan por "0", así solo dejando los "k " valores singulares de $\sum$; entonces $\sum$ puede ser reducida hasta las k-primeras entradas a lo largo de la diagonal, así reduciendose a la matriz $\sum_k$, donde $\sum_k$ es una matriz que solamente contiene los k-valores singulares que se tomaron, Por su parte las matrices S y U también se reducen en S_k y U_k^T para obtener las k columnas y filas respectivamente.
De esta manera la matriz A es ahora aproximada por :

$$A_k = S_k \sum_k U_k^T \tag{3.9}$$

Tomando en cuenta que S_k, $\sum_k$ y U_k^T son matrices de dimensiones Mxk, kxk, kxN respectivamente y su producto es de nuevo una matriz de dimensión MxN.
Intuitivamente los k autovectores en S y U corresponden a los k "conceptos escondidos " donde términos y documentos participan.
Los términos y documentos tienen ahora una nueva representación en términos de estos conceptos escondidos. Por su parte los términos son representados por el vector filas de la matriz Mxk:

$$S_k \sum_k . \tag{3.10}$$

Además los documentos por los vectores columnas en la matriz kxN

$$\sum_k U_k^T . \tag{3.11}$$

Vector de consultas

Un vector de consultas $\vec{q}$ es un vector donde $q_i = 1$ si el término de la consulta ocurre en el corpus y $q_i = 0$ si no ocurre una coincidencia. Una vez realizado el proceso de LSI el vector de consulta va a tener una nueva representación dada por:

$$q = \vec{q}^T S_k (\sum)_k^{-1}. \tag{3.12}$$

Donde la última reprsentación es la del vector $\vec{q}$ en el espacio de dimensión menor. Por su parte para ordenar la importancia de los vectores consultas en los documentos se hace uso de la función

de similitud del coseno, calculando la similitud de un vector consulta con cada documento en el corpus:

$$Sim(\vec{q}, d_i) = \frac{\vec{q} \cdot d}{|\vec{q}||d_i|}. \tag{3.13}$$

posterior a este cálculo se ordena en orden decreciente las similitudes donde para los valores más altos, el vector consulta tiene más importancia en esos documentos.

3.5 Análisis de componentes principales (ACP)

3.5.1 Resumen de resultados de la estadística

Media: Es una medida de tendencia central que resulta de efectuar una serie determinada de operaciones con un conjunto de datos númericos y que en determinadas condiciones, puede representar por sí solo a todo el conjunto. La media se define como la suma de todos los valores observados, dividido por el número total de observaciones.

$$\bar{x} = \frac{\sum_{i=1}^{N} x_i}{N}. \tag{3.14}$$

Desviación estándar: Es una medida de dispersión que se obtiene como la distancia promedio desde la media del conjunto de datos a otro punto.

$$S = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \bar{x})}{n-1}}. \tag{3.15}$$

Varianza: Es una medida de dispersión de la información, se obtiene como el cuadrado de la desviación estándar.

$$S^2 = \frac{\sum_{i=1}^{N}(x_i - \bar{x})}{n-1}. \tag{3.16}$$

Covarianza: Es una medida de la fuerza de la relación lineal entre dos variables cuantitativas y se calcula como (caso bi-dimensional):

$$Cov(x, y) = \frac{\sum_{i=1}^{N}(x_i - \bar{x}) * (x_i - \bar{x})}{n-1}. \tag{3.17}$$

Una manera útil de obtener todos los posibles valores de covarianza entre todas las diferentes dimensiones es colocarlas en un arreglo matricial, dicho arreglo es conocido como **matriz de covarianza:**

$$C^{NxN} = (c_{i,j}, c_{i,j} = cov(dim_i, dim_j)). \tag{3.18}$$

donde C^{NxN} es una matriz con N filas y N columnas.

El signo del valor de covarianza se analiza de la siguiente manera:

1. Sí es un valor positivo la covarianza indica que hay dependencia directa (positiva), es decir, a grandes valores de x corresponden grandes valores de y.

2. Sí el signo es negativo la covarianza indica que hay dependencia inversa o negativa, es decir, a grandes valores de x corresponden pequeños valores de y.

3. Sí es 0 interpreta como la no existencia de una relación lineal entre las dos variables estudiadas.

Los resultados del resumen de álgebra lineal también serán utilizados en la explicación del algoritmo de ACP.

3.5.2 Análisis de componentes principales (ACP)

ACP es un método de reducción de dimensionalidad en el cual el análisis de covarianza entre factores toma importancia. El conjunto de datos original es mapeado a un nuevo sistema de coordenadas basado en la varianza de la data. ACP aplica métodos matemáticos para transformar un número de posibles variables correlacionadas en un número mas pequeño de variables no correlacionadas llamadas componentes principales.

La primera componente representa la mayor cantidad de variabilidad en el conjunto de datos como sea posible y cada una de las componentes siguientes representa la mayor cantidad de variabilidad restante como sea posible.

ACP es útil cuando es aplicado a largos números de variables donde probablemente hay algo de redundancia en esas variables en este caso la redundancia significa que algunas de las variables estan correlacionadas con otras. ACP es recomendado como una herramienta exploratoría para descubrir tendencias desconocidas en la data. Esta técnica ha sido aplicada en los campos de compresión de imagenes, reconocimientos de rostros y es comúnmente utilizado para encontrar patrones en datas de grandes dimensiones.

3.5.3 Algoritmo para ACP

El algoritmo para encontrar las componentes principales es el siguiente:

1. Para que ACP trabaje apropiadamente se debe normalizar el conjunto de datos, restando la media para cada dimensión de la data. La media sustraída es el average a lo largo de cada dimensión, es decir a cada x_i se le es restado $\bar{x_i}$ por lo que genera un conjunto de datos de media cero.

2. Calcular la matriz de covarianza generada en el paso anterior; se explicó anteriormente que la matriz de términos-documentos no es una matriz cuadradada por lo que se calcula la matriz de covarianza a las matrices $B = AA^T$ y $C = A^TA$ donde A es la matriz de términos-documentos.

3. Una vez obtenida la matriz de covarianza, se procede a calcular los autovalores y autovectores de la matriz.

4. Del paso anterior, el autovector asociado al autovalor de mayor magnitud es llamado **componente principal** del conjunto de datos. En general una vez encontrados los autovalores y sus respectivos autovectores se ordenan de forma decreciente generando así todas las componentes con un orden de significancia, donde se descartan las componentes asociadas a los autovalores de magnitudes pequeñas.

5. Se construye el vector de características colocando los autovectores como columnas de una matriz.

$$\text{Vector de características} = (eig_1 eig_2 .....eig_n). \tag{3.19}$$

6. Una vez escogidas las componentes (autovectores) se quiere obtener el nuevo conjunto de datos, para ello, se calcula la transpuesta al vector de característica, y a se multiplica por la transpuesta de la matriz obtenida después del primer paso, la cual sera denotada como llamada Data Ajustada así:

$$DF = \text{Vector de características}^T * DataAjustada^T. \tag{3.20}$$

   donde Vector de características^T tiene como filas los autovectores, donde los más significantes estan en las primeras filas, y cada columna de la matriz $DataAjustada^T$ tiene una dimensión diferente.

En conclusión el análisis de componentes princiaples es una técnica de reducción de dimensionalidad que describe la información de un conjunto de variables observadas mediante un conjunto de variables mas pequeño (componentes principales) las cuales son combinaciones lineales de las variables de partida, permitiendo el análisis de las características que definen a cada elemento en la muestra inicial en función de las componentes de máxima varianza.

3.5.4 Relación entre ACP y SVD

Los algoritmos anteriormente explicados estan estrechamente relacionados, definamos una matriz X de orden mxn y ahora definamos una matriz Y_{nxm} de la siguiente manera:

$$Y = \frac{1}{\sqrt{n}} x^T. \tag{3.21}$$

donde cada columna de Y poseé media cero, ahora multiplicando por Y^T obtenemos:

$$Y^T Y = (\frac{1}{\sqrt{n}} x^T)^T (\frac{1}{\sqrt{n}} x^T).$$

$$Y^T Y = \frac{1}{n} X X^T.$$

$$Y^T Y = C_X.$$

Por construcción $Y^T Y$ es igual a la matriz de covarianzas de X.

Si calculamos la descomposición en valores singulares de Y obtenemos:

$$Y = U \sum V^T. \tag{3.22}$$

donde las columnas de V son las componentes principales X. Esto significa que V se genera por el espacio filas de $Y = \frac{1}{\sqrt{n}} X^T$ por lo tanto V también debe generar el espacio columna de $Y = \frac{1}{\sqrt{n}} X$ por lo que se puede concluir que la búsqueda de las componentes principales equivale a la búsqueda de de una base ortonormal que genere el espacio columna de X.

3.6 Logaritmo del cociente de verosimilitudes

Este método estadístico es aplicado para la identificación de palabras que sean altamente descriptivas en los documentos. Esta palabras que describen a los documentos tradicionalmente son llamadas "Topic signatures "(T-S). El uso de estas palabras como representación de los documentos genera un buen balance en la selección de contenido importante dentro de los mismos. Los T-S son palabras que ocurren comunmente en unos documentos y raras veces en otros. La decisión para asumir si una palabra es descriptiva o no es basada en una prueba de hipótesis.
La prueba de hipótesis sobre el conjunto de documentos provee una manera de determinar la frecuencia de la ocurrencia de palabras en el corpus. Para calcular el logaritmo del cociente de verosimilitudes, se hará la prueba para un bi-grama el cual es la combinación de dos térmninos.
Sean w_1 y w_2 , por su parte:

$$H_1 : P(w_2|w_1) = p = P(w_2|\urcorner w_1).$$

$$H_2 : P(w_2|w_1) = p_1 \neq p_2 = P(w_2|\urcorner w_1).$$

La hipótesis 1 es una formalización de independencia (La ocurrencia de w_2 es independiente de la previa ocurrencia de w_1) y por su parte la hipótesis 2 es una formalización de dependencia.

La ocurrencia de cada palabra es un ensayo de Bernoulli con probabilidad p de suceder, el cual ocurre cuando $w_i = w$. La probabilidad de observar k repeticiones de la palabra w en N ensayos está dada por la distribución Binomial:

$$b(k, N, p) = \binom{N}{k} p^k (1-p)^{N-k}. \tag{3.23}$$

Por lo que el logaritmo del cociente de verosimilitudes es definido a partir de la ecuación anterior de la siguiente manera:

$$\lambda = \frac{b(k, N, p)}{b(k_{w_1}, N_{w_1}, p_{w_1}) * b(k_{\urcorner w_1}, N_{\urcorner w_1}, p_{\urcorner w_1})}. \tag{3.24}$$

Pero haciendo uso del estadístico $-2log\lambda$ este es asintoticamente una distribución chi-cuadrado (χ^2). Esta distribución asintótica se usa para llevar a cabo la prueba de hipótesis.

Capítulo 4

Clusterización de documentos.

4.1 Aspectos generales

El proceso de clusterización es aquel que agrupa un conjunto de datos en grupos (clusters) basándose solamente en la información que describe al conjunto para identificar las similitudes (o relaciones) si pertenecen al mismo grupo ó las diferencias con respecto a los objetos de otros grupos. términos como segmentación y particionamiento son usados como sinónimos de clusterización.

Entre las técnicas de clusterización existen diferentes tipos, predominando la clusterizacíon jerárquica y la particional (no supervisado) donde el método mas común en la técnica particional es el algoritmo *K-medias* ó una variación de la anterior basando el el algoritmo de componentes principales. A continuación describiremos los algoritmos mas usados en el área y además su implementación en la minería de textos para la clusterización de documentos.

4.2 El módelo de espacio vectorial y la clusterización de documentos.

Para los algoritmos utilizados en la clusterización, los documentos son representados usando el método de espacio vectorial. En este módelo cada documento "d" es considerado como un vector **d**, en esta simple forma cada documento es representado por el vector de las frecuencias de los términos (**TF**):

$$d_{TF} = (t_{f_1}, t_{f_2}, \ldots, t_{f_n}) \tag{4.1}$$

donde cada t_{f_i} es la frecuencia del término i^{th} en el documento. Cabe destacar que para la representación del documento se asume que ya pasó por la etapa de preprocesamiento estudiada en el capítulo anterior.

Típicamente se usa la versión de este módelo en el cual los pesos de cada término están basados en el peso **IDF** estudiado en el capítulo anterior.

4.3 Medidas de distancia o similitud

Las medidas de similitud o distancia son expresiones matemáticas que permiten resumir en un número, el grado de relación entre dos entidades, sobre la base de semejanza ó la desigualdad

entre la cualidad o la cantidad de sus atributos, ó ambas. Cómo medidas de similitud o distancia existen muchas opciones para la clusterización de documentos como la distancia euclídea, el indice de jaccard, pero la más usada es la medida de similaridad del coseno.

4.3.1 Distancia Euclídea

La distancia euclídea es una métrica estándard para problemas geométricos, es la distancia ordinaria entre dos puntos en el espacio y su cálculo es muy simple. La distancia euclídea es ampliamente utilizada en las técnicas de clusterización incluyendo la clusterización de documentos, además es la distancia por defecto en el algoritmo de K-medias.

Dados dos documentos d_1 y d_2 representados vectorialmente usando la función de peso **ITF** obteniendo $d_{1,i}$, $d_{2,i}$ respectivamente. La distancia euclídea entre dos documentos viene dada por:

$$D(d_1, d_2) = \sqrt{\sum_{i=0}^{n} |d_{1,i} - d_{2,i}|^2}. \tag{4.2}$$

Considerando conjunto de la unión de todos los términos que aparecen en cada documento.

4.3.2 Similaridad del Coseno

Cuando los documentos son representados como vectores, la similaridad entre dos documentos corresponde a la correlación entre los vectores, esto es cuantificado como el coseno del ángulo entre vectores.

Por su parte esta medida es la más utilizada en la minería de texto y viene definida de la siguiente manera:

$$Cos(d_1, d_2) = \frac{d_1 \cdot d_2}{\|d_1\| \cdot \|d_2\|}. \tag{4.3}$$

donde es normalizada con la distacia euclídea ($\|\cdot\|_2$).

4.3.3 Coeficiente de Jaccard

El coeficiente de Jaccard, mide la similaridad como la intersección dividida entre la unión de los objetos. En las técnicas de minería de texto, el coeficiente de Jaccard compara la suma de los pesos de los términos comunes y la suma de los términos que están presentes en cualquiera de los dos documentos pero no son los términos comunes.
La definición formal es:

$$J(d_1, d_2) = \frac{d_1 \cdot d_2}{|d_1|^2 + |d_2|^2 - d_1 \cdot d_2}. \tag{4.4}$$

Donde $d_1 \cdot d_2$ es el producto escalar entre dos vectores cuyas entradas son ceros y unos, por lo que esta operación cuenta el número de términos comunes entre ambos.

El coeficiente de Jaccard es una medida de similaridad con rango entre 0 y 1, Es 1 cuando $d_1 = d_2$ y 0 cuando d_1 y d_2 son disjuntos, interpretando lo anterior 1 significa que los documentos son el mismo y 0 significa que son completamente diferentes.

La correspondiente medida de distancia viene dada por:

$$D_J = 1 - J(d_1, d_2). \tag{4.5}$$

4.4 Clusterización jerárquica

Este algoritmo consiste en la construcción de un árbol mediante un proceso de adición de clusters. Para el proceso de análisis de **K** muestras se asume como condición inicial la presencia de K-clusters, conteniendo cada uno de ellos una única muestra mientras que en cada paso ejecutado permite la adición de los dos clusters más cercanos en un solo cluster. Repitiendo **K** veces esta operación se consigue obtener un único gran cluster formado por **K** muestras.

A posteriori hay que analizar el árbol y elegir un nivel de agregación satisfactorio, las formas más comunes son las siguientes tomando una distancia máxima entre clusters ó la elección de un número mínimo de clusters.

Este tipo de clusterización consta de varias etapas, el agrupamiento jerárquico ascendente da inicio separando objeto en un cluster por sí mismo, mientras se desarrolla cada etapa del análisis, el criterio por el cual los objetos son separados trata de identificar la manera de enlazar los dos clusters más similares hasta que todos los objetos sean agrupados en un árbol de jerárquia completo.
La pauta más básica para cualquier agrupación es la distancia. Los objetos que están cerca uno del otro perteneceran al mismo cluster y los objetos que están lejos uno del otro perteneran a distintos clusters.

Realizar el cálculo de la proximidad entre dos clusters es lo que diferencia entre sí a las técnicas de agrupamiento jerárquico. Existen diversas técnicas de agrupamiento aglomerativas como lo son:

1. **Single linkage:** Define la proximidad del grupo como la proximidad entre los dos puntos más cercanos que estan en diferentes grupos. Sea S_i el grupo i-ésimo y x_i un elemento de grupo i-ésimo, podemos definir al single linkage como:

$$x_i \in S_i, x_j \in S_j; \min d(x_i, x_j). \tag{4.6}$$

2. **Complete linkage:** Define la proximidad del grupo como la proximidad entre los puntos más lejanos pertenecientes a diferentes grupos. Sea S_i el grupo i-ésimo y x_i un elemento de grupo i-ésimo, podemos definir al complete linkage como:

$$x_i \in S_i, x_j \in S_j; \max d(x_i, x_j). \tag{4.7}$$

3. **Average linkage:** Define la proximidad entre los distintos grupos como la proximidad media entre pares de todos los pares de puntos de distintos grupos. Sea S_i el grupo í-esimo y x_i un elemento de grupo i-ésimo, podemos definir al average linkage como:

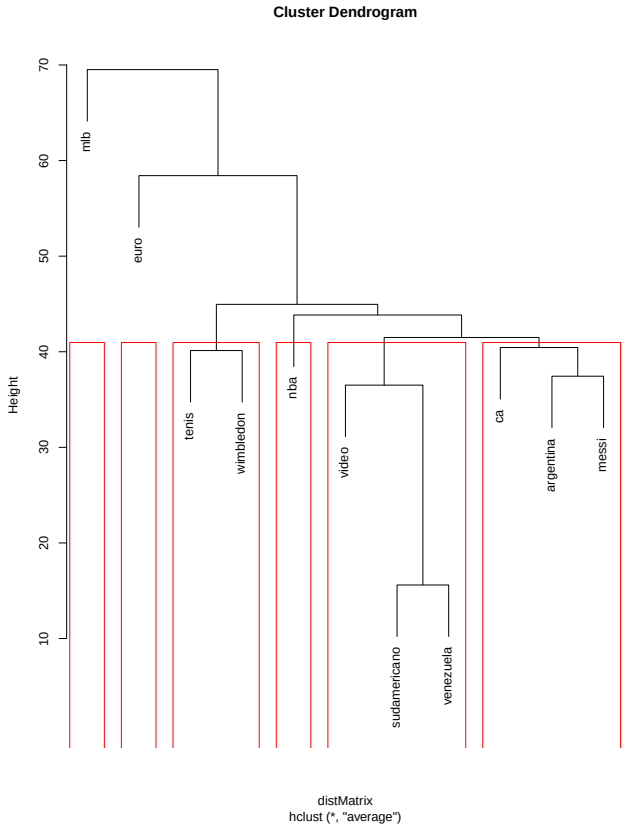
$$\frac{1}{|x_i||x_j|} \sum_{x_i \in S_i} \sum_{x_j \in S_j} d(x_i, x_j). \tag{4.8}$$

4.4.1 Dendograma

Para la representación gráfica del método de clusterización jerárquica hacemos uso de lo que se conoce como un dendograma. Un dendograma es un tipo de representación gráfica o diagrama de datos en forma de árbol que organiza los datos en subcategorías que se van dividiendo en otros hasta llegar al nivel de detalle deseado, asemejándose a las ramas de un árbol que se van dividiendo en otras sucesivamente.

Este tipo de representación permite apreciar claramente las relaciones de agrupación entre los datos e incluso entre los grupos de ellos aunque no las relaciones de similaridad o cercanía entre categorías. Observando las sucesivas subdivisiones se podrá crear una idea sobre los criterios de agrupación de los mismos, la distancia entre los datos según las relaciones establecidas.

El dendograma es la representación gráfica que mejor ayuda a interpretar el resultado de un análisis de cluster. A continuación un ejemplo de como es la representación gráfica del dendograma:



4.5 Clusterización K-medias

4.5.1 Agrupamiento particional

Este tipo de algoritmo obtiene como resultado una partición de los datos iniciales. Un algoritmo particional asigna a un conjunto de objetos **K** grupos sin estructura jerárquica, siendo **K** un número natural menor que el número total de objetos. Este tipo de algoritmos son muy eficientes en aquellas aplicaciones con conjuntos de datos de grandes dimensiones.

Estos algoritmos se caracterizan por producir grupos optimizando una función criterio definida sobre un subconjunto de datos. En los algoritmos particionales la función criterio más utilizada es el error cuadrático medio. Dado un conjunto de entradas $x_j \in \mathbb{R}^d$, $j = 1, \ldots, N$ que se quiere agrupar en un conjunto de **K** grupos , $C = \{C_1, \cdots, C_K\}$; entonces el error cuadrático medio se define como:

$$E(\Gamma, M) = \sum_{i=1}^{K} \sum_{j=1}^{N} \gamma_{ij} \|x_j - m_i\|^2 \tag{4.9}$$

donde $\Gamma = [\gamma_{ij}]$ es la matriz de elementos siendo:

$$\gamma_{ij} = \begin{cases} 1 & \text{sí} x_j \in \text{cluster i} \\ 0 & \text{a otro.} \end{cases}$$

Con

$$\sum_{i=1}^{K} \gamma_{ij} = 1 \forall j; \tag{4.10}$$

y donde $M = [m_1, \cdots, m_K]$ es la matriz de medias, siendo $m_i = \frac{1}{N_i} \sum_{j=1}^{K} \gamma_{ij} x_j$ una muestra de la media del grupo i dados N_i objetos de ese grupo.

Por su parte Γ es llamada matriz de particiones y M una matriz de medias o centroides. Los grupos resultantes de esta función de error cuadrática son frecuentemente llamados particiones de varianza mínima.

4.5.2 Clustering K-medias

El algoritmo de clusterización de **K-medias** fue propuesto por **McQueen** en el año 1967, y en la actualidad es el más usado en las técnicas de clusterización particional, debido a que es eficiente, rápido y simple.

K-medias es basado en la idea de un punto central ó mejor conocido como centroide que es simplemente el punto medio dentro de un grupo de datos.

Descripción del método K-medias

El algoritmo K-medias particiona una colección de vectores $\{d_1, \cdots, d_n\}$ dentro de un conjunto de clusters $C = \{C_1, \cdots, C_K\}$; El algoritmo necesita la selección inicial de K-clusters para la inicialización y encontrar los puntos que minimizen la expresión:

$$\min \sum_{l=1}^{K} \sum_{i: x_{i \in C_l}}^{K} (x_i - m_{il})^2 \tag{4.11}$$

donde:

- m_{il} es el centroide que le toca al punto i

$$m_{il} = \frac{1}{|C_l|} \sum_{i:x_{i \in C_l}}^{K} x_i \tag{4.12}$$

  donde $m_{il} = m_l$, para todo x_i en C_l.

- K: Número de clusters.

Algoritmo iterativo

1. Seleccionar K centroides m_l.
2. Para cada punto x_i se escoge $m_l = \arg\min_l d(x_i, m_l)$.
3. Los puntos se asocian al cluster de acuerdo a (2) y se recalculan los m_l.
4. volvemos a (2).

4.6 Clusterización con restricciones.

La clusterización con restricciones ó también conocida como clustering semi supervisado, ha tomado auge en los últimos años en el área de la minería de datos, la cual permite hacer uso de la experticía del dominio del problema. Este tipo de algoritmo trabaja en la incorporación de de conocimiento previo del problema en forma de restricciones con niveles de instancias.

Los tipos de restricciones fueron introducidos por **Wagstaff** y son conocidos como:

- **Must-link** denotado por $C_{=}(x, y)$: Dadas dos instancias x e y, estas deben pertenecer al mismo cluster.
- **Cannot-link** denotado por $C_{\neq}(x, y)$: Dadas dos instancias x e y, estas no pueden pertenecer al mismo cluster.

Las restricciones must-link y cannot-Link comparten propiedades interesantes. Las restricciones tipo must-link son ejemplos de una relación de equivalencia por lo que son simétricas, reflexivas y transitivas, esto significa que si tenemos $C_{=}(x, y)$ y $C_{=}(y, z)$ entonces $C_{=}(x, z)$ tal que x, y, z forman una componente conectada (connected component), i.e, cada una es conectada a la otra vía explícita ó implícita de una restricción must-link.

Similarmente múltiples componentes conectadas de restricciones must-link puede aumentar las restriciones que conlleva a una restricción cannot-link, entre pares de instancias en componentes diferentes. Las restricciones must-link y cannot-link poseen la característica especial de dividir el conjunto de datos X y especificar cualquier partición del mismo, estas restricciones pueden usarse para mejorar las técnicas de clusterización de diferentes maneras haciendo el estudio del algoritmo **COP-K-medias**.

COP-Kmedias

En el contexto de algoritmos particionales los niveles de instancias de las restricciones son una manera útil de expresar a priori conocimiento sobre cuales instancias deberan o no ser agrupadas considerando las restricciones enunciadas anteriormente llamadas **must-link** y **cannot-link**.
Este algoritmo es una variación del algoritmo de K-medias, valiendose del uso de restricciones en la asignación de los objetos a los distintos clusters.

Algoritmo K-medias con restriciones (COP-Kmedias): Sea D el conjunto de datos, además $C_{=}, C_{\neq} \subseteq DXD$

1. Inicializar los K centroides.
2. Para cada punto d_i en D asignarlo al K mas cercano sin que violen las restricciones.
   Para cada objeto asignado al k más cercano:
   - Restricciones no violadas, asignar al objeto al cluster mas cercano.
   - Restricciones violadas, sí hay un grupo cercano al objeto volver 2 sino a 4.
3. Actualizar los centroides, calculando las medias de los objetos que lo constituyen.
4. Iterar entre 2 y 3 hasta converger.
5. Retornar $C_1 \cdots C_k$.

Este algoritmo aplica a un conjunto de datos D, una partición del mismo haciendo uso de los conjuntos de restricciones must-link y cannot-link, donde la partición de las instancias en D satisfacen todas las restricciones específicas, en el proceso de actualización de los clusters se verifica que ninguna de las restricciones haya sido violada, cabe destacar que las restricciones propuestas nunca podran ser violadas y los objetos que pertenezcan a un cluster satisfacen cada una de las restricciones que posea el cluster.

4.7 Selección del término K

Uno de los puntos importantes al clusterizar via K-medias, es la selección del término K, existen una gran variedad de métodos que tratan de estimar la mejor selección de dicho término. A continuación explicaremos dos de estos métodos: Método del "codo" y usando componentes principales.

4.7.1 Método del codo

Este método consiste en estimar el valor del K, calculando la variación total intra-cluster y este sea minimizado:

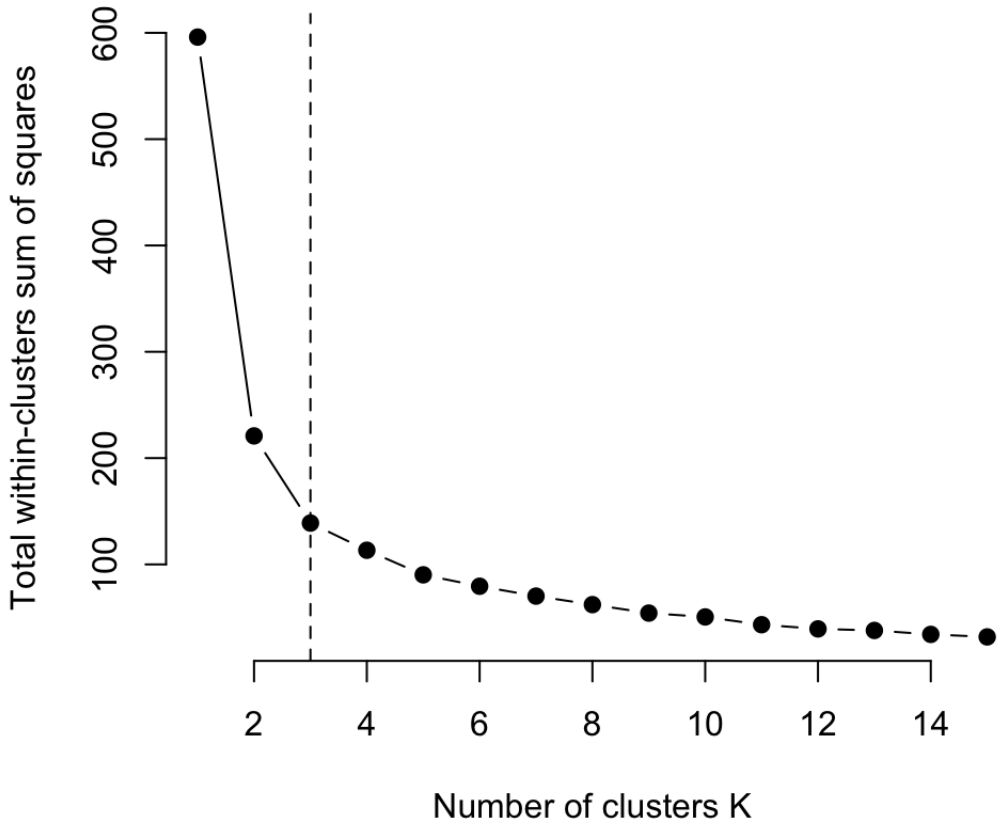
$$\sum_{i=1}^{K} W_{(c_k)} \tag{4.13}$$

donde c_k es el K-ésimo cluster y $W_{(c_k)}$ viene definido como la suma de la distancia al cuadrado entre cada miembro del cluster en ese centroide (error cuadrático medio).

Algoritmo

- Correr el algoritmo de clusterización para diferentes valores de K.
- Para cada K calcular, $W_{(c_k)}$.
- Graficar la curva $W_{(c_k)}$ de acuerdo al número de K.
- Visualizar en la gráfica donde se genera "el codo", este punto es el considerado como el punto optimo para la selección de del número de cluster.

  Un ejemplo de este método es el siguiente:



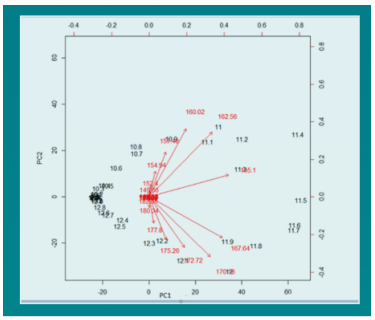
Se puede tomar como un valor optimo $K = 3$.

4.7.2 Clusterización usando ACP

El algoritmo mencionado en el capítulo anterior para el análisis de componentes principales y su relación con la descomposición en valores singulares, también es usado como técnica de clusterización, mediante el estudio de las componentes principales usando la representación gráfica más común llamada **biplot**.

El biplot es una representación gráfica simultanea de los objetos que en nuestro caso son los términos de la matriz términos-documentos (mediante puntos) y de las variables (documentos) mediante vectores, en un mismo sistema bidimensional construido en base de las dos primeras componentes principales. Por lo general las primeras componentes abarcan más del 90% de varianza entre los datos, permitiendo mediante la representación de los biplots interpretar el significado de las componentes (la primera en el eje horizontal y la segunda en el eje vertical) en base de las direcciones de los vectores.

Un ejemplo de lo anterior es lo siguiente:

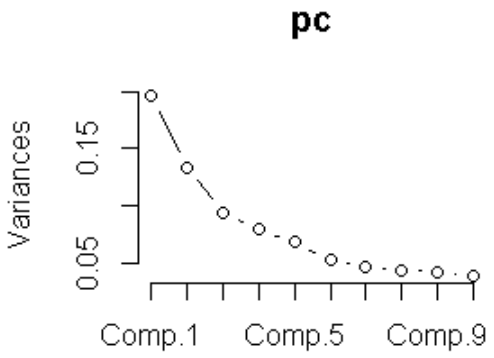


En el proceso de interpretación del biplot, se valoran como parecidos los términos cuyos puntos estan próximos en el biplot, de igual modo tendrán correlación positiva los documentos (variables) con vectores semejantes, asimismo los términos que se encuentran en la dirección de cierto vector tendrán observaciones altas en los documentos representados por el vector.

Algoritmo

- Aplicar ACP al conjunto de datos.
- Observar mediante un gráfico, las componentes que contengan mas de un 85% de varianza.
- Seleccionamos como k al número de componentes que satisfagan la condición el paso anterior.

Un ejemplo de lo anterior:



Se puede tomar como un valor optimo $K = 4$.

Capítulo 5

Clasificación de textos

Un problema de clasificación es aquel que asume que hay un conjunto de objetos etiquetados por algún atributo o rasgo que pertenece a una o a diferentes clases, la etiqueta de la clase es un valor discreto y es conocido para ciertos objetos en un conjunto de datos. El objetivo es construir modelos de clasificación (clasificadores) para asignar la etiqueta de clase correcta a objetos antes no vistos y sin etiquetas, estos métodos fueron llevados a la minería de textos para resolver el problema de clasificación de textos los cuales consiste en asignar a cada documento de una colección una etiqueta que designa la clase a la que este documento pertenece. Durante este capítulo se estudiaran los métodos más usados en la minería de textos: **Maquínas de soporte vectoriales**, el cual es un algoritmo de aprendizaje supervisado y el método probabilístico de **Naive Bayes** (binomial y multinomial) .

5.1 Preliminares para módelos de clasificación

La programación no lineal (PNL) es un proceso de resolución de un sistema de igualdades y desigualdades sujetas a un conjunto de restricciones sobre un conjunto de variables reales desconocidas con una función objetivo a maxímizar o minimizar cuando alguna de las restricciones o la función objetivo no son lineales. De manera general un problema PNL consiste en encontrar $x = (x_1, x_2, \cdots, x_n)$ tal que sea una solución al problema:

$$\begin{gathered} \text{máx } f(x) \\ \text{sujeto a } \ g_i(x) \leq b_i \forall i = 1, 2, \cdots, m \\ x \geq 0, \end{gathered}$$

donde $f(x)$ y $g_i(x)$ son funciones dadas de n variables de decisión. Análogamente puede tratarse de un problema de minimización:

$$\begin{gathered} \text{mín } f(x) \\ \text{sujeto a } \ g_i(x) \geq b_i \forall i = 1, 2, \cdots, m \\ x \geq 0. \end{gathered}$$

Programación cudrática: Es un problema de **NLP**, donde la función objetivo es una función cuadrática, es decir incluye también términos cuadrados o productos de variables.

Condiciones de Karush-Kuhn-Tucker

Teorema Karush-Kuhn-Tucker: Sean $f(x)$, $g_1(x)$, $\cdots$, $g_m(x)$ funciones diferenciables que satisfacen ciertas condiciones de regularidad entonces $\bar{x} = (\bar{x_1}, \bar{x_2}, \cdots, \bar{x_n})$ puede ser una solución óptima para un problema **PNL** sólo si existen m números u_1, u_2,$\cdots$, u_m que satisfagan las siguientes condiciones Karush-Kuhn-Tucker:

1. $\frac{\partial f(\bar{x})}{\partial x_j} - \sum_{i=1}^{m} u_i \frac{\partial g_i(\bar{x})}{\partial x_j} \leq 0, \quad j = 1, 2, \cdots, n.$
2. $\bar{x_j}(\frac{\partial f(\bar{x})}{\partial x_j} - \sum_{i=1}^{m} u_i \frac{\partial g_i(\bar{x})}{\partial x_j}) = 0, \quad j = 1, 2, \cdots, n.$
3. $g_i(\bar{x}) - b_i \leq 0, \quad i = 1, 2, \cdots, m.$
4. $u_i(g_i(\bar{x}) - b_i) = 0, \quad i = 1, 2, \cdots, m.$
5. $\bar{x_j} \geq 0, \quad j = 1, 2, \cdots, n.$
6. $u_i \geq 0, \quad i = 1, 2, \cdots, m.$

Conjunto de entrenamiento y de prueba: En el proceso de trabajo en la minería de datos es importante separar el conjunto de datos en conjuntos de entrenamiento y un conjunto de pruebas. Una gran parte de los datos se representan en un conjunto de entrenamiento y uno aparte menor en el conjunto de prueba, en el conjunto de entrenamiento es donde se construyen los módelos, una vez aplicado el
módelo, se puede aplicar este en un conjunto de prueba para el análisis de resultados.

Teorema de Mercer: Para cualquier función K:$X \times X \longrightarrow \mathbb{R}$ que sea simétrica y semidefinida positiva existe un espacio de Hilbert F y una función $\Phi : X \longrightarrow F$ tal que:

$$K(x, y) = \Phi(x)\Phi(y) \; \forall \; x, y \; \in X \tag{5.1}$$

Donde un espacio de Hilbert es un espacio vectorial de dimensión N con propiedades equivalentes a las de $\mathbb{R}^N$.

5.2 Máquinas de soporte vectoriales

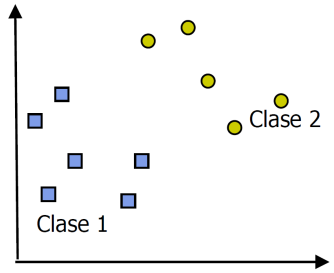
SVM de sus siglas en ingles ó maquinas de soporte vectoriales es una técnica de clasificación [7, 27] la cual en años recientes es considerada más eficaz que las redes neuronales [16], por ende, su gran uso para la resolución de problemas de clasificación.

Para el estudio de **SVM** hay que tomar en cuenta dos posibles casos en la representación del conjunto de datos, el lineal y no lineal, donde la mayoría de los problemas en la realidad son casos no lineales.

Caso linealmente separable

Sean un conjunto D de puntos etiquetados, un ejemplo en un conjunto de entrenamiento puede ser el siguiente:

$$(x_1, y_1), (x_2, y_2), \cdots, (x_i, y_i) \tag{5.2}$$



Cada punto de entrenamiento $x_i \in \mathbb{R}^N$, pertenece a alguna de las dos clases, las cuales se le ha dado un etiqueta $y_i \in \{-1, 1\}$ para $i = 1, 2, \cdots, N$. Este método tiene como objetivo la búsqueda de un híperplano adecuado que separe las dos clases, pero es demasiado restrictivo para que sea de uso práctico. Una solución a esta situación es mapear el espacio de entrada en un espacio de características de una dimensión mayor y buscar en él, el híperplano optimo. Sea $z = \phi(x)$ la notación del correspondiente vector en el espacio de características con un mapeo ϕ de $\mathbb{R}^N$ a un espacio de características de Z, y se requiere encontrar el híperplano

$$wz + b = 0 \tag{5.3}$$

Definido por el par (w, b), tal que se pueda separar el punto x_i de acuerdo a la función

$$f(x_i) = sign(wz_i + b) = \begin{cases} 1 \quad si \quad y_i = 1 \\ \\ -1 \quad si \quad y_i = -1. \end{cases} \tag{5.4}$$

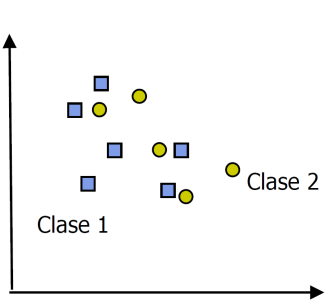
Donde $w \in Z$ y $b \in \mathbb{R}^N$. El conjunto D se dice que es linealmente separable si existe (w, b) tal que las desigualdades

$$\begin{cases} (wz_i + b) \geq 1 \quad si \quad y_i = 1 \quad i = 1, 2, \cdots, l \\ \\ (wz_i + b) \leq -1 \quad si \quad y_i = -1 i = 1, 2, \cdots, l \end{cases} \tag{5.5}$$

sean válidas para todos los elementos del conjunto D. Para este caso podemos encontrar un único híperplano optimo para el cual el margen entre las proyecciones de los puntos de entrenamiento de dos de las diferentes clases sea maximizado.

Caso no linealmente separable

Si el conjunto D no es linealmente separable:



el análisis anterior puede ser generalizado introduciendo algunas variables no-negativas $\varepsilon_i \geq 0$ de tal modo que (4.5) es modificado a:

$$y_i(wz_i + b) \geq 1 - \varepsilon_i, i = 1, 2, \cdots, l. \tag{5.6}$$

Los $\varepsilon_i \geq 0$ en (4.6) son aquellos para los cuales el punto x_i no satisface (4.5), entonces el término $\sum_{i=1}^{l} \varepsilon_i$, puede ser tomado como algún tipo de medida de error en la clasificación, por lo que el problema de hallar el híperplano optimo es entonces redefinido cómo la solución al problema:

$$\begin{aligned} \min \{\frac{1}{2} ww + C \sum_{i=1}^{l} \varepsilon_i\}, \\ \text{sujeto a} y_i(wz_i + b) \geq 1 - \varepsilon_i, \qquad i = 1, 2, \cdots, l, \\ \varepsilon_i \geq 0 \qquad i = 1, 2, \cdots, l. \end{aligned} \tag{5.7}$$

Donde C es una constante definídida como un parámetro de regularización. Este es el único parámetro libre de ser ajustado en la formulación de la **SVM**, el ajuste de este parámetro puede ser un balance entre la maximización del margen y la violación a la clasificación estudiados por [27] [16]. La búsqueda del híperplano optimo es un problema **QP** (programación cuadrática) [16], el cual puede ser resuelto construyendo un lagrangiano y transformándolo en el dual.

$$\begin{aligned} \max w(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j z_i z_j, \\ \text{sujeto a} \sum_{i=1}^{l} y_i \alpha_i = 0, \qquad 0 \leq \alpha_i \leq C, i = 1, \cdots, l. \end{aligned} \tag{5.8}$$

Donde $\alpha = (\alpha_1, \alpha_2, \cdots, \alpha_l)$ es un vector de multiplicadores de lagrange positivos asociados con las constantes en (4.6).

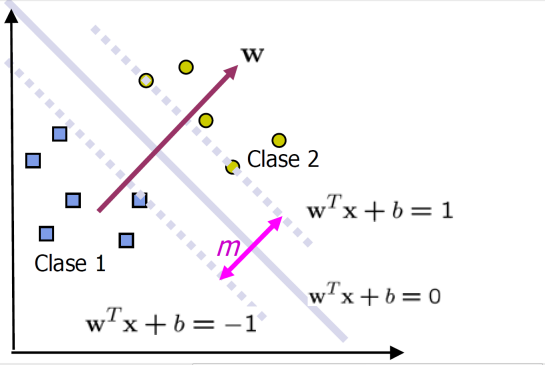
El teorema de **Khun-Tucker** es muy utilizado en la teoria de las **SVM**, de acuerdo a este teorema la solución $\bar{\alpha_i}$ del problema (4.8) satisface:

$$\bar{\alpha_i}(y_i(\bar{w}z_i + \bar{b}) - 1 + \bar{\varepsilon_i}) = 0, \tag{5.9}$$

$$(C - \alpha_i)\varepsilon_i = 0, \qquad i = 1, 2, \cdots, l. \tag{5.10}$$

De esta igualdad se puede deducir que los únicos valores $\bar{\alpha_i} \geq 0$ (4.10) son aquellos para los cuales las constantes en (4.6), se satisfacen con el signo de igualdad. El punto x_i correspondiente

con $\bar{\alpha_i} \geq 0$ es llamado vector de soporte.



En los casos no separables pueden existir dos tipos de vectores de soporte:

Caso 1 : $0 \leq \bar{\alpha_i} \leq C$, el correspondiente vector de soporte x_i satisface las igualdades:

$$y_i(\bar{w}z_i + b) = 1\,, \quad \text{y} \quad \varepsilon_i = 0.$$

Caso 2: $\bar{\alpha_i} = C$, el correspondiente ε_i es diferente de cero y el correspondiente vector soporte x_i no satisface (4.5), se refieren a estos vectores de soporte como errores, el punto x_i correspondiente con $\bar{\alpha_i} = 0$ es clasificado correctamente y esta claramente alejado del margen de decisión.

Para construir el híperplano optimo $\bar{w}z + \bar{b}$ se utiliza

$$\bar{w} = \sum_{i=1}^{l} \bar{\alpha_i} y_i z_i. \tag{5.11}$$

Y el escalar b puede ser determinado de las condiciones de **Khun-Tucker**, la función de decisión de (4.4) y (4.11) es tal que:

$$f(x) = sign(wz + b) = sign(\sum_{i=1}^{l} \alpha_i y_i z_i z + b). \tag{5.12}$$

Truco del kernel para el caso no linealmente separable

Al momento de la práctica, la mayoria de los problemas no son linealmente separables, por lo que se necesitan transformar cada uno de los puntos en el conjunto de entrenamiento a un espacio vectorial de mas alta dimensión (denominado espacio de características) donde sea posible la separación lineal.

Para ello se define una función de transformación $\phi(x) : D \to F$

$$\phi(x) = \begin{cases} |D| = n, |F| = N \\ y \quad N \gg n. \end{cases}$$

La cual recibe vectores del espacio de entrada X, y los transforma en vectores del espacio de características F. Para encontrar esta función ϕ se hace mediante el uso de las llamadas "funciones

Kernel ", las cuales se aplican sobre vectores de X y su resultado es un producto escalar sobre algún espacio de características F.

Funciones kernel

Una función kernel $K(X,Y) : X \times X \rightarrow \mathbb{R}$, asigna a cada par de objetos de entrada x e y un valor real que se corresponde con el producto escalar de sus respectivas imágenes en el espacio de características F; Es decir $k(x,y) = \phi(x)\phi(y)$ para alguna función de transformación implícita, $\phi(x) : X \rightarrow F$. Las funciones kernel permiten calcular productos escalares en F (espacio de características) aplicando la respectiva función kernel sobre X (espacio de entrada). En el proceso del algoritmo se sustituye el producto escalar por la función kernel. El truco del kernel permite que algoritmos lineales se apliquen sobre problemas no-lineales. Haciendo uso del teorema de Mercer se caracterizan las funciones kernel de la siguiente manera:

Teorema: Para cualquier función $K : X \times X \rightarrow \mathbb{R}$ que sea simétrica y semidefinida positiva, existe un espacio de Hilbert F y una función $\phi(x) : X \rightarrow F$ tal que:

$$k(x,y) = \phi(x)\phi(y) \quad \forall x, y \in X.$$

donde un espacio de Hilbert es un espacio vectorial de dimensión N con propiedades equivalentes a las de $\mathbb{R}^N$.

En las **SVM** existen funciones kernel típicas para llevar a cabo la solución de un problema de tipo no-lineal m, estas son las siguientes:

1. **Kernel identidad:** $K(x,y) = xy$.
2. **Kernel polinómico:** $k(x,y) = (xy + r)^p$, $\quad$ r $\in \mathbb{R}$,
   $p =$ Grado del polinomio.
3. **Kernel gaussiano:** $k(x,y) = e^{\dfrac{-\|x-y\|^2}{2\sigma^2}}$, $\quad \sigma^2 = Varianza$.

Algoritmo de SVM en minería de textos

Para modelar el problema de categorización **P** de textos con las **SVM** se define lo siguiente:

- D: Dominio de todos los documentos.
- C: Dominio de todas las categorias pre-definidas.
- Representar al conjunto de documentos, en una forma vectorial, haciendo uso preferiblemente del peso TD-IDF.

- Binarizar el problema P, para tal efecto se considerara cada categoría posible de C, como un problema binario por separado.
- Para cada categoría se genera una **SVM** que decide si cada documento d_i pertenece a la categoría asociada:

$$\phi : D \to C, \quad \text{tal que } \phi(d_i) = c_i$$

d_i es un documento cualquiera.
x_i es el vector de las categorias a las que pertence el documento d_i.

En D aplicaremos cada M_i (SVM) a los documentos y devuelve como resultado las categorias asociadas a M_i que han clasificado como positivas.

Con las clases ya creadas se estudia la similitud de un texto; cuando se ingresa uno nuevo se compara con cada uno de los textos de cada clase que tenga mayor similitud, si no existe similitud se crea una nueva clase para el texto analizado.

5.3 Naive Bayes en clasificación de textos.

A menudo un documento es representado como una "bolsa de palabras"(bag of words) donde una "bolsa" se define como un conjunto que permite la repetición de los elementos, esta representación es muy simple [13] ya que solo usa las palabras y su frecuencia y no toma en cuenta en que orden aparecen en él.

Un documento D, D se clasifica en la clase que tenga la probabilidad previa más alta $P(C|D)$ la cual es expresada usando el teorema de Naive Bayes:

$$P(C|D) = \frac{P(C|D)P(C)}{P(D)} \propto P(D|C)P(C). \tag{5.13}$$

Haciendo uso del teorema de Naive Bayes, estudiaremos dos módelos de documentos probabilísticos, ambos representan los documentos usando el vector de características (feature vector), cuyas componentes son las diferentes frecuencias de las palabras ó mediante una función de peso como TF-IDF, si tenemos un vocabulario V, conteniendo $|V|$ tipos de palabras entonces ft tiene dimensión $d = |V|$.

5.4 Módelo de Bernoulli

En este módelo el documento es representado por un vector binario, el cual representa un punto en el espacio de las palabras y si tenemos un vocabulario V que contiene un conjunto $|V|$, entonces la t-ésima dimensión de un vector corresponde con la palabra w_i en el vocabulario.

Sea b_i el vector de características para el i-ésimo documento D, entonces el t-ésimo elemento de b_i es reescrito como b_{ii} es 0 ó 1 representando la ausencia ó presencia de la palabra w_i en el i-ésimo

documento.

Sea $P(w_i|C)$ la probabilidad de una palabra w_i ocurriendo en un documento de clase C. La probabilidad de w_i de no ocurrencia en un documento de esta clase es dada por $(1 - P(w_t|C))$, si tomamos la suposición de Naive Bayes de que la probabilidad de ocurrencia de cada palabra en un documento es independiente de las ocurrencias de las otras palabras entonces podemos escribir la probabilidad del documento $P(D_i|C)$ en términos de la probabilidad de la palabra $P(w_i|C)$:

$$P(w_i|C) \sim P(b_i|C) = \prod_{i=1}^{|V|}[b_{it}P(w_t|C) + (1 - b_{it})(1 - P(w_t|C)]. \tag{5.14}$$

Este producto va sobre todas las palabras en el vocabulario, sí la palabra w_t esta presente entonces $b_{it} = 1$ y la probabilidad requerida es $P(w_t|C)$; si la palabra w_t nó esta presente entonces $b_{it} = 0$ y la probabilidad requerida es $(1-P(w_t|C)$, podemos imaginar a este módelo como la generación de vectores de características de documentos de la clase C, en el que el vector b_i de documentos se modela con una colección de $|V|$ ensayos de Bernoulli que tiene una probabilidad de exito igual a $P(w_t|C)$. Cada uno de los parámetros son las probabilidades de cada palabra dada la clase de documentos $P(w_t|C)$; El módelo es también parametrizado por las probabilidades previas $P(C)$, se puede estimar estos parámetros desde un conjunto de entrenamiento de documentos etiquetados por las clases $C = k$. Sea $n_k(w_t)$ el número de elementos de la clase $C = k$ en el cual w_t es observado y sea N_k el número total de documentos de esa clase. Podemos estimar los parámetros de las probabilidades de la palabra como:

$$\hat{P}(w_t|C = k) = \frac{n_k(w_t)}{N_k}. \tag{5.15}$$

La frecuencia relativa de los documentos de la clase $C = k$, que contienen la palabra w_t.

Si hay N documentos en total en el conjunto de entrenamiento entonces la probabilidad previa de la clase $C = k$ puede estimarse como la frecuencia relativa de los documentos de la clase $C = k$:

$$\hat{P}(C = k) = \frac{N_k}{N}. \tag{5.16}$$

Así al conjunto de entrenamiento de documentos (cada uno etiquetado con una clase) y un conjunto de K clases, se puede estimar el módelo de Bernoulli de clasificación de textos como sigue:

1. Definir el vocabulario V, el número de palabras en el vocabulario define la dimensión del ft.
2. Contar el siguiente conjunto de entrenamiento:
   - N: Número total de documentos.
   - N_k: Número de documentos etiquetados de la clase $C = K$, para $k = 1, 2, \cdots, K$.
   - $n_k(w_y)$: Número de elementos de clase $C = k$ que contienen la palabra w_t para toda clase y para cada palabra en el vocabulario.
3. Estimar las probabilidades $P(w_t|C = k)$ usando la ecuación 5.15.
4. Estimar las probabilidades previas $P(C = k)$ usando 5.16.

Para clasificar un documento D_j no etiquetado, se estima la probabilidad posterior de cada clase combinando las ecuaciones 5.13 y 5.14

$$\begin{aligned} P(C|D_j) &= P(C|b_j), \\ &\propto P(b_j|C)P(C), \\ &\propto P(C)\prod_{i=1}^{|V|}[b_{it}P(w_t|C) + (1-b_{it})(1-P(w_t|C)]. \end{aligned} \tag{5.17}$$

La distribución multinomial

Generalmente si tenemos n-items de d-tipos, n_1 de tipo 1, n_2 de tipo 2, $\cdots$, n_d de tipo d (tal que $n_1 + n_2, \cdots, n_d = n$) entonces el número total de permutaciones es dada por

$$\frac{n!}{n_1!n_2!\cdots n_d!}. \tag{5.18}$$

Estos números son llamados coeficientes multinomiales. Ahora supongamos que tenemos un conjunto que d items con $d \geq 2$ tipos diferentes y la proporción de items que son de tipo t es $P_t(t = 1, 2, \cdots, d)$ con $\sum_{t=1}^{d} P_t = 1 \qquad P_t > 0 \quad \forall t$. Supongamos que n items son extraídos al azar (con reemplazo) y X_t denota el número de items de tipo t. El vector $X = (x_1, x_2, \cdots, x_d)^T$ tiene como distribución normal con parámetros n y $p_1, p_2, \cdots$, p_d definida por:

$$P(x) = \frac{n!}{x_1!x_2!\cdots x_d!}p_1^{x_1}p_2^{x_2}\cdots p_d^{x_d} = \frac{n!}{\prod_{i=1}^{d} x_t}\prod_{i=1}^{d} p_t^{x_i}. \tag{5.19}$$

El producto $\prod_{i=1}^{d} p_t^{x_i}$ da la probabilidad de una secuencia de resultado con conteo X, el coeficiente multinomial, cuenta el número de tales secuencias que ahí se encuentran.

5.5 Módelo multinomial para clasificación de documentos.

En el módelo multinomial, los vectores características del documento capta la frecuencia de palabras, no así la presencia o ausencia.

Sea x_i el vector de características del módelo multinomial por el i-ésimo documento D_i , el t-ésimo elemento de x_i es escrito como x_{it}, el conteo del número de veces de la palabra w_t en el documento D_i.

Sea $n_i = \sum_{i=1} x_{it}$, el número total de palabras en el documento D_i y sea $P(w_t|C)$ de nuevo la probabilidad de la palabra w_t que ocurre en la clase C. Esta es estimada usando la información de la frecuencia de la palabra en el vector de características del documento, se considera la suposición de Naive Bayes, la cual consiste en asumir que las palabras en el documento son todas independientes entre si, podemos escribir la probabilidad del documento $P(D_i|C)$ como una distribución multinomial como en la expresión en 5.18 donde el número de extracción corresponde con la longitud del documento y la proporción de extraer el item t es la probabilidad de la palabra

de tipo t que ocurre en un documento de clase C, $P(w_t|C)$, además

$$P(D_i|C) \sim P(x_i|C) = \frac{n_i!}{\prod_{t=1}^{|V|} x_{it}!} \prod_{t=1}^{|V|} p_t(w_t|C)^{x_{it}} \propto \prod_{t=1}^{|V|} p_t(w_t|C)^{x_{it}}. \tag{5.20}$$

A menudo no se necesita el término de normalización $\frac{n_i!}{\prod_{t=1}^{|V|} x_{it}!}$ porque este no depende de la clase C.

El término $\prod_{t=1}^{|V|} p_t(w_t|C)^{x_{it}}$ puede ser interpretado como el producto de probabilidades de palabras para cada palabra en el documento con las palabras que participan en la repetición.

Como en el módelo de Bernoulli los parámetros de las probabilidades, son las probabilidades de cada palabra de la clase de documento $P(w_t|C)$ y los parámetros del módelo también incluyen las probabilidades previas $P(C)$, para estimar estos parámetros el vector de características desde el conjunto de entrenamiento es etiquetado con la clase $C = k$, sea z_{ik} una variable indicadora que es igual a 1 cuando D_i tiene la clase $C = k$ ó 0 en otro caso. Sí N es de nuevo el número total de documentos entonces tenemos:

$$\hat{P}(w_t|C = k) = \frac{\sum_{i=1}^{N} x_{it} z_{ik}}{\sum_{i=1}^{|V|} \sum_{i=1}^{N} x_{it} z_{ik}}. \tag{5.21}$$

Como una estimación de la probabilidad $P(w_t|C = k)$, como la frecuencia relativa de w_t en documentos de clase $C = k$ con respecto al número total de palabras en el documento de esa clase. La probabilidad previa de la clase $C = k$ es estimada como en el caso anterior. Así dado un conjunto de entrenamiento de documentos (cada uno etiquetado con una clase) y un conjunto de k clases, podemos estimar el módelo de clasificación de textos multinomial como sigue:

1. Definir el vocabulario V, el número de palabras en el vocabulario define la dimensión del vector de características.
2. Contar el conjunto de entrenamiento:
   - N: Número total de documentos.
   - N_k: Número de elementos etiquetados con la clase $C = k$ para cada clase $k = 1, 2, \cdots, k$.
   - x_{ij}: Frecuencia de la palabra w_t en el documento D_k calculado para cada palabra w_t en V.
3. Estimar las probabilidades $P(w_t)|C = k$ usando 5.20.
4. Estimar las probabilidades previas $P((C = k))$ usando 5.16.

Para clasificar un documento D_i no etiquetado, estimar la probabilidad posterior para cada clase combinando 5.13 y 5.19

$$P(C|D_j) = P(C|x_j) \propto P(x_j|C)P(C) \propto P(C) \prod_{t=1}^{|V|} P_t(w_t|C)^{x_{it}}. \tag{5.22}$$

A diferencia del módelo de Bernoulli, las palabras que no ocurren en el documento (i.e para x_{it}=0) no afectan la probabilidad, así podemos escribir la probabilidad posterior en términos de palabras u las cuales ocurren en el documento:

$$P(C|D_j) \propto P(C) \prod_{h=1}^{|D_j|} P(u_h|C). \tag{5.23}$$

Donde u_h es la h-ésima palabra en el documento D_j.

Problema de la probabilidad cero

Un inconveniente con la estimación de la frecuencia relativa en la ecuación 5.20 para el módelo multinomial es que si el resultado del conteo da 0, este estima que la probabilidad es 0, lo anterior es un incoveniente [13] debido a que la ecuación de Naive Bayes para la probabilidad en 5.19 implica tomar un producto de probabilidades, si alguno de los términos del producto es cero entonces implica que todo el producto es cero, de donde la probabilidad de que el documento pertenezca a la clase es cero, lo cual es imposible. El hecho de que una palabra no se encuentra en una clase de documentos en el conjunto de entrenamiento no significa que no pueda ocurrir en cualquier documento de esa clase, el problema de la ecuación 5.19 es que no estima las probabilidades de las palabras que no ocurren en el conjunto de datos, incluso sí la palabra no se observa para la clase $C = k$ en el conjunto de entrenamiento, aunque por lo anterior seria útil que $P(w|C = k) > 0$. Dado de que la suma de las probabilidades debe ser 1 y de que las probabilidades de las palabras no observadas no estan bien definidas, dar solución a este problema consiste en extraer en cantidad pequeña la probabilidad asignada a los eventos observados y distribuirlas a través de los eventos no observados, una manera simple de hacer esto es usar la estimación de Laplace la cual viene dada por la siguiente ecuación

$$P_l(w_t|C = k) = \frac{1 + \sum_{i=1}^{N} x_{it} z_{ik}}{|V| + \sum_{i=1}^{|V|} \sum_{i=1}^{N} x_{it} z_{ik}} \tag{5.24}$$

Capítulo 6

Implementación de códigos en R.

Debido a la creación de software (públicos y privados) capaces de implementar los distintos algoritmos de la minería de datos y a su vez poder manejar grandes volúmenes de datos, nos permiten optimizar nuestro trabajo. El software por excelencia para el análisis estadístico es **R**, el cual está desarrollado bajo el proyecto Apache de licencia GNU de uso libre para cualquiera, además es un proyecto colaborativo.

En este trabajo se utilizará **R** para la implementación de los algoritmos estudiados de la mineríá de texto, analizando la cuenta de un usuario de la apliacación web de microblogging **Twitter** y también para el desarrollo de una aplicación web, usando los diferentes paquetes de **R**. Cada uno de estos códigos serán explicados a continuación, además seran compartidos en un repositorio de la plataforma Github.

6.1 R

Es un entorno y lenguaje de programación enfocado en el análisis estadístico, creado por Robert Gentleman y Ross Ihaka del departamento de estadística de la universidad de Auckland en 1993. **R** es parte del sistema GNU y se distibuye bajo la licencia GNU GPL y esta disponible los sistemas operativos Linux, Unix, Windows y Mac.

R proporciona al usuario una amplia gama de herramientas estadísticas (módelos lineales, no lineales, análisis de series temporales, algoritmos de machine learning, gráficas , etc). **R** como lenguaje de programación, permite al usuario el agregar sus propias funciones además, **R** es compatible con diversos formatos de datos y su capacidad gráfica es de alta calidad.

6.1.1 Extensiones y paquetes

R forma parte de un proyecto colaborativo y abierto, sus usuarios pueden publicar paquetes que extienden su configuración básica, existe un repositorio oficial de paquetes que para el año 2009 supero la cifra de los 2000.

6.2 Twitter

Es un servicio de microblogging, creado por Jack Dorsey en el año 2006. La red permite enviar mensajes de texto plano de corta longitud máximo 140 caracteres llamados Tweets, que se muestran en la página principal del usuario. Los usuarios pueden suscribirse a los tweets de otros usuarios, a esto se le llama "seguir" y a los usuarios abonados se les llama seguidores.

6.2.1 Como obtener data generada por Twitter

Para obtener datos desde Twitter, se utiliza el protocolo OAuth, el cual permite a los sitios web ó aplicaciones acceder a recursos protegidos de un servicio web a través de una API (Application Programing Interface), la cual es un conjunto de funciones y procedimientos que permite al usuario extraer los datos del sitio web.

Para poder registrar la API hay que iniciar sesión en la página de desarrolladores de Twitter `https://dev.twitter.com/` posteriormente registrar la API para poder generar los tokens de acceso, los cuales permitirán descargar la data de twitter usando un script en **R**.

Dentro del protocolo OAuth es necesario conocer una serie de parámetros, los cuales son usados para crear cualquier API con Twitter.

- Proveedor de servicio (Service Provider): Aplicación web donde se usara el protocolo OAuth (Twitter).
- Usuario (User): Individuo que posee una cuenta en el proveedor de servicio.
- Consumidor (Consumer): Un sitio web ó aplicación con acceso al proveedor de servicio.
- Recursos Protegidos (Protected Resources): Datos controlados por el proveedor de servicios con acceso mediante la autentificación del usuario.
- Clave del Consumidor (Consumer Key): Valor usado por el consumidor para identificarse en el proveedor de servicio.
- Secreto del Consumidor (Consumer secret): Valor secreto que determina la propiedad de la clave del consumidor.
- Solicitud de Token (Request Token): Valor usado por el consumidor para obtener una autorización del usuario.
- Token de Acceso (Acces Token): Valor usado por el consumidor para obtener acceso a los recursos protegidos por el usuario.
- Token secreto (Token secret): Valor secreto usado por el consumidor para establecer la propiedad del token.

6.3 Códigos en R

Para esta sección realizaremos el estudio de las cuentas de twitter de los usuarios **Hilary Clinton** y **Donald Trump**. En **R** implementaremos una serie de scripts, cada uno encargado de los siguientes procesos:

- Descargar de manera automática la data de twitter, posteriormente generar el conjunto de datos en formato **RData**.
- Convertir el conjunto de documentos (tweets) de ambas cuentas, en un conjunto tratable, concepto explicado en el segundo capítulo.
- Crear la matriz de Términos-Documentos, para encontrar los términos frecuentes, aplicar los métodos de clusterización jerárquica y K-medias, encontrar las palabras relacionadas basándose en un nivel de correlación.
- Implementar un clasificador Naive Bayes y SVM para clasificar un conjunto de tweets.

6.3.1 Paquetes a utilizar de R

- **TwitterR**: Encargado de la comunicación de R con la plataforma Twitter.
- **tm**: Paquete que provee herramientas para la minería de texto.
- **stringr**: Funciones para trabajar con cadenas de caracteres.
- **ggplot2**: Permite realizar gráficas de alta calidad.
- **SchedulerR**: Ayuda a generar rutinas automáticas de scripts de R.
- **Shiny**: Paquete que permite el desarrollo de aplicaciones web.
- **RTexTool**: Paquete de Herramientas para minería de texto.

6.3.2 Descarga de los tweets

```
#R  paquetes
library(twitteR)
library(httr)
#Definimos el directorio de trabajo
setwd("C:/Users/Leonardo/Documents/schedulerR")
#Protocolo OAuth de twitter
api_key <- "xxxxxxxxxxxxxxxxxxxxxxxxx"

api_secret <- "yyyyyyyyyyyyyyyyyyyyyyy"

access_token <- "zzzzzzzzzzzzzzzzzzzzzzz"

access_token_secret <- "ooooooooooooooooooooooooo"

setup_twitter_oauth(api_key,api_secret,access_token,access_token_secret
   )
```

```
1

#
#Una vez lista la autentificacion, procedemos a extraer los tweets
   usando la funcion
#usertimeline, en la cual hay que indicar  de que usuario queremos
   extraer los tweets  y
#que cantidad queremos.


# extraer tweets usando el usertimeline
#HillaryClinton
clinton_tweets <- userTimeline("HillaryClinton", n = 1600)
tweetsc.df <- twListToDF(clinton_tweets)
dim(tweetsc.df)
1


#ahora generamos una variable llamada date  y la convertimos en
   character
date<-Sys.Date()
date<-as.character(date)
name<-paste(date,".RData")
#ahora  guardamos los tweets bajo el nombre del dia de la descarga
save(tweetsc.df, file =name)
```

Para que la descarga se haga de manera automática, haremos uso del paquete schedulerR, este paquete permite que se ejecute de manera automatica un script en **R** especificando la fecha, hora, y por cuanto tiempo se quiera ejecutar la rutina.

```
taskscheduler_create(taskname = "taskteetsclinton", rscript =
   clintontweets,
schedule = "DAILY", starttime = "11:30", startdate = format(Sys.Date(),
    "%d/%m/%Y"))
```

Lo anterior se aplicó de la misma manera para la otra cuenta.

```
#R  paquetes
library(twitteR)
library(httr)
setwd("C:/Users/Leonardo/Documents/schedulerR")
# Twitter protocolo OAuth


api_key <- "xxxxxxxxxxxxxxxxxxxxxxxxxxx"

api_secret <- "yyyyyyyyyyyyyyyyyyyyyyy"

access_token <- "zzzzzzzzzzzzzzzzzzzzzzzz"

access_token_secret <- "ooooooooooooooooooooooooo"
```

```
15
16 setup_twitter_oauth(api_key,api_secret,access_token,access_token_secret
       )
17 1
18
19 # extraer tweets usando el usertimeline
20 #Donald Trump
21 trump_tweets <- userTimeline("realDonaldTrump", n = 1600)
22 tweetst.df <- twListToDF(trump_tweets)
23 dim(tweetst.df)
24
25
26
27
28 #ahora generamos una variable llamada date  y la convertimos en
       character
29 date<-Sys.Date
30 date<-as.character(date)
31 name<-paste(date,".RData")
32 #ahora  guardamos los tweets bajo el nombre del dia de la descarga
33 save(tweetst.df, file =name)
34
35 #schedulerR
36 taskscheduler_create(taskname = "taskteetsclinton", rscript =
       trumptweets,
37 schedule = "DAILY", starttime = "15:30", startdate = format(Sys.Date(),
       "%d/%m/%Y"))
```

6.3.3 Convertir los tweets en textos tratables

Se aplican las técnicas estudiadas en el primer capítulo, para ello se utilizó el paquete de **R** llamado **tm**.

```
1  View(clintontweets)# Observemos el conjunto de datos
2  #Paquetes a utilizar
3  library(tm)
4  library(ggplot2)
5  library(stringr)
6  #Limpieza del Texto Pre-Procesamiento
7  #Construyamos el Corpus indicando que la fuente es un vector de
       caracteres
8  myCorpus<-Corpus(VectorSource(clintontweets$text))
9
10 #El texto del corpus  se convierte en texto plano
11 mycorpus <- tm_map(myCorpus, PlainTextDocument)
12
13 #El texto del corpus  se convierte en texto plano
14 mycorpus <- tm_map(myCorpus, PlainTextDocument)
15 #Se remueven los signos de puntuacion
16 myCorpus <- tm_map(myCorpus, removePunctuation)
```

```
# Se remueven los numeros
myCorpus <- tm_map(myCorpus, removeNumbers)
# Se remueven los  URLs
removeURL <- function(x) gsub("http[[:alnum:]]*", "", x)
myCorpus <- tm_map(myCorpus, removeURL)
#El texto del corpus  se convierte en  minusculas
myCorpus <- tm_map(myCorpus, tolower)
#El texto del corpus  se convierte en texto plano
mycorpus <- tm_map(myCorpus, PlainTextDocument)
#Se remueven las  stop words
myStopwords <- c(stopwords("english"))
myStopwords <- c(stopwords("english"), "the", "to","of","on","that","in
    ","in","for","a")
myCorpus <- tm_map(myCorpus, removeWords, myStopwords)
#Guardamos una copia del  corpus
myCorpusCopy <- myCorpus
# Aplicamos el proceso de steaming
myCorpus <- tm_map(myCorpus, stemDocument)

#Observemos una muestra del corpus
for (i in 6:11) {
cat(paste("[[", i, "]] ", sep = ""))
writeLines(myCorpus[[i]])
}
```

Una vez generado el corpus y convertido en un texto tratable pasamos a crear la matriz de términos-documentos.

```
tdm <- TermDocumentMatrix(myCorpus,control = list(wordLengths = c(1,
    Inf)))
tdm2 <- removeSparseTerms(tdm, sparse = 0.95)
m2 <- as.matrix(tdm2)
```

Una vez creada la matriz de términos-documentos, buscamos los términos frecuentes con una frecuencia mínima de 45 y los visualizamos haciendo uso del paquete ggplot2.

```
#Veamos los terminos con una frecuencia minima de 45 apariciones
term.freq <- rowSums(as.matrix(tdm))
term.freq <- subset(term.freq, term.freq >= 45)
#Lo anterior se guarda en data frame y en RData
df <- data.frame(term = names(term.freq), freq = term.freq)
save(df,file="tqfclinton50.RData")
#Ahora los terminos Frecuenest los visualizamos con ggplot2
library(ggplot2)
ggplot(df, aes(x = term, y = freq)) + geom_bar(stat = "identity") +
xlab("Terms") + ylab("Count") + coord_flip()
```

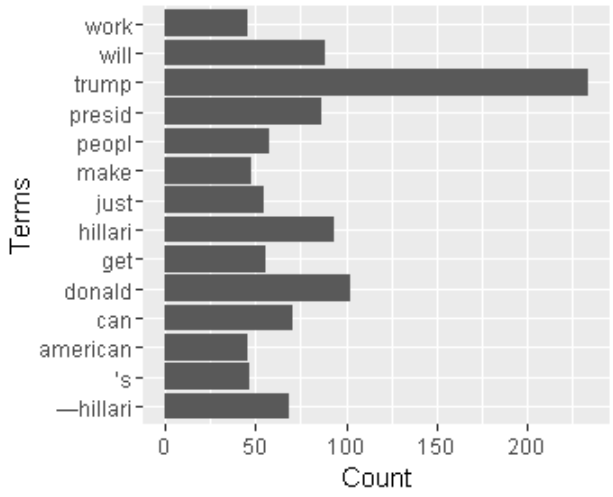


Imagen 6.1: Términos frecuentes.

Del gráfico de términos frecuentes, podemos observar que el término ***Trump*** es él que más tiene aparición en los tweets, también con una frecuencia alta aparecen la palabras ***president, people y can***.

Asociación de términos

Para ciertas palabras, veremos que términos estan asociados unos con otros, para ello previamente se necesita escoger un nivel minímo de correlación. Escogeremos un nivel de 0.2, si se fuera un escogido un valor muy cercano a 1 el proceso de asociación fuera sido muy restrictivo, debido a eso es conveniente tomar un nivel de correlación que permita ver las distintas asociaciones entre todos los términos en el documento.

Relaciones con "Obama"

```
findAssocs(tdm, "obama", 0.2)
```

```
##         obama
## barack   0.53
## barack  0.46
## data     0.46
## didnt    0.46
## econom   0.46
## owe      0.46
## plain    0.46
## sendoff  0.46
## simple   0.46
## admit    0.37
## vote    0.35
## apolog   0.32
## peddl    0.32
## born     0.30
## show     0.26
## said     0.23
## novemb   0.21
```

De los valores observados el nivel de correlación promedio es de 0.34, además destaca la correlación del término ***Obama*** con los térmimos ***economy, plain*** los cuales tienen un nivel de correlación de 0.46.

Relaciones con "need"

```
findAssocs(tdm, "need", 0.2)
```

```
##            need
## understand 0.33
## add        0.25
## answernow 0.25
## danger     0.25
## healthi    0.25
## kill       0.25
## light      0.25
## name       0.25
## presidentw 0.25
## qualif     0.25
## seriously 0.25
## tide       0.25
## unbear     0.25
## come       0.24
## list       0.22
## union      0.22
## action    0.21
## humbl      0.21
## pose       0.21
## quit      0.21
## tempera    0.21
## ties      0.21
## tuition    0.21
## someon     0.20
```

De los valores observados el nivel de correlación promedio es de 0.21, además destaca la correlación del término ***need*** con el térmimo ***understand*** con un nivel de 0.33 y con los términos ***qualified, seriously*** los cuales tienen un nivel de correlación de 0.25.

Relaciones con "work"

```
findAssocs(tdm, "work", 0.2)
```

```
##                work
## deal           0.26
## togeth         0.26
## alongsid       0.25
## around         0.25
## dictator       0.25
## disabilities   0.25
## headon         0.25
## incid          0.25
## known          0.25
## main           0.25
## mark           0.25
## qaddafi        0.25
## respect        0.25
## reward         0.25
## shot           0.25
## street         0.25
## street         0.25
## tell           0.25
## terencecrutch  0.25
## unarm          0.25
## wellsfargo     0.25
## buy            0.21
## done           0.21
## hard           0.21
```

De los valores observados el nivel de correlación promedio es de 0.24, además destaca la correlación del término ***work*** con los térmimos ***deal y together*** con un nivel de 0.26 y con los términos ***known, respect, dictator*** un nivel de correlación de 0.25.

Relaciones con "mexico"

```
findAssocs(tdm, "mexico", 0.2)
```

```
##           mexico
## case        1.00
## disastr     1.00
## studi       1.00
## school      0.58
## diplomaci   0.50
## trip        0.45
## wall        0.35
```

De los valores observados el nivel de correlación promedio es de 0.69, además destaca la correlación con los térmimos ***study, case y dissaster*** con un nivel de 1 y con el término ***wall*** un nivel de correlación de 0.35.

Relaciones con "trump"

```
findAssocs(tdm, "trump", 0.2)
```

```
##           trump
## donald     0.54
## answernow  0.22
## light      0.22
```

De los valores observados el nivel de correlación promedio es de 0.33, además destaca la correlación con el térmimo ***answer-now*** con un nivel de 0.22.

Se realizó el mismo proceso anterior a los tweets de la otra cuenta a estudiar:

```

Paquetes a utilizar
library(tm)
library(ggplot2)
library(stringr)
#Limpieza del Texto Pre-Procesamiento
#Construyamos el Corpus indicando que la fuente es un vector de
   caracteres
myCorpus<-Corpus(VectorSource(tweetst.df$text))

#El texto del corpus  se convierte en texto plano
mycorpus <- tm_map(myCorpus, PlainTextDocument)
#El texto del corpus  se convierte en  minusculas
myCorpus <- tm_map(myCorpus, tolower)
#El texto del corpus  se convierte en texto plano
mycorpus <- tm_map(myCorpus, PlainTextDocument)
#Se remueven los signos de puntuacion
myCorpus <- tm_map(myCorpus, removePunctuation)
# Se remueven los numeros
myCorpus <- tm_map(myCorpus, removeNumbers)
# Se remueven los  URLs
removeURL <- function(x) gsub("http[[:alnum:]]*", "", x)
myCorpus <- tm_map(myCorpus, removeURL)

#Se remueven las  stop words
myStopwords <- c(stopwords("english"))
myStopwords <- c(stopwords("english"), "the", "to","of","on","that","in
   ","in","for","a")
myCorpus <- tm_map(myCorpus, removeWords, myStopwords)
#Guardamos una copia del  corpus
myCorpusCopy <- myCorpus
```

```
# Aplicamos el proceso de steaming
myCorpus <- tm_map(myCorpus, stemDocument)

#Observemos una muestra del corpus
for (i in 6:11) {
cat(paste("[[", i, "]] ", sep = ""))
writeLines(myCorpus[[i]])
}

#Ahora ya el conjunto de tweets paso a ser un texto tratable
#se crea la matriz de Terminos-Documentos
tdm <- TermDocumentMatrix(myCorpus,control = list(wordLengths = c(1,
    Inf)))
tdm2 <- removeSparseTerms(tdm, sparse = 0.95)
m2 <- as.matrix(tdm2)

#Veamos los terminos con una frecuencia minima de 25 apariciones
term.freq <- rowSums(as.matrix(tdm))
term.freq <- subset(term.freq, term.freq >= 25)
#Lo anterior se guarda en data frame y en RData
df <- data.frame(term = names(term.freq), freq = term.freq)
save(df,file="tqfclinton50.RData")
#Ahora los terminos Frecuenest los visualizamos con ggplot2
library(ggplot2)
ggplot(df, aes(x = term, y = freq)) + geom_bar(stat = "identity") +
xlab("Terms") + ylab("Count") + coord_flip()
```

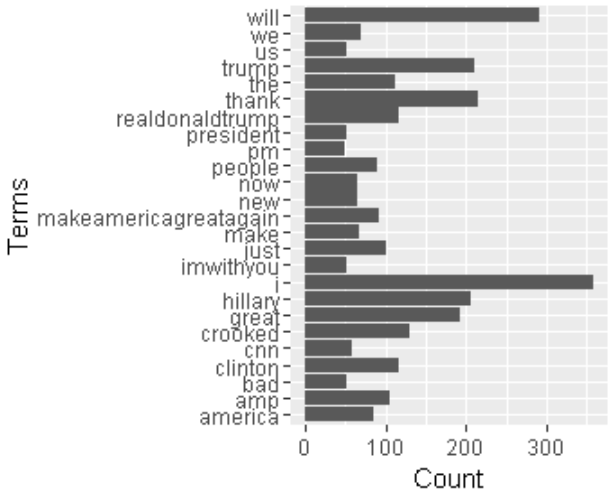


Imagen 6.2: Términos frecuentes.

Del gráfico de términos frecuentes, podemos observar que el término ***I*** es él que más tiene aparición en los tweets, también con una frecuencia alta aparecen la palabras ***president, make America great again, croocked y cnn***.

Asociación de términos

Basándonos del mismo análisis anterior seleccionamos 0.2 como nivel de correlación.

Relaciones con "cnn"

```
findAssocs(tdm, "cnn", 0.2)
```

```
##             cnn
## rate       0.26
## annebellar 0.25
## garbag     0.25
## higher     0.25
## network    0.25
## predict    0.25
## panel      0.24
## bias       0.23
```

De los valores observados el nivel de correlación promedio es de 0.25, además destaca la correlación con el término ***rate*** con un nivel de 0.26, con los términos ***garbage y predit*** un nivel de correlación de 0.25

Relaciones con "bad"

```
findAssocs(tdm, "bad", 0.2)
```

```
##                bad
## judgement     0.53
## decis         0.26
## allen         0.24
## brainpow      0.24
## overratedprob 0.24
## skill         0.24
## track         0.24
## unfit         0.20
```

De los valores observados el nivel de correlación promedio es de 0.27, además destaca la correlación con el término ***judgement*** con un nivel de 0.53, con los términos ***decision, skill, brainpower*** un nivel de correlación de 0.24

Relaciones con "mexico"

```
findAssocs(tdm, "mexico", 0.2)
```

```
##          mexico
## benefit    0.48
## lui        0.48
## nieto      0.34
## invit      0.28
## belt       0.24
## courant    0.24
## enriqu     0.24
## f          0.24
## financ     0.24
## hartford   0.24
## mcallen    0.24
## mexican    0.24
## minist     0.24
## pena       0.24
## peña       0.24
## qualiti    0.24
## rail       0.24
## rust       0.24
## stolen     0.24
## vicent     0.24
## wonder     0.20
```

De los valores observados el nivel de correlación promedio es de 0.26, además destaca la correlación con el término ***benefit*** con un nivel de 0.48, con los términos ***Peña, invite y financial*** un nivel de correlación de 0.24

Relaciones con "Clinton"

```
findAssocs(tdm, "clinton", 0.2)
```

```
##           clinton
## hillari      0.50
## crook        0.31
## judgement    0.21
```

De los valores observados el nivel de correlación promedio es de 0.34, además destaca la correlación con el término ***crooked*** con un nivel de 0.31, con el término ***judgement*** un nivel de correlación de 0.21

Relaciones con "Obama"

```
findAssocs(tdm, "obama", 0.2)
```

```
##          obama
## presid    0.40
## maker     0.31
## correct   0.23
## ice       0.21
## natur     0.21
## socal     0.21
## stairway  0.21
## syria     0.21
## trail     0.21
## crimea    0.20
## els       0.20
## wouldnt   0.20
```

De los valores observados el nivel decorrelación promedio es de 0.23, además destaca la correlación con el término ***Syria*** con un nivel de 0.21, con los términos ***Crimea y would not*** un nivel de correlación de 0.20

6.4 Proceso de clusterización

Una vez los documentos ya preprocesados con las técnicas estudiadas y obteniendo su representación matricial procedemos a realizar los análisis de clusterización para poder encontrar los temas de que más se comentan en la colección de tweets.

6.4.1 Cluster jerárquico para la cuenta Clinton

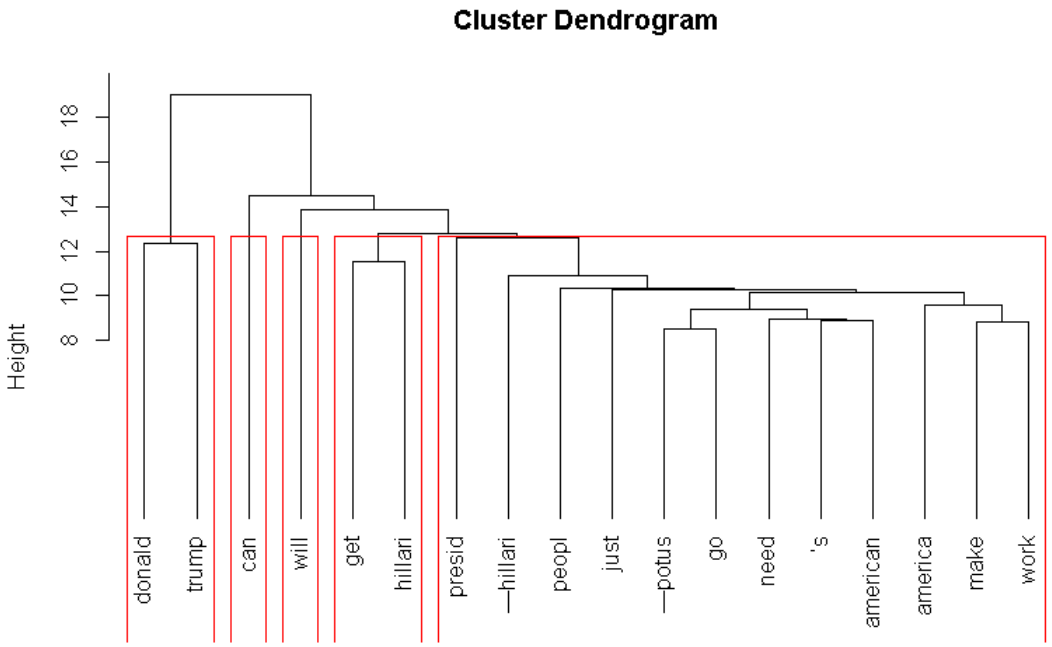
Se realizó la clusterización jerárquica usando las tres medidas de distancia explicada en los capítulos anteriores: Euclídea, Coseno, Jaccard, la idea de implementar el método con varias distancia es para comparar cual de ellas forma una manera más clara la selección de dichos grupos:

```
# remover los terminos sparse
tdm2 <- removeSparseTerms(tdm, sparse = 0.95)
m2 <- as.matrix(tdm2)
# cluster terms jeraquico
m <- as.matrix(m2)
d<- dist(m2)
deuclidea<-dist(m2, method = "euclidean")
d2<-dist(m2, method = "manhattan")
distcos<-dissimilarity(x=m2,method='cosine')
distjaccard<-dissimilarity(x=m2,method='jaccard')
groups <- hclust(distjaccard,method="complete")
groups <- hclust(deuclidea,method="complete")
groups <- hclust(distcos,method="complete")
```

```
#plot dendograma
plot(groups, hang=-1)
rect.hclust(groups,5)
```

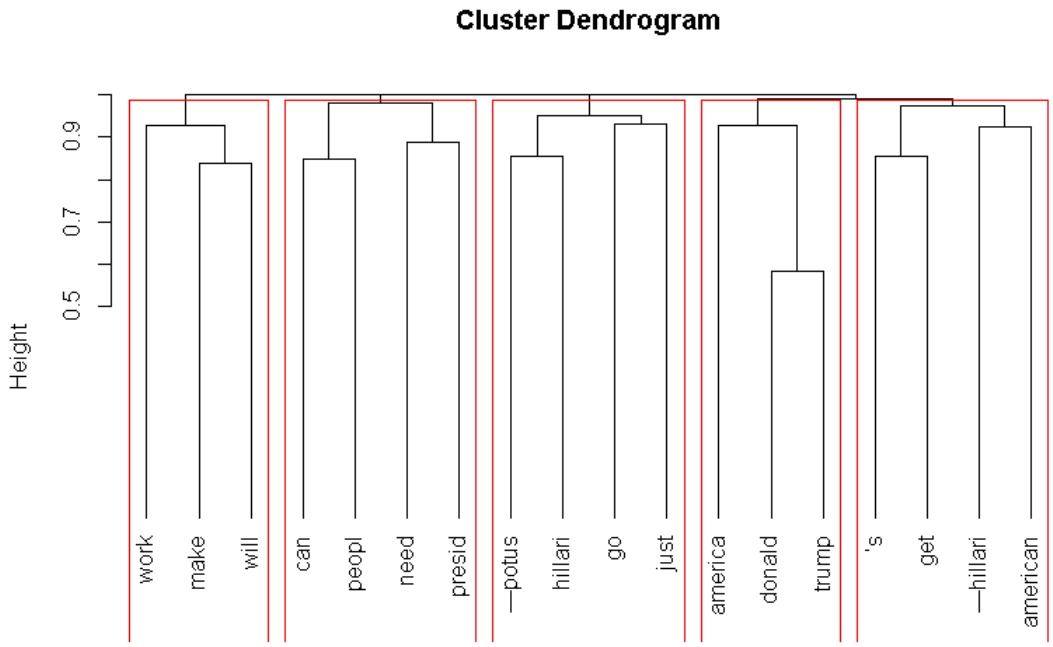
Obteniendo la siguiente representación del mismo, mediante un dendograma, en el cual podemos observar los grupos que se forman.

Distancia Euclídea:



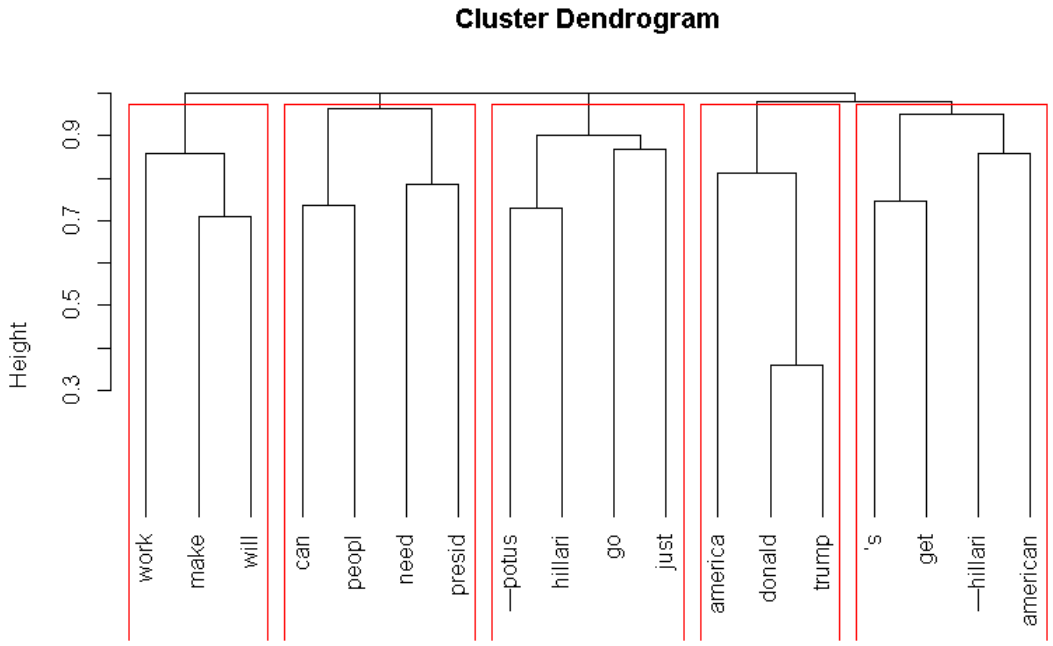
Podemos observar que se generan 5 clusters, donde en uno de ellos se encuentran mas del 90% de los términos, además hay dos grupos que contienen un solo término lo cual no deja mucha información sobre los grupos que se forman, por lo que podriamos concluir que esta distancia no es la más idónea para la visualización de los mismos.

Distancia Jaccard:



Haciendo uso de esta distancia se pueden observar que se forman 5 clusters, además haciendo uso de esta distancia los términos están bien representados por medio de los clusters a los cuales estos pertenecen.

Distancia Coseno:

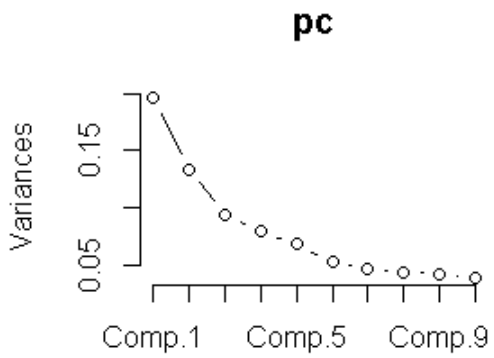


Haciendo uso de esta distancia se pueden observar que se forman 5 clusters además haciendo uso de esta distancia los términos están bien representados por medio de los clusters a los cuales estos pertenecen.

De los tres cluster podemos concluir que el uso de las distancias ***coseno y Jaccard*** genera una mejor reprentación de los grupo superando de esta manera el uso de la distancia ***Euclídea*** la cual no generó una buena representación de los clusters.

6.4.2 K-medias cuenta Clinton

Para la realización del método de K-medias previamente se realizó un análisis de componenentes principales para determinar que términos estan correlacionados y para la selección del término K.



Podemos escoger como un valor optimo para realizar K-medias, $k = 5$.

```
pc <- princomp(tdm)
plot(pc)
data<-tdm
wss <- (nrow(data)-1)*sum(apply(data,2,var))
for (i in 2:15) wss[i] <- sum(kmeans(data,
centers=i)$withinss)
plot(1:15, wss, type="b", xlab="Number of Clusters",
ylab="Within groups sum of squares")



m3 <- t(tdm2) # transpuesta de la matriz terminos documentos
set.seed(1222) # definimos una semilla para  poder volver
# a usar en un futuro valores aleatorios generados
k <- 5# number of clusters
kmeansResult <- kmeans(m3, k)
km<-round(kmeansResult$centers, digits = 3)
#guardamos lo anterior en un csv
write.csv(km,file="kmeansclinton.csv")
```

```
#ahora  con el siguiente loop podemos ver los cluster


for (i in 1:k) {
cat(paste("cluster ", i, ": ", sep = ""))
s <- sort(kmeansResult$centers[i, ], decreasing = T)
cat(names(s)[1:5], "\n")
}
# imprime los terminos  para cada cluster

write.csv(s,file="clusterkm.csv")
```

Obteniendo como resultado los siguientes clusters

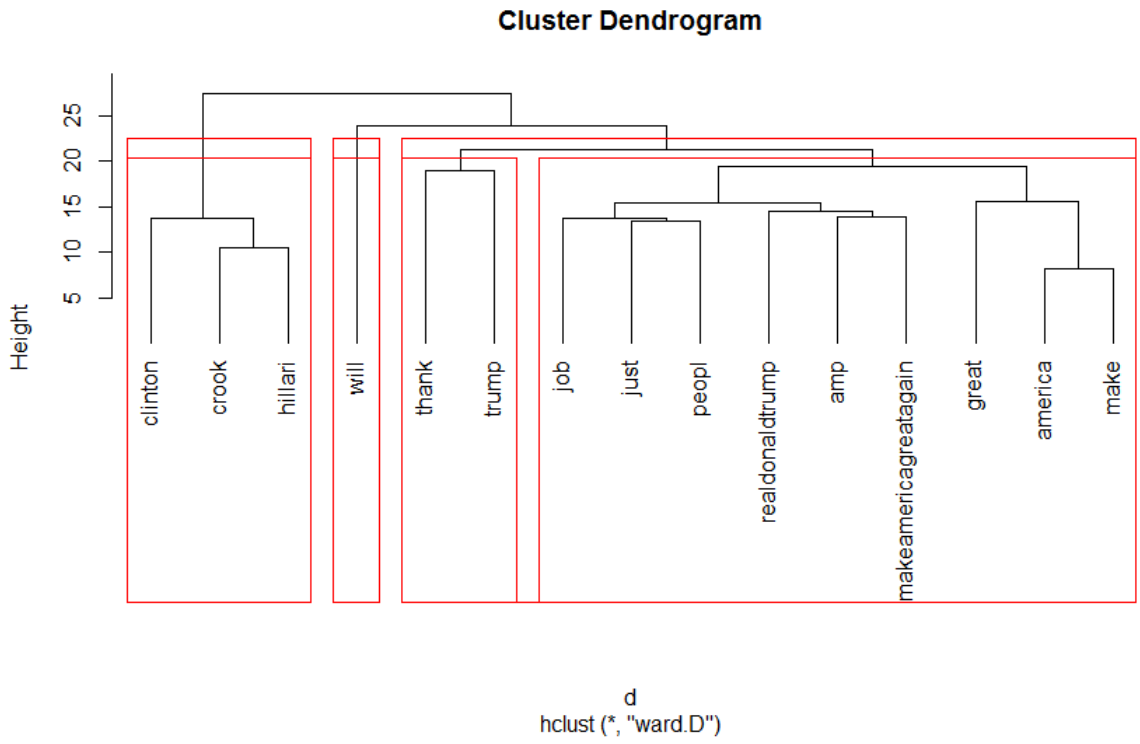
```
cluster 1: trump donald –hillari we presid
cluster 2: becom compani crimin foreign leader
cluster 3: hillari clinton –potus i get
cluster 4: will can make we presid
cluster 5: look everi like take matter
```

Se puede concluir que estos términos se pueden proponer como los temas que más se comentaron en los tweets recolectados.

6.4.3 Cluster jerárquico para la cuenta Trump

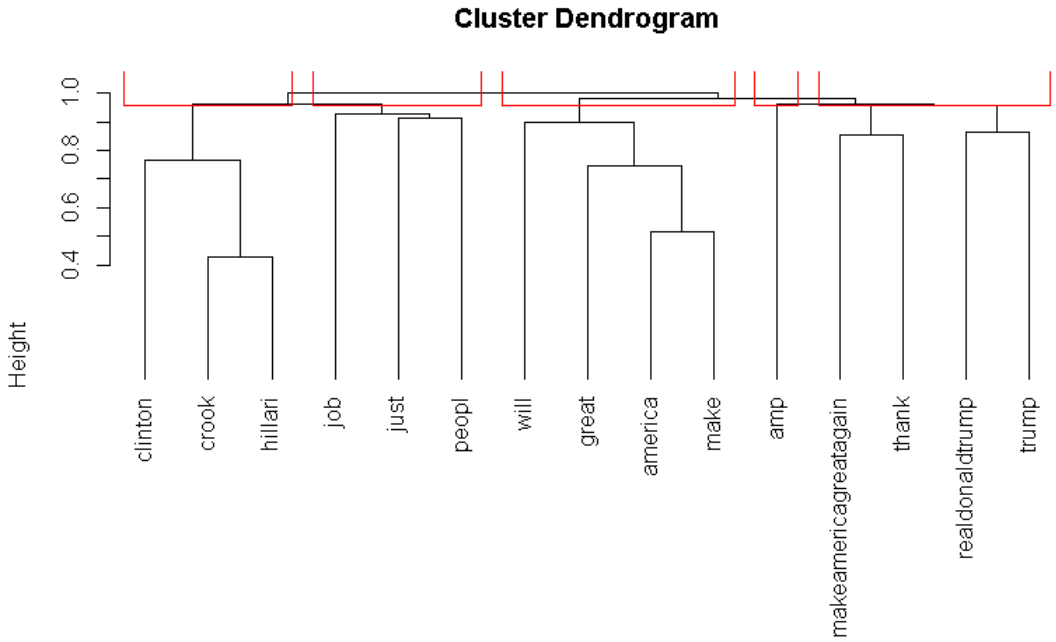
```
# remover los terminos sparse
tdm2 <- removeSparseTerms(tdm, sparse = 0.95)
m2 <- as.matrix(tdm2)
# cluster terms jeraquico
m <- as.matrix(m2)
d<- dist(m2)
deuclidea<-dist(m2, method = "euclidean")
d2<-dist(m2, method = "manhattan")
distcos<-dissimilarity(x=m2,method='cosine')
distjaccard<-dissimilarity(x=m2,method='jaccard')
groups <- hclust(distjaccard,method="complete")
groups <- hclust(deuclidea,method="complete")
groups <- hclust(distcos,method="complete")
#plot dendograma
plot(groups, hang=-1)

rect.hclust(groups,5)
```

Obteniendo como resultado los siguientes clusters:
Distancia Euclídea:



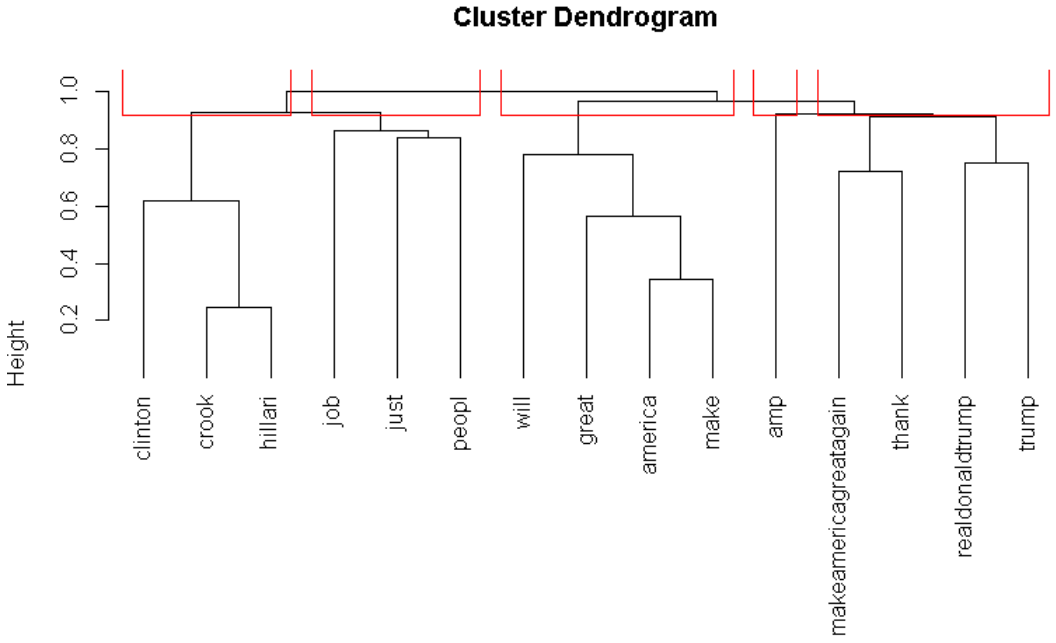
Podemos observar que se generan 5 clusters, donde en uno de ellos se encuentran mas del 90% de los términos, además hay dos grupos que contienen un solo término lo cual no deja mucha información sobre los grupos que se forman por lo que podriamos concluir que esta distancia no es la más idónea para la visualización de los mismos.

Distancia Jaccard:



Haciendo uso de esta distancia se pueden observar que se forman 5 clusters, además haciendo uso de esta distancia los términos están bien representados por medio de los clusters a los cuales estos pertenecen.

Distancia Coseno:

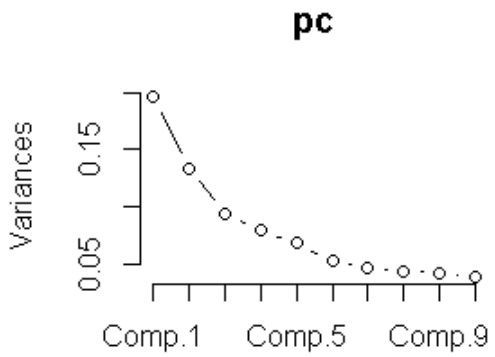


Haciendo uso de esta distancia se pueden observar que se forman 5 clusters además haciendo uso de esta distancia los términos están bien representados por medio de los clusters a los cuales estos pertenecen.

De los tres cluster podemos concluir que el uso de las distancias ***coseno y Jaccard*** genera una mejor reprentación de los grupo superando de esta manera el uso de la distancia ***Euclídea*** la cual no generó una buena representación de los clusters.

6.4.4 K-medias cuenta Trump

De igual manera que el análisis de cluster anterior via K-medias, se realizó un análisis de componenentes principales para determinar que términos estan correlacionados y para la selección del término K.



Podemos escoger como un valor optimo para realizar K-medias, $k = 5$.

```
m3 <- t(tdm2) # transpuesta de la matriz terminos documnetos
set.seed(1222) # definimos una semilla para  poder volver
# a usar en un futuro valores aleatorios generados
k <- 5# number of clusters
kmeansResult <- kmeans(m3, k)
km<-round(kmeansResult$centers, digits = 3)
#guardamos lo anterior en un csv
write.csv(km,file="kmeanstrump.csv")

#ahora  con el siguiente loop podemos ver los cluster


for (i in 1:k) {
cat(paste("cluster ", i, ": ", sep = ""))
s <- sort(kmeansResult$centers[i, ], decreasing = T)
cat(names(s)[1:5], "\n")
}
# imprime los terminos  para cada cluster

```

```
write.csv(s,file="clusterkmt.csv")
```

Obteniendo como resultado los siguientes clusters:

```
## cluster 1: great thank will peopl just
## cluster 2: will hillari crook trump thank
## cluster 3: great make america will thank
## cluster 4: hillari trump thank crook clinton
## cluster 5: amp will hillari trump thank
```

Se puede concluir que estos se pueden proponer como los temas que más se comentaron en los tweets recolectados.

6.5 Clasificadores

El siguiente código es la implementación de los metodos de clasificación de Naive Bayes y SVM para textos, el problema a resolver es clasificar un conjunto de tweets en dos posibles categorías para poder identificar quien publicó cada tweet, estas categorías son **"Trump"** y **"Clinton"**. Por su parte se necesitó dividir el conjunto de datos en dos, un conjunto con el 60% de la data para entrenar el modelo y otro con el resto (40%) para realizar el test y obtener los resultados. A continuación su implementación en R:

6.5.1 Clasificador Naive Bayes

Se implemento el siguiente código para este clasificador. Para ello se necesitó dividir el set de datos dos conjuntos uno de entrenamiento y otro para la prueba del mismo. Posterior a eso se procedió en llevar a cabo la etapa de preprocesamiento para poder generar las matrices términos-documnetos. Una vez obtenida estas matrices se implementó el algoritmo de Naive Bayes.

```
#Clasificador  Naive Bayes  Identificador de Tweets
#Cargamos el set de datos
clinton<-clintontweets
trump1<-tweetst.df
trump2<-tweetst.df[1:566,]
clin2<-clinton[1:566,]
trump3<-tweetst.df[567:755,]
clint3<-clinton[567:755,]

#Generamos un conjunto de entrenamiento y otro de prueba  para el
   modelo
training<-rbind(trump2,clin2)
test<-rbind(trump3,clint3)


#Le agregamos a cada uno de los conjuntos , una nueva clase , al
   conjunto de datos de los
#tweets de la cuenta de trump le agregamos la  variable "trump",
   respectivamente a los
```

```
#tweets de clinton le agregamos "clinton"


trump2["class"]<-rep("trump",nrow(trump2))
clin2["class"]<-rep("clinton",nrow(clin2))

#Limpieza del Texto Pre-Procesamiento
#Construyamos el Corpus indicando que la fuente es un vector de
   caracteres
myCorpus<-Corpus(VectorSource(clin2$text))

#El texto del corpus  se convierte en texto plano
mycorpus <- tm_map(myCorpus, PlainTextDocument)

#El texto del corpus  se convierte en texto plano
mycorpus <- tm_map(myCorpus, PlainTextDocument)
#Se remueven los signos de puntuacion
myCorpus <- tm_map(myCorpus, removePunctuation)
# Se remueven los numeros
myCorpus <- tm_map(myCorpus, removeNumbers)
# Se remueven los  URLs
removeURL <- function(x) gsub("http[[:alnum:]]*", "", x)
myCorpus <- tm_map(myCorpus, removeURL)
#El texto del corpus  se convierte en  minusculas
myCorpus <- tm_map(myCorpus, tolower)
#El texto del corpus  se convierte en texto plano
mycorpus <- tm_map(myCorpus, PlainTextDocument)
#Se remueven las  stop words
myStopwords <- c(stopwords("english"))
myStopwords <- c(stopwords("english"), "the", "to","of","on","that","in
   ","in","for","a")
myCorpus <- tm_map(myCorpus, removeWords, myStopwords)
#Guardamos una copia del  corpus
myCorpusCopy <- myCorpus
# Aplicamos el proceso de steaming
myCorpus <- tm_map(myCorpus, stemDocument)




myCorpus2<-Corpus(VectorSource(trump2$text))

#El texto del corpus  se convierte en texto plano
mycorpus2<- tm_map(myCorpus2, PlainTextDocument)

#El texto del corpus  se convierte en texto plano
mycorpus2 <- tm_map(myCorpus2, PlainTextDocument)
#Se remueven los signos de puntuacion
myCorpus2 <- tm_map(myCorpus2, removePunctuation)
# Se remueven los numeros
myCorpus2 <- tm_map(myCorpus2, removeNumbers)
```

```
# Se remueven los  URLs
removeURL <- function(x) gsub("http[[:alnum:]]*", "", x)
myCorpus2 <- tm_map(myCorpus2, removeURL)
#El texto del corpus  se convierte en  minusculas
myCorpus2<- tm_map(myCorpus2, tolower)
#El texto del corpus  se convierte en texto plano
mycorpus2 <- tm_map(myCorpus2, PlainTextDocument)
#Se remueven las  stop words
myStopwords <- c(stopwords("english"))
myStopwords <- c(stopwords("english"), "the", "to","of","on","that","in
    ","in","for","a")
myCorpus2 <- tm_map(myCorpus2, removeWords, myStopwords)
#Guardamos una copia del  corpus
myCorpusCopy2 <- myCorpus2
# Aplicamos el proceso de steaming
myCorpus2<- tm_map(myCorpus2, stemDocument)




myCorpus3<-Corpus(VectorSource(test$text))

#El texto del corpus  se convierte en texto plano
mycorpus3<- tm_map(myCorpus3, PlainTextDocument)

#El texto del corpus  se convierte en texto plano
mycorpus3 <- tm_map(myCorpus3, PlainTextDocument)
#Se remueven los signos de puntuacion
myCorpus3 <- tm_map(myCorpus3, removePunctuation)
# Se remueven los numeros
myCorpus3 <- tm_map(myCorpus3, removeNumbers)
# Se remueven los  URLs
removeURL <- function(x) gsub("http[[:alnum:]]*", "", x)
myCorpus3 <- tm_map(myCorpus3, removeURL)
#El texto del corpus  se convierte en  minusculas
myCorpus3<- tm_map(myCorpus3, tolower)
#El texto del corpus  se convierte en texto plano
mycorpus3 <- tm_map(myCorpus3, PlainTextDocument)
#Se remueven las  stop words
myStopwords <- c(stopwords("english"))
myStopwords <- c(stopwords("english"), "the", "to","of","on","that","in
    ","in","for","a")
myCorpus3 <- tm_map(myCorpus3, removeWords, myStopwords)
#Guardamos una copia del  corpus
myCorpusCopy3 <- myCorpus3
# Aplicamos el proceso de steaming
myCorpus3<- tm_map(myCorpus3, stemDocument)




```

```
115
116 #Generamos la matriz T-D
117
118 tmatrix <- t(TermDocumentMatrix(myCorpus,control = list(wordLengths=c
    (4,Inf))));
119 cmatrix <- t(TermDocumentMatrix(yCorpus2,control = list(wordLengths=c
    (4,Inf))));
120 testmatrix <- t(TermDocumentMatrix(yCorpus3,control = list(wordLengths=
    c(4,Inf))));
121
122 # ahora se tiene que construir el modelo, para ello  se tiene que
    calcular las
123 #probabilidades  del modelo, contar el numero de apariciones de cada
    palabra, anadir  la
124 #estimacion de
125 #laplace, posterior calcular el log de las probabilidades y guardar en
    un
126 #archivo csv.
127 #
128 probabilityMatrix <-function(docMatrix)
129 {
130 # Sumar las frecuencias
131 termSums<-cbind(colnames(as.matrix(docMatrix)),as.numeric(colSums(as.
    matrix(docMatrix))))
132 # sumar 1 "laplace"
133 termSums<-cbind(termSums,as.numeric(termSums[,2])+1)
134 # calcular las probabilidades
135 termSums<-cbind(termSums,(as.numeric(termSums[,3])/sum(as.numeric(
    termSums[,3]))))
136 # Calcular el log natural de las probabilidades
137 termSums<-cbind(termSums,log(as.numeric(termSums[,4])))
138 # Adicionar nombres a las columnas
139 colnames(termSums)<-c("term","count","additive","probability","
    lnProbability")
140 termSums
141 }
142
143
144
145 #hacemos el llamado
146 tp<-probabilityMatrix(tmatrix)
147 cp<-probabilityMatrix(cmatrix)
148
149 #guardamos en un archivo csv
150
151
152 write.csv(file="trumpprobmatrix.csv",tp)
153 write.csv(file="clintonprobmatrix.csv",cp)
154
155 #ahora se usa Bayes y se prueba el modelo con el conjunto de test
156 #se quiere comparar los tweets del conjunto de entrenamiento  con las
```

```
    dos matrices de
#probabilidades cada tweets se compara con las matrices , queremos
    saber cuantas palabras no #aparecen , si esto pasa se le agrega  la
     estimacion de laplace  , luego obtenemos las suma  de las #
    probailidades de las palabras que si aparecen .

getProbability <- function(testChars,probabilityMatrix)
{
charactersFound<-probabilityMatrix[probabilityMatrix[,1] %in% testChars
    ,"term"]
# cuenta cuantas palabras aprecen el los matriz
charactersNotFound<-length(testChars)-length(charactersFound)
# agregamos las  probabilidades normalizadas de las palabras que si
    fueron encontradas
charactersFoundSum<-sum(as.numeric(probabilityMatrix[probabilityMatrix
    [,1] %in% testChars,"lnProbability"]))
# usamosln(1/total de las palabras con estimacion de laplace)  para
    palbras no encontradas
charactersNotFoundSum<-charactersNotFound*log(1/sum(as.numeric(
    probabilityMatrix[,"additive"])))
#esta es la probailidad
prob<-charactersFoundSum+charactersNotFoundSum
prob
}

#La funcion anterior se usa para cada tweet, para ello se genera un
    loop

# obtenemos la matriz
testmatrix<-as.matrix(testmatrix)

classified<-NULL

for(documentNumber in 1:nrow(testmatrix))
{

tweets.test.chars<-names(testmatrix[documentNumber,testmatrix[
    documentNumber,] %in% 1])
# Obtenemos las probabilidades
trumpprob <- getProbability(tweets.test.chars,tp)
clintonprob <- getProbability(tweets.test.chars,cp)
# Add it to the classification list
classified<-c(classified,ifelse(trumpprob>clintonprob,"trump","clinton"
    ))
}

#visualizamos los resultados del clasificador
View(cbind(classified,test$test))
```

De los 604 documentos(tweets) que tiene el conjunto de prueba, donde 302 tweets le pertenecen a la cuenta de Clinton y 302 a la de Trump, al implementar el clasificador se obtuvieron los

siguientes resultados

$Tweet$	$Clinton$	$Trump$
Clinton	273	29
Trump	29	302

Tabla 6.1: Resultados.

Podemos ver que 331 tweets fueron etiquetados como Trump y 273 como Clinton, de los 331 que se etiquetaron como Trump, 29 pertenecen a la clase Clinton por lo que estan mal etiquetados. El clasificador obtuvo un 90% de precisión. Para mejorar la precisión del clasificador se necesitaría ir recolectando muchos mas tweets para agregarlos al conjunto de entrenamiento del clasificador. Como la descarga de data de twitter tiene restricciones en la descarga diaria de los tweets, es útil la función creada basada en el paquete SchedulerR explicada anteriormente, fijando mínimo un período de un mes o dos para la recoleción de los mismos.

6.5.2 Clasificador SVM

A continuación su implementación en R, además se hace uso de los tres kernels explicados en el capítulo anterior. El motivo de realizar el algoritmo de **SVM** haciendo uso de los diferentes kernels es para poder escoger el que obtenga una mejor precisión.

```
library(RTextTools)
library(e1071)
library(tm)

clinton<-clintontweets
trump1<-tweetst.df
trump2<-tweetst.df[1:453,]
clin2<-clinton[1:453,]
trump3<-tweetst.df[454:755,]
clint3<-clinton[454:755,]

#Generamos un conjunto de entrenamiento y otro de prueba  para el
   modelo
training<-rbind(trump2,clin2)
training <- training[sample(1:906,size = 906),]
test<-rbind(trump3,clint3)


#Le agregamos a cada uno de los conjuntos , una nueva clase , al
   conjunto de datos de los #tweets de la cuenta de trump le agregamos
    la  variable "trump", respectivamente a los #tweets de clinton le
   agregamos "clinton"


trump2["class"]<-rep("trump",nrow(trump2))
clin2["class"]<-rep("clinton",nrow(clin2))

```

```
#ahora implmentaremos  svm
#matriz doc-Ter
matrix= create_matrix(training[,1], language="english", removeStopwords
   =FALSE, removeNumbers=TRUE, stemWords=FALSE)

#se genera el training set y el test set de
container = create_container(matrix, as.numeric(as.factor(training
   [,20])),trainSize=1:600, testSize=601:906,virgin=FALSE)
#aplicacion de svm usando los diferentes kernels
SVM <- train_model(container,"SVM",kernel = "sigmoid")
SVM <- train_model(container,"SVM",kernel = "linear")
SVM <- train_model(container,"SVM",kernel = "polynomial")
SVM_CLASSIFY <- classify_model(container, SVM)
#Ahora se observan el resumen de resultados
table(training$class[601:906],SVM_CLASSIFY$SVM_LABEL)
```

Kernel Gaussiano

Para la clasificación de 302 tweets se obtuvieron los siguientes resultados

$Tweet$	$Clinton$	$Trump$
Clinton	143	12
Trump	20	131

Tabla 6.2: Resultados.

De los 302 tweets los cuales fueron seleccionados aleatoriamente, 155 fueron etiquetados como Clinton de los cuales 12 fueron mal etiquetados. Por su parte 151 fueron etiquetados como Trump de los cuales 20 fueron mal etiquetados.
Donde la precisión del clasificador fue de un 90%.

Kernel lineal

Para la clasificación de 302 tweets se obtuvieron los siguientes resultados

$Tweet$	$Clinton$	$Trump$
Clinton	145	10
Trump	13	138

Tabla 6.3: Resultados.

De los 302 tweets los cuales fueron seleccionados aleatoriamente, 155 fueron etiquetados como Clinton de los cuales 10 fueron mal etiquetados. Por su parte 151 fueron etiquetados como Trump de los cuales 13 fueron mal etiquetados. Donde la precisión del clasificador fue de un 93%.

Kernel polinomial

Para la clasificación de 302 tweets se obtuvieron los siguientes resultados De los 302 tweets los

$Tweet$	$Clinton$	$Trump$
Clinton	120	35
Trump	41	110

Tabla 6.4: Resultados.

cuales fueron seleccionados aleatoriamente, 155 fueron etiquetados como Clinton de los cuales 35 fueron mal etiquetados. Por su parte 151 fueron etiquetados como Trump de los cuales 41 fueron mal etiquetados. Donde la precisión del clasificador fue de un 80%.

Se puede observar que los kernels que mejor se ajustan : gaussiano y lineal, ya que son más precisos al momento de clasificar, mientras que el kernel polinomial tiene un nivel bajo de precisión. Se puede concluir que los kernels gaussiano y lineal son los que mejor se ajustan al conjunto de datos.

Comparando ahora los resultados dados por los kernels gaussiano y lineal con el obtenido con Naive Bayes, se puede observar que los tres som muy efectivos y hay poca diferencia pero la implementación de **SVM** con el kernel lineal fue de los tres el que obtuvó una mejor precisión la cual fue de 93%.

6.6 Aplicación Web

Haciendo uso del paquete **Shiny** se implementó una aplicación web basada en los distintos modelos de clasificación especialmente el método estudiado en el capítulo anterior de **SVM**.

La aplicación consiste en un **IRC** (Internet Relay Chat) es un protocolo de comunicación en tiempo real basado en texto, que permite debates entre dos o más personas. El objetivo de esta aplicación es clasificar el texto en el chat para poder detectar si algún usuario esta usando "malas palabras" en sus mensajes.

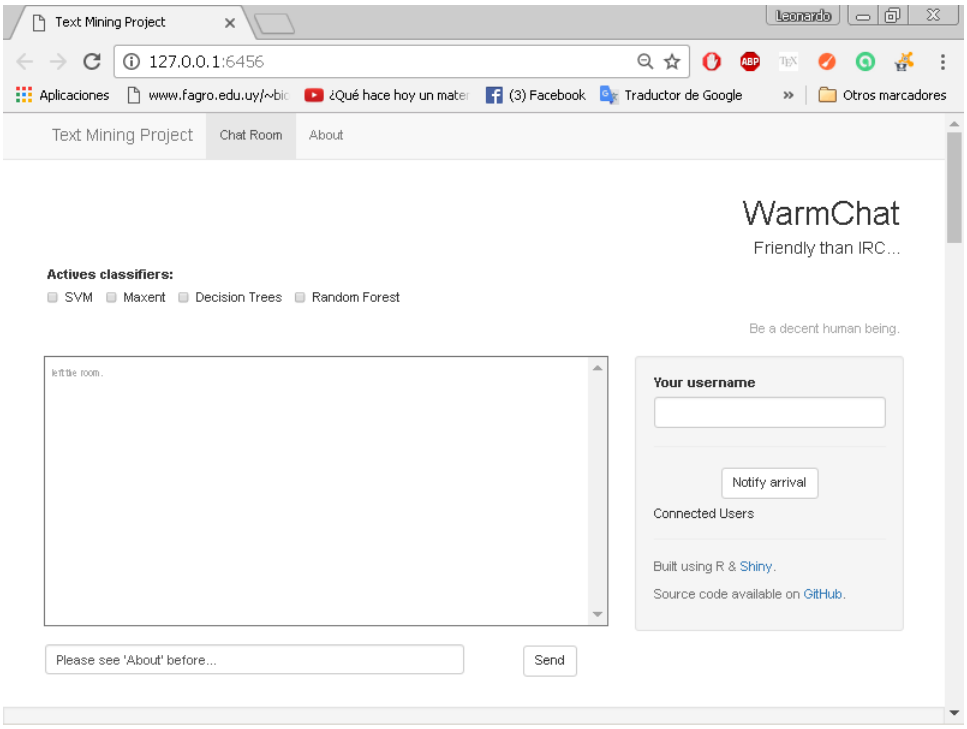
Recomendaciones de uso

- Para que la aplicación detecte los insultos, previamente hay que elegir un clasificador en el panel de control.
- El IRC está dirigido al idioma ingles.
- La aplicación no entiende el sarcasmo.
- Para que los modelos se implementen se necesita un mínimo de 4 palabras.

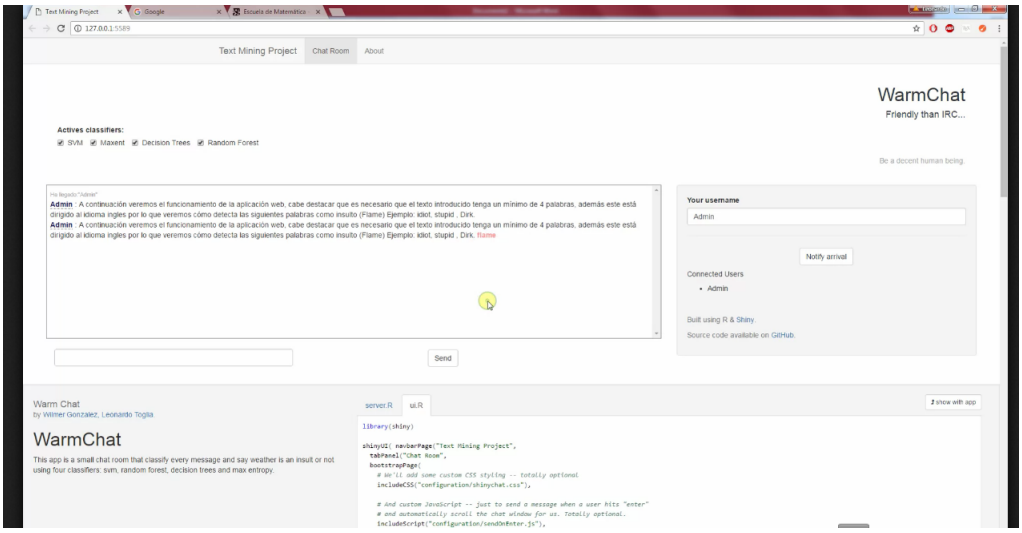
Funcionamiento

- Se aplica el Pre-procesamiento al texto introducido en el IRC.

- Se utilizó el conjunto de entrenamiento para los modelos de la pagina web: `https://www.kaggle.com/c/detecting-insults-in-social-commentary`
- Se implementaron cada uno de los modelos haciendo uso de los paquetes de R.



en esta imagen se puede observar como la aplicación esta clasificando un texto.



La aplicación fue desarrollada en conjunto con el Br Wilmer González, como parte del proyecto del curso de minería de texto dictado por la tutora la Dra Carenne Ludeña.

Todos los códigos de los análisis y la aplicación se encuentran en un repositorio en Github. Link del repositorio:
`https://github.com/LTogliaUCV/Codes_Twitter_Analysis.git`.

Conclusiones

La metodología propuesta permite estudiar y encontrar información en un conjunto de documentos. Además las herramientas matemáticas permiten estudiar la información contenida en los textos así como su agrupamiento y clasificación usando algoritmos de aprendizaje estadístico sobre las matrices de documentos y términos.

Se pudo concluir que la automatización del proceso de recolección de datos es muy útil para poder generar conjunto de datos más grandes por lo que la función desarrollada en este trabajo puede ser útil para trabajos a futuro.

La metodología implementada muestra la importancia de utilizar varias distancias en el caso de clusterización jerárquica además de que la implementación de la técnica de ACP permite reducir la dimensionalidad del conjunto de datos lo cual hace al algoritmo de k-medias mas eficiente pues se obtienen mejores cluster.

Para los trabajos a futuro se recomienda lo siguiente:

- Implementar la metodología en un ámbito de **Big data.**
- Implementar en R las herramientas matemáticas que aún no se encuentren desarrolladas.
- Encontrar topologías adecuadas sobre conjunto de documentos.

Bibliografía

[1] AGGARZAWAL, C, XHIANG ZHAI, Z, *Mining text data*, Springer,2012.

[2] ATKINS, C, OSTLER, *Corpus and Design Criteria*, Literary and Linguistic Computing, 1992

[3] ASHOK, S, MENHARN, S, *Text mining classidfication clustering and applications*, CRC PRESS,2009.

[4] BERRY, M, CASTELLANOS, M, *Survey of text mining II*, Springer, 2008.

[5] BERRY, M, KOGAN, J, *Text mining applications and theory*, Wiley, 2010

[6] BIEMANN, C, MEHLER, A, *Text mining from ontology learning to automated text processing applications*, Springer, 2014.

[7] CORTES, C, VAPNIK,V.Ñ, *Support vector networks, Machine learning*, vol.20, pp 273-297, 1995.

[8] CRISTIANINI, N SHAWE-TAYLOR, J *Support vector machines and other kernel-based learning methods*. Cambridge University Press, Cambridbe MA, 2000.

[9] EÍTO, R, SENSO, J.A, *Mineria Textual* , 2014.

[10] FELDMAN, R, SANGER, J, *Text mining Handbook*, Cambridge University Press, 2007.

[11] GUPTA, V, GRURPREET, S, LEHAL, *A survey of text mining techniques and applications*, Department of Computer Science, Pnjab University Chandirgerb, India.

[12] HEARST, M , *Untagling Text Data Mining*, School of information and Management and System, University of Berkeley.

[13] HIROSHI, S, *Naive Bayes and text classification*, 10-02-2015

[14] HUANG AMA, *Similiraty meauseres for text document clustering*, Department of Computer Science, The university of Waikato, Nueva Zelanda.

[15] KAO, A, POTEET, S, *Natural language processing and text mining* , Springer, 2007.

[16] MEHLER, A, *Text mining from ontology learning to automated text processing applications*, Springer, 2014.

[17] MINER,G, DELEN, D,ELDER, J *Practical text mining and statiscal analysis for Non-structured data text data applications*, Elsevier, 2012.

[18] ROSARIO,B, *Latent semantic indexing : An overview*, Springer, 2000.

[19] SANTALLA DEL RIO, M.P, *La elaboración de corpus linguísticos*, Universidad Santiago de Compostela, 2005.

[20] STEINBACH, M, KARYPIS, G, KUMAR,V, *Acomparison of clustering techniques*, Department of Computer Science , University of Minnesota.

[21] STEINBACH, M, PANG, N.T,KUMAR, V,*Introduction to data mining* , capítulo 8 , Pearson,2005.

[22] SHLENS, J, *A tutorial on Principal Components Analysis*, Google researh, 2014.

[23] SHUTZE , H, MANNING, C, *Foundations of statistical natural language processing*, MIT Press 2000.

[24] SMITH, L, *A tutorial on Principal Components Analysis*, 2002.

[25] SULLIVAN , D, *Document warehousing and text mining*, Wiley Computer Publishing.

[26] TOGNINI-BONELLI, E ,*Corpus Theory and practice*, Birmingham:twc , 1996.

[27] VAPNIK, V.Ñ, *Statistical learning theory*, New York: Wiley, 1998.

[28] VIJAYARANI,S , ITALMATHI, I, NITHYA, textitPreprocessing Techniques for text mining, An overview, International Journal of Computer Science and Communication Networks, 2010.

[29] WAGSTAFF, K, CARDIE, C, *Constrained K-means, clustering with Background Knowledge*, Department of Computer Science Corneel University, NY,Eighteenth International Conference on Machine Learning, págs. 577-584 , 2001.

[30] YANCHANG, Z,*R and data mining: Examples and cases of studies*, 2013.

[31] YANCHANG, Z, YONGHUA, C *Data mining applications with R*, Elsevier, 2014.

[32] YANCHANG, Z, *Text mining with R and analysis of twitter Data*, 2014.