INSTITUTO TECNOLÓGICO DE BUENOS AIRES – ITBA

ESCUELA DE (INGENIERÍA Y TECNOLOGÍA – INGENIERÍA Y GESTIÓN - POSTGRADO)

DETECCIÓN DE TÓPICOS

Utilizando el Modelo LDA

AUTOR/ES: Hammoe, Luciano (Leg. N° 103995)

DOCENTE/S TITULAR/ES O TUTOR/ES: Arjones, Gustavo

TRABAJO FINAL PRESENTADO PARA LA OBTENCIÓN DEL TÍTULO DE ESPECIALISTA EN CIENCIA DE DATOS

BUENOS AIRES

SEGUNDO CUATRIMESTRE, 2018

Introducción

Con la explosión de la redes sociales, se ha generado un nuevo universo donde las personas comparten información de lo que les gusta y lo que no, de lo que hacen, de lo que usan (sea un producto o un servicio) y muchas otras cuestiones.
Esta información resulta de muchísimo valor para las grandes empresas ya que les permite adoptar diversas estrategias para captar nuevos clientes, para mantener la fidelidad de aquellos que ya son clientes, para ofrecer nuevos productos, servicios o promociones.

Es muy interesante tener en cuenta también que los usuarios de redes sociales, suelen escribir en las mismas mencionando a una empresa o marca, cuando algo no anda bien, es decir, para quejarse al respecto de un servicio o producto. Por el contrario es poco común que un usuario escriba para expresar un comentario positivo sobre dicha empresa o marca.
Con lo dicho en el párrafo anterior, intento explicar que hoy en día las redes sociales son un gran termómetro que mide si las empresas están brindando productos y servicios de calidad y eso impacta fuertemente en la decisión de compra de un consumidor (tanto para alguien que ya es cliente de esa empresa como para un potencial nuevo cliente).

Existen diversos estudios que indican que la intención de compra no se logra por el sólo hecho de que la compañía o marca hable bien de sus servicios / productos. A esto se lo conoce como OSM (owned social media). OSM puede lograr lo que se llama “brand awareness” y “customer satisfaction”, es decir, el conocimiento de la marca y satisfacción del cliente pero no puede lograr generar intención de compra (“purchase intent”).
La mejor estrategia para aumentar la intención de compra de un consumidor es lograr que los consumidores de dicha marca se mantengan satisfechos. Esto se logra con una buena atención post-venta, atendiendo sus reclamos y problemas y reforzando el valor de la compañía.
Los potenciales consumidores de una marca suelen observar a consumidores actuales de la misma para decidir la compra de un producto o servicio. Por lo tanto aquellos consumidores satisfechos de una marca van a generar comentarios positivos en las redes, aumentando por lo tanto la intención de compra de potenciales consumidores[1].

A partir de lo enunciado anteriormente existe una fuerte necesidad de las compañías de poder analizar de manera simple y eficiente los comentarios que los usuarios de redes sociales realizan sobre sus productos con el fin de mantener satisfechos a aquellos consumidores que ya son clientes de la compañía, y a su vez, captar potenciales clientes a través de los comentarios positivos en las redes y de la exposición de las marcas en las mismas.

El objetivo de este trabajo es lograr una herramienta que permita procesar grandes volúmenes de comentarios sobre una compañía en la red social Twitter y permita realizar lo que se llama “Topic Detection”, es decir, detectar cuáles son los temas o tópicos sobre lo que más se habla

[1] https://phys.org/news/2017-09-social-media-consumers-big-brands.html

a cerca de una compañía. Esto permite a dicha compañía entender de una manera simple qué concepto tienen los usuarios de la red social (sean o no clientes de la marca) sobre si misma, y sobre los productos / servicios de sus marcas.
En función de estos tópicos, la empresa podrá adoptar estrategias de acuerdo a los objetivos que se quieran lograr.

Para lograr la detección de tópicos, la idea principal es analizar una gran cantidad de tweets sobre diversas compañías de telecomunicaciones utilizando un modelo generativo del aprendizaje automático (machine learning) llamado LDA (Latent Dirchlet Allocation)[2].

[2] https://es.wikipedia.org/wiki/Latent_Dirichlet_Allocation

Estado de la Cuestión

La detección de tópicos es un tema conocido y estudiado en el mundo de Big Data. Existen diversos trabajos relacionados al tema en cuestión. Muchos difieren en los algoritmos que utilizan para obtener sus resultados. Otros analizan el tema “Topic Detection” desde diferentes puntos de vista.

A continuación se mencionarán algunos trabajos referidos al tema, que me parecen interesantes para entender la situación actual y el estado de la cuestión.
Estos trabajos, se basan en los algoritmos aplicados para detectar tópicos, en la comparación de dichos algoritmos, o también, en la aplicación y comparación de los algoritmos sobre documentos de texto y tweets.
Como he comentado hay bastante trabajo realizado sobre el tema por lo tanto intentaré ilustrar su estado actual de la forma más simple.

Topic Detection / Modelling

Risch, 2016, p. 11, presenta un trabajo sobre “Topic Modelling” o “Topic Detection”, que intenta explicar cómo se comporta el método LDA (Latent Dirichlet Allocation) aplicado sobre un stream de tweets, observando los pros y contras del método.
Topic Modelling, básicamente es identificar tópicos o temas en textos. Generalmente esto se logra detectando patrones en una colección de documentos llamado “corpus” y agrupando las palabras que aparecen en dichos documentos en tópicos. Los tópicos son generalmente definidos como una función de densidad pi(w) donde w es una palabra y pi es la función de densidad para el tópico i. Por lo tanto, pi(w) es la probabilidad de la palabra w dado el tópico i.
Se define como Topic Modelling generativo a un modelo que permite la clasificación de documentos aún no procesados una vez que el modelo fue entrenado sin pasar por el corpus completo. Esto último es un atributo muy importante para un modelo que se ejecuta con un stream de documentos (que es lo mismo que un corpus infinito).

LDA (Latent Dirichlet Allocation)

Por otro lado, Risch, 2016, p. 12, define en qué consiste el método LDA.
Latent Dirichlet Allocation es un modelo de tópicos generativo. Este modelo asume que cada palabra en un documento es generada a partir de un tópico que es tomado de una distribución de tópicos para cada documento. La distribución en sí de tópicos para cada documento, es generada a partir de una distribución de Dirichlet lo que significa que LDA permite que un documento sea parte de varios tópicos cada uno con un peso diferente.
Blei, Ng, & Jordan (2003) definen que en el modelo LDA cada documento puede ser visto como una mezcla de varios tópicos donde cada documento se considera que puede tener un set de tópicos asignados mediante LDA. Este método es idéntico a pLSA (probabilistic latent semantic analysis en inglés), excepto que en LDA la distribución de tópicos se define mediante la

Distribución de Dirichlet como se menciona previamente.
Por ejemplo, un modelo LDA puede tener tópicos clasificados en “relacionados a gatos” y “relacionados a perros”. Un tópico tiene probabilidades de generar varias palabras como leche, miau y gatito, las cuales pueden ser clasificadas e interpretadas como “relacionadas a gatos”. Naturalmente, la palabra gato en sí misma tendrá una alta probabilidad dado este tópico. Por otro lado, los tópicos “relacionados a perros” tienen probabilidades altas para las siguiente palabras: cachorro, ladrar y hueso. Las palabras sin una relevancia especial como “El o La”, es decir, artículos, pronombres, preposiciones no tendrán una probabilidad muy alta (estas palabras son denominadas “stopwords”).

CTM (Correlated Topic Model)

En el trabajo de Blei & Lafferty, 2007, se presenta el Modelo de Tópicos Correlacionados (CTM de las siglas en inglés), el cual explícitamente modela la correlación entre los tópicos latentes en la colección y facilita la construcción de grafos de tópicos y de navegadores de documentos, que permiten a un usuario explorar la colección de documentos a través de los tópicos mencionados.

Un gran ejemplo de uso de este modelo es en el archivo JSTOR del “Journal Science”. Éste es un diario fundado por Thomas Edison en 1880 y que continúa siendo uno de los diarios con mayor influencia en el ámbito de la ciencia hoy en día. La variedad del material que existe en este diario y la cantidad de artículos que contiene, explica la necesidad de métodos automatizados y de modelos estadísticos de tópicos para poder navegar la información contenida en el diario.

CTM, se construye sobre el método ya mencionado LDA. En muchos trabajo recientes se utiliza LDA como base para construir modelos de tópicos más sofisticados tanto para configuraciones que impliquen texto como también para casos “nontext”, es decir, procesamiento de imágenes, sistemas de recomendación, modelado de perfiles de usuarios, entre otros.

La principal característica que utiliza CTM para poder tener en cuenta la correlación de tópicos entre sí, es cambiar la distribución que utiliza, esto es, se reemplaza la distribución de Dirichlet por una distribución normal logística. Si bien el cambio de la distribución otorga esta habilidad para considerar y modelar la correlación, se termina sacrificando alguna de las bondades computacionales que LDA brinda.

BTM (Biterm Topic Model)

Este modelo surge del trabajo de Yan, Guo, Lan, & Cheng, 2013, para afrontar el problema que tienen los textos cortos respecto de la baja co-ocurrencia de los términos de un documento.
Los modelos de tópicos convencionales aprenden en base a patrones de co-ocurrencia de las palabras dentro de un documento o texto. Estos patrones no son muy efectivos en los casos donde los textos son cortos, en donde, la co-ocurrencia se hace más esporádica en cada documento. Para afrontar este problema este modelo propone este novedoso modelo de

bi-términos (biterm), el cual aprende los tópicos en textos cortos modelando la generación de todos los bi-términos que contenga el “corpus” (las palabras del documento).

La idea principal proviene de pensar que los tópicos son grupos de palabras correlacionadas y que la correlación es revelada por patrones de co-ocurrencia de las palabras en los documentos. Además estos patrones de co-ocurrencia permitirán mejorar el problema que sufren los textos cortos respecto de la escasez / dispersión de los términos.

Para explicar mejor el modelo, un bitérmino es un par de palabras desordenadas que co-ocurren en un contexto corto. Como mencionamos antes los tópicos son representados como grupos de palabras correlacionadas, mientras que la correlación es revelada por los patrones de co-ocurrencia existentes en los documentos.
Por ejemplo, si las palabras “apple”, “iphone”, “ipad” y “app” co-ocurren frecuentemente entre sí, en el mismo contexto, podemos identificar que corresponden a un mismo tópico (ej: La empresa Apple y sus productos). Los modelos de tópicos convencionales capturan implícitamente esos patrones de co-ocurrencia a partir de modelar la generación de palabras a nivel documento. En cambio, BTM modela dicha generación a partir de los bitérminos. El bitérmino es una instancia del patrón de co-ocurrencia de palabras. En textos reducidos, debido a que los documentos son usualmente cortos y específicos, se toma a cada documento como una unidad individual de contexto. Luego se extraen como bitérminos todas las palabras de a pares que sean distintas. Por ejemplo en un documento de texto corto “Yo visito apple store”, ignorando la palabra “Yo” (ya que es una stopword y no aporta información), existen tres bitérminos, “visito apple”, “visito store”, “apple store”. Los bitérminos extraídos de todos los documentos componen los datos de entrenamiento de BTM.

Planteo del Problema

Debido a la aparición y al exponencial crecimiento del uso de las redes sociales, las grandes compañías se encuentran muy interesadas en utilizarlas para publicitarse y por sobre todo para conocer acerca de las opiniones que los usuarios de las mismas tienen sobre sus marcas.

La opinión de los consumidores de una marca de una compañía es muy importante para las decisiones de compra de los consumidores. El análisis de los comentarios en las redes es útil para determinar qué se dice acerca de la marca en cuestión y en función de esos tópicos, orientar los equipos internos en la toma de decisiones.

Claramente el gran volumen existente de comentarios en las redes sociales, impide que se puedan procesar por un humano y es ahí donde aparecen diferentes métodos computacionales que pueden resolver este problema con mayor dinamismo y eficiencia.

La idea principal en este trabajo es utilizar la red social Twitter y procesar una gran cantidad de tweets de usuarios de la red que mencionan a una marca o compañía de telecomunicaciones y aplicando Machine Learning poder conocer sobre los tópicos principales que se hablan acerca de dichas compañías.

Hoy en día existen empresas que dan soluciones de Big Data a medianas y grandes compañías.
En particular la empresa ABC S.A., dedicada a dar soluciones de este tipo, carece de una herramienta de detección o modelado de tópicos.
Por lo tanto la idea es generar una herramienta utilizando el modelo LDA que provea de una solución a dicha empresa.

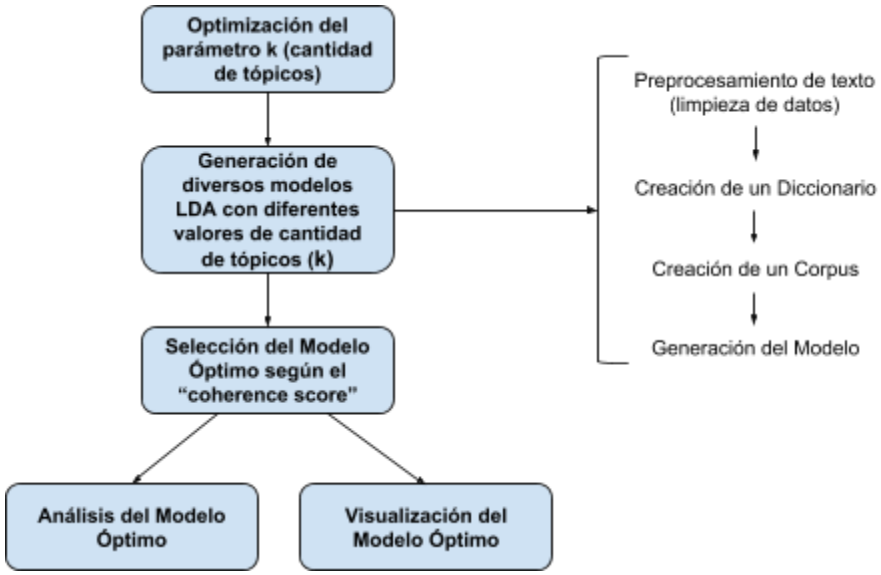
Solución

La solución consiste en una herramienta hecha en lenguaje Python que permita la lectura de archivos en formato json con tweets relacionados a diversas empresas de telecomunicaciones de Argentina para finalmente obtener los tópicos principales de dichos tweets.

El modelo que se utiliza para desarrollar la herramienta es el ya mencionado LDA (Latent Dirichlet Allocation). La elección de LDA se debe a que es el algoritmo más utilizado y probado para este tipo de problemas. Como primera aproximación a la creación de una herramienta de detección de tópicos, me resulta el más indicado para resolver el problema.
En el paquete gensim de Python existe una implementación de este modelo y es el que se utiliza.
La herramienta se desarrollará en el entorno JupyterLab que tiene soporte para notebooks y que provee de manera simple la codificación y pruebas en el lenguaje mencionado.

Etapas

A continuación se ilustra la solución mediante un flowchart para ayudar a explicar los pasos que sigue la herramienta para desarrollar la solución:



Datos a procesar y formato

A continuación se muestra en la Fig. 1, un extracto de los primeros registros de un archivo de ejemplo con tweets que son procesados por la herramienta:

```
{
  "url": "http://twitter.com/Telco1Arg/statuses/1013226819183575041",
  "timestamp": "2018-07-01 01:04:33",
  "text": "@patitaraiolo Te escribí por DM, desde ya estamos a tu disposición, seguimos por esa vía."
}

{
  "url": "http://twitter.com/Telco1Arg/statuses/1013230348497539074",
  "timestamp": "2018-07-01 01:18:34",
  "text": "@miggus1981 buenas noches, necesito que me indiques por DM, tu ubicación actual, así verifico el estado de la cobertura en la zona.  https://t.co/iJYic9wvqb"
}

{
  "url":"http://twitter.com/Telco1Arg/statuses/1013231535116505088",
  "timestamp":"2018-07-01 01:23:17",
  "text":"@miggus1981 Además te voy a pedir que me detalles tu número de línea, sin 0, ni 15 y con la característica de la zona. ¡Te espero!"
}

{
  "url":"http://twitter.com/Telco1Arg/statuses/1013242707706351616",
  "timestamp":"2018-07-01 02:07:41",
  "text":"@TucuViejo ¡Buenas noches! Te escribí por DM para realizar las verificaciones correspondientes, seguimos por esa vía."
}

{
  "url":"http://twitter.com/Telco1Arg/statuses/1013257703513296902",
  "timestamp":"2018-07-01 03:07:16",
  "text":"@DeemiCAI ¡Hola! Buenas noches. ☺ Contanos un poco más en detalle tu propuesta por DM, para chequear que podemos hacer al respecto. ¡Te espero!"
}
```


Fig. 1

Como se visualiza en la Fig. 1, los registros tienen formato json con las siguientes claves:

- “url”: la URL del tweet, la misma contiene quien generó el tweet.
- “timestamp”: fecha en formato AAAA-MM-DD HH:MM:SS
- “text”: texto del tweet.

Herramienta y algoritmo

El procesamiento que realizará la herramienta consistirá de un algoritmo que se puede dividir en tres partes:

1. Limpieza del texto (“Text Cleaning”).
2. Aplicación del modelo LDA (uso de la librería “gensim”).
3. Interpretación de Tópicos de manera visual (uso de la librería “pyLDAvis”).

Limpieza del texto

Lo primero que la rutina de limpieza hace es quitar todos los tipos de acentos y diéresis para unificar el texto. Los famosos y conocidos “hashtags” (son tópicos que define la gente mediante el símbolo # y a continuación una palabra que defina el tópico de lo que se habla) obviamente no se limpian ya que proporcionan información relevante para detectar tópicos.
Luego se transforman las palabras en “tokens” (con la librería Spacy), filtrando aquellos que sean URLs, espacios, menciones a usuarios (expresiones que empiezan con “@”). Si el token no es ninguno de estos caso, se convierte a minúscula.
Luego se verifica que el token tenga una longitud mayor a 4 para limpiar palabras muy cortas.
Posteriormente se limpian las llamadas “stop words”, que principalmente son artículos y preposiciones que no aportan al descubrimiento de los tópicos. También se quitan aquellos términos que no están relacionados con el negocio en cuestión y que pueden generar ruido en los resultados.
Por último los tokens que sirven son transformados a su raíz de origen como palabra mediante la librería “wordnet” (esa transformación se la conoce como lemmatizer).

Modelo LDA de gensim

Para aplicar el método LDA de la librería gensim, se crea un diccionario y un corpus o “bolsa de palabras”[3].
El método que implementa el modelo LDA, presenta ciertos parámetros para definir cómo entrenar y generar el modelo de tópicos. Los principales parámetros son:

1. Cantidad de tópicos: parámetro principal e importante poder definir de alguna forma.
2. Alfa y eta: son hiperparámetros que afectan la densidad de tópicos. El valor por defecto de ambos es 1/num_topics
3. Chuncksize: es el número de documentos a ser utilizados en cada pasada de entrenamiento.
4. Update_every: cada cuanto se actualizan los parámetros del modelo.
5. Passes: la cantidad de pasadas por el corpus durante el entrenamiento.

Visualización con pyLDAvis

La librería pyLDAvis ayuda a interpretar fácilmente los tópicos obtenidos del modelo aplicado. Algunos parámetros que muestra son:

- Saliency: es una medida de cuánto un término dice acerca de un tópico.
- Relevance: un peso promedio de la probabilidad de una palabra dado un tópico y de esa palabra dado el tópico normalizado por la probabilidad del tópico.

[3] https://en.wikipedia.org/wiki/Bag-of-words_model

El tamaño de las burbujas miden la importancia de los tópicos en relación a los datos.

Optimización de Parámetros

Para finalizar, se proveerá a la herramienta de lo que se conoce como “parameter tuning”. Esto implica buscar los mejores parámetros a utilizar en un modelo, en nuestro caso, el LDA.

Al analizar grandes volúmenes de datos para poder detectar los tópicos a los que hacen referencia dichos datos, surge una pregunta clave: ¿cuántos tópicos estamos intentado encontrar?

El modelo LDA requiere que se le defina de antemano la cantidad de tópicos a buscar por lo cual es importante hacer un análisis de este parámetro, previo a ejecutar el modelo.
Por lo tanto basaremos la optimización en el parámetro k “cantidad de tópicos” que es un parámetro clave en el modelo en cuestión.

Si bien en este trabajo no se optimizarán los valores de alfa y beta utilizados por el modelo LDA, es importante remarcar que dependen de cómo uno asuma la simetría de la distribución LDA. En nuestro caso se asume simétrica[4].

[4] https://www.thoughtvector.io/blog/lda-alpha-and-beta-parameters-the-intuition/

Validación

A continuación se explicará los pasos realizados para validar la solución planteada.
Se procesaron 31478 tweets del mes de Julio de 2018 vinculados a importantes compañías telefónicas.
Como he mencionado, la solución es una herramienta de software que utiliza el modelo LDA. Éste modelo impone definir de antemano la cantidad de tópicos a encontrar sobre el set de datos a analizar. Es por esto que este parámetro es de suma importancia conocer antes de entrenar el modelo y por lo tanto comenzaremos explicando cómo optimizar dicho parámetro. Posteriormente, utilizando el parámetro óptimo, se genera el modelo en cuestión observando los tópicos obtenidos y finalmente analizaremos dichos tópicos mediante visualizaciones.

Optimización de Parámetros

Número Óptimo de Tópicos

La forma utilizada para obtener el número óptimo de tópicos es construir varios modelos LDA con diferente número de tópicos (k) y tomar aquel que devuelva el valor de coherencia (c_v)[5] más grande considerando el contexto. Dicho valor de coherencia es una forma simple de ver qué tan bueno es el modelo.
Para la elección del valor k, el c_v no tiene que ser simplemente el más grande y es por eso que también es importante la cantidad de datos que uno está procesando.
Si observamos las mismas palabras clave repetidas en muchos de los tópicos, es probablemente una señal de que el parámetro k es muy grande.
El siguiente gráfico representa los diferentes valores de coherencia obtenidos para distintos k con los que se entrenó el modelo LDA:

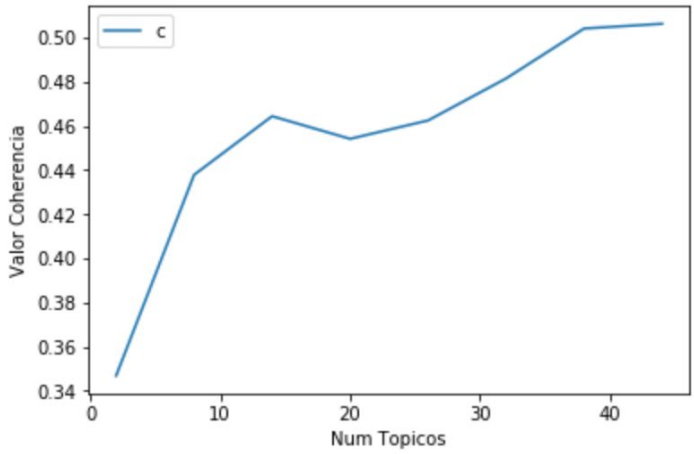


Fig. 3

[5] https://rare-technologies.com/what-is-topic-coherence/

El gráfico de la Fig. 3 se obtuvo a partir de los siguientes valores utilizados:

- Num Topics = 2 has Coherence Value of 0.3468
- Num Topics = 8 has Coherence Value of 0.4378
- Num Topics = 14 has Coherence Value of 0.4644
- Num Topics = 20 has Coherence Value of 0.4541
- Num Topics = 26 has Coherence Value of 0.4625
- Num Topics = 32 has Coherence Value of 0.4815
- Num Topics = 38 has Coherence Value of 0.5041
- Num Topics = 44 has Coherence Value of 0.5063

Como se observa en el gráfico el valor óptimo de k obtenido y utilizado fue 38.
Cuando el valor de coherencia sigue aumentando, lo mejor es elegir el que dé el máximo valor de coherencia antes de que la curva se empiece a hacer más llana.
Para ilustrar a continuación mostramos algunos valores de tiempo que tomó generar cada uno de los modelos y el proceso total de cálculo del valor de coherencia:

--- 160.56327867507935 seconds for model with k = 2 ---
--- 153.87559008598328 seconds for model with k = 8 ---
--- 160.5194797515869 seconds for model with k = 14 ---
--- 171.46977472305298 seconds for model with k = 20 ---
--- 167.3679337501526 seconds for model with k = 26 ---
--- 165.57352209091187 seconds for model with k = 32 ---
--- 182.43980526924133 seconds for model with k = 38 ---
--- 191.59338641166687 seconds for model with k = 44 ---
--- 1553.8482978343964 seconds for compute_coherence_values ---

Observación de los tópicos

Como hemos mencionado previamente, el modelo LDA fue construido con 38 tópicos diferentes, donde cada uno de estos tópicos es una combinación de palabras claves y cada palabra clave contribuye con un cierto peso al tópico.
Podemos ver cada palabra clave de cada tópico y el peso o importancia de cada palabra clave como se muestra a continuación:

	word1	word2	word3	word4	word5
0	0.059*"recibir"	0.036*"participa"	0.033*"disculpas"	0.030*"tenes"	0.022*"canal"
1	0.216*"celular"	0.080*"equipo"	0.061*"combo"	0.030*"telefonia"	0.028*"movil"

2	0.144*"servicio"	0.066*"internet"	0.040*"sumas"	0.038*"problema"	0.038*"quieren"
3	0.133*"empresa"	0.100*"gente"	0.053*"playlist"	0.049*"siguen"	0.043*"cambiar"
4	0.099*"mande"	0.088*"pedido"	0.088*"entradas"	0.060*"veces"	0.043*"puntos"
5	0.212*"hacer"	0.187*"podes"	0.120*"empresas"	0.101*"multadas"	0.059*"cliente"
6	0.087*"puede"	0.076*"chance"	0.046*"beneficios"	0.037*"llamo"	0.028*"disponible"
7	0.154*"quiero"	0.080*"dolares"	0.069*"nuevo"	0.062*"escuchas"	0.037*"linea"
8	0.176*"paginas"	0.092*"consulta"	0.047*"alguna"	0.046*"quedo"	0.032*"disposicion"
9	0.161*"reclamo"	0.088*"esperando"	0.037*"samsung"	0.025*"segun"	0.024*"promocion"
10	0.153*"amigo"	0.104*"mejor"	0.047*"caradeemoji"	0.040*"verdad"	0.030*"emoji"
11	0.105*"junta"	0.037*"chico"	0.030*"quisieron"	0.027*"ayuda"	0.027*"vamos"
12	0.070*"estan"	0.059*"llega"	0.051*"lineas"	0.048*"movistarelegicuidarte"	0.047*"tener"
13	0.076*"temas"	0.071*"claromusica"	0.059*"trastienda"	0.059*"meetandgreet"	0.035*"2.017"
14	0.193*"telecom"	0.059*"cuanto"	0.057*"agosto"	0.041*"pasar"	0.040*"plane"
15	0.085*"puedo"	0.068*"manana"	0.055*"ayudar"	0.047*"pedir"	0.041*"momento"
16	0.111*"nuevos"	0.050*"pedimos"	0.049*"todavia"	0.047*"datos"	0.035*"sigue"
17	0.081*"solucion"	0.059*"informacion"	0.036*"falta"	0.033*"ningun"	0.031*"servicio"
18	0.127*"permiten"	0.048*"saber"	0.042*"entra"	0.039*"clientes"	0.035*"problemas"
19	0.115*"funciona"	0.088*"telefono"	0.069*"lapso"	0.069*"funcione"	0.064*"necesito"
20	0.157*"personal"	0.073*"saludos"	0.073*"tenes"	0.069*"encanta"	0.064*"grosa"
21	0.190*"privado"	0.121*"mensaje"	0.096*"nombre"	0.042*"pas"	0.027*"cuenta"
22	0.287*"pueden"	0.142*"respuesta"	0.036*"saldo"	0.026*"base"	0.020*"espero"
23	0.155*"servicios"	0.110*"queres"	0.074*"simona"	0.069*"pagando"	0.061*"andar"

24	0.063*"tuenti"	0.043*"factura"	0.023*"pagar"	0.022*"cambio"	0.020*"descuento"
25	0.063*"chubut"	0.057*"compania"	0.056*"cuerpo"	0.055*"confirmado"	0.032*"formulario"
26	0.093*"portabilidad"	0.084*"nunca"	0.057*"espera"	0.057*"llego"	0.048*"horas"
27	0.108*"roaming"	0.099*"datos"	0.083*"cangrejo"	0.082*"show"	0.082*"compartan"
28	0.192*"boton"	0.115*"medio"	0.095*"seguimos"	0.061*"recibi"	0.047*"contacto"
29	0.068*"digital"	0.058*"reclama"	0.051*"promo"	0.047*"conmigo"	0.033*"volver"
30	0.138*"linea"	0.115*"numero"	0.089*"espero"	0.057*"ayudarte"	0.040*"poder"
31	0.092*"millones"	0.082*"ustedes"	0.061*"atras"	0.053*"me"	0.038*"siempre"
32	0.195*"datos"	0.065*"ordene"	0.064*"preserve"	0.063*"operativo"	0.044*"exijimos"
33	0.321*"gracias"	0.089*"paraguay"	0.083*"celulares"	0.075*"respondi"	0.039*"muchas"
34	0.068*"claro"	0.052*"senal"	0.050*"atencion"	0.044*"servicio"	0.036*"movistar"
35	0.061*"numeros"	0.048*"llamar"	0.043*"credito"	0.035*"llaman"	0.034*"decir"
36	0.094*"envie"	0.087*"conoce"	0.075*"averigualo"	0.074*"laliga3x3movistar"	0.046*"respondimos"
37	0.124*"mismo"	0.059*"pasame"	0.041*"vemos"	0.040*"contame"	0.034*"acabo"

El resultado anterior se interpreta de la siguiente forma:
El tópico 9 es presentado por:

9	0.161*"reclamo"	0.088*"esperando"	0.037*"samsung"	0.025*"segun"	0.024*"promocion"

Los pesos reflejan qué importancia otorga cada palabra clave a ese tema.
Observando las palabras claves se debe poder resumir cuál es el tópico en cuestión al que pertenecen. En el caso del tópico 9 podríamos decir que está asociado al tópico “Reclamos empresa por premios o promoción”.

De la misma forma, vamos viendo las palabras claves de los otros tópicos para definir a qué hace referencia cada uno.

Visualización de Palabras Clave de los Tópicos

Para tener una visualización interesante de los tópicos y sus palabras claves, se utilizó la librería pyLDAvis, que nos provee de una forma simple e interactiva analizar los resultados obtenidos. Esta visualización permite enriquecer la herramienta a desarrollar.
A continuación mostramos lo obtenido con los datos utilizados:

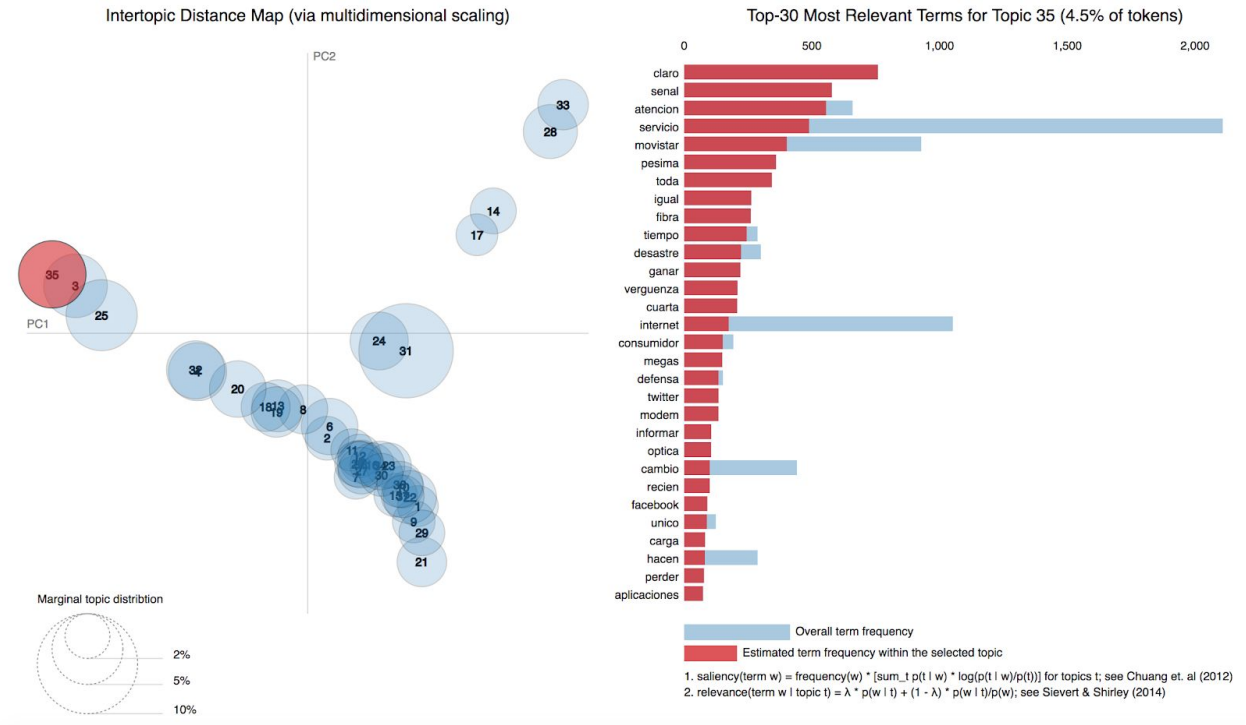


Fig. 2

En la Fig. 2 se muestra una imagen de la visualización obtenida.
Cada burbuja en la parte izquierda de la visualización representa un tópico. Cuanto más grande la burbuja, más predominante es ese tópico.
Un buen modelo de tópicos es aquel que tiene burbujas bastante grandes y que no se solapan, dispersas en todo gráfico en lugar de estar todas juntas y clusterizadas en un único cuadrante.
Un modelo con muchos tópicos seguramente tendrá burbujas pequeñas, ubicadas en una misma región del gráfico y con muchos casos de solapamiento.
Si movemos el cursor sobre cada una de las burbujas, las palabras y las barras del gráfico de la derecha se actualizarán. Estas palabras son las palabras clave más importantes que forman el tópico seleccionado.
Las burbujas que se solapan hacia la abajo del gráfico (para mencionar algunos 4, 32, 20), si observamos las palabras claves, están claramente definidas en un tópico que podríamos llamar “Reclamos y Quejas”, siendo más específico, el 32 está asociado a “Respuesta a un problema

inconclusa” y en el caso del 20 a un “Problemas técnicos”, y el 4 a “Clientes insatisfechos con voluntad de cambiar de empresa”.
Por otro lado, otro caso más disperso a la derecha (por ejemplo la 31) están claramente asociadas a “Atención al cliente”. La burbuja que se ve en la parte superior (33) está asociada a temas no relacionados con las telecomunicaciones con lo cual podemos decir que está relacionada la actividad muy común en las redes conocida como “trolling”[6].
Se adjunta a este trabajo el archivo html interactivo para que se puedan corroborar y apreciar de una mejora forma los resultados explicitados.

[6] https://es.wikipedia.org/wiki/Trol_(Internet)

Conclusiones y Futuros Trabajos

A lo largo de este trabajo he podido mostrar, a través del desarrollo de una herramienta, cómo podemos procesar un gran volumen de tweets y poder detectar tópicos o temas a los que hacen referencia los usuarios de una red social.

En nuestro caso trabajamos con datos de empresas de telecomunicaciones, pero es evidente que es de muchísima importancia para cualquier gran compañía (muy presentes hoy en día en las redes sociales), conocer lo que los usuarios comentan o expresan acerca de sus servicios y/o productos.
Todo el conocimiento adquirido permite a las empresas conocer bondades y problemas de los servicios que brindan, qué nuevos servicios, productos u oportunidades pueden ofrecer a sus cliente actuales y/o potenciales, qué decisiones tomar respecto de diversas cuestiones como la calidad del servicio, la fidelidad de sus clientes y otras tantas cuestiones que preocupan a las grandes compañías.

Es por esto que la herramienta desarrollada, aporta una buena aproximación para lograr automatizar y resolver el problema de detección de tópicos de manera muy rápida y simple.
La herramienta no sólo automatiza el problema de procesar grandes volúmenes de datos, sino que también provee visualizaciones que permiten analizar de manera sencilla los tópicos a descubrir.

Es importante destacar que la herramienta desarrollada puede ser mejorada trabajando en conjunto con un especialista en el negocio en cuestión. Éste puede moldear la herramienta para obtener mejores resultados, como por ejemplo, filtrando ciertas “stopwords” que no aportan valor agregado al modelo.
Además, por la dinámica de las redes sociales, el proceso de entrenamiento del modelo debe ser constante ya que aparecen nuevas palabras, modas y formas nuevas de expresarse que ameritan el reentrenamiento y el trabajo en conjunto con el especialista.

Una característica importante que se incluyó en la herramienta es la posibilidad de optimizar el parámetro k de tópicos a obtener. Esto es clave en el modelo utilizado ya que dicho parámetro es necesario para el entrenamiento del mismo.

Si bien la herramienta presenta muchísimas bondades, está claro que presenta ciertas limitaciones que podrían mejorar aún más el proceso de detección de tópicos.
El modelo LDA utilizado en esta herramienta fue optimizado para los datos de prueba procesados, cuando el dataset cambie será importante optimizar los parámetros nuevamente para mantener la calidad de los resultados.

Como primera mejora, sería de gran utilidad implementar la obtención de los datos utilizando streaming con alguna política de cortes (por día por ejemplo) ya que que la solución planteada en la herramienta se basa en la lectura de un archivo con los datos a procesar.

Por otro lado, la solución planteada utiliza un diccionario de “stopwords de negocio” para evitar considerar palabras no relevantes al negocio. Dicho diccionario puede mejorarse aún más trabajando en junto con un especialista en la cuestión.

También sería interesante explotar la herramienta con datos en otros idiomas, con lo cual debería soportar el preprocesamiento del texto en diferentes lenguajes, incorporando la posibilidad de configurarlo.

Por otro lado el modelo LDA, que si bien es el más utilizado para la detección de tópicos, no es el único existente para resolver el problema. Sería interesante, agregar a la herramienta la posibilidad de entrenar un mismo set de datos con diversos modelos (como “correlated topic model” o “biterm topic model”) y poder comparar los resultados obtenidos.

Además sería de sumo interés mejorar la herramienta para que pueda procesar datos de otras redes sociales como Facebook e Instagram, ya que es sabido que las grandes compañías tienen presencia en las redes mencionadas.

Bibliografía y Referencias

When it comes to social media, consumers trust each other, not big brands. (2017, September 19). Retrieved August 5, 2018, from https://phys.org/news/2017-09-social-media-consumers-big-brands.html

Risch, J. 2016. Detecting Twitter topics using Latent Dirichlet Allocation. Uppsala Universitet

Hoffman, M., Bach F., & Blei, D. (2010). Online Learning for Latent Dirichlet Allocation. Advances in Neural Information Processing Systems 23. pages 856–864. Curran Associates Inc.

Blei, D., Ng A. & Jordan, M (2003). Latent Dirichlet Allocation. Journal of Machine Learning Research, 994–1022, January 2003.

Blei, D. & Lafferty J (2007). A CORRELATED TOPIC MODEL OF SCIENCE. Princeton University and Carnegie Mellon University. 17–20.

Yan X., Guo J., Lan Y. & Cheng X (2013). A Biterm Topic Model for Short Texts. Institute of Computing Technology, CAS Beijing, China. 1444-1447. From http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.402.4032&rep=rep1&type=pdf

LDA Alpha and Beta Parameters - The Intuition. (2018, October), Retrieved October 10, 2018 from https://www.thoughtvector.io/blog/lda-alpha-and-beta-parameters-the-intuition/

Anexo I: Código de la herramienta

```
# Preprocessing functions before applying LDA model.

import unicodedata
def accent_fold(s):
    return ''.join(c for c in unicodedata.normalize('NFD', s)
                   if unicodedata.category(c) != 'Mn')

# Terms that dont belong to the business case
terms_dictionary = ["buenas noches", "buenas tardes", "buen dia", "buenas", "hola", "que tal",
"correspondiente", "respecto",
               "algun", "alguno", "tambien", "tampoco", "tarde", "noche", "dia", "dias", "tardes",
"noches",
                "ahora", "despues", "santiago", "maldonado", "santiagomaldonado", "lleral",
"gendarme", "gendarmes", "violencia"]

import gensim
import pickle
import spacy
spacy.load('es')
from spacy.lang.es import Spanish
parser = Spanish()
def tokenize(text):
    lda_tokens = []
    tokens = parser(text)
    for token in tokens:
        if token.orth_.isspace():
            continue
        elif token.like_url:
            lda_tokens.append('URL')
        elif token.orth_.startswith('@'):
            #lda_tokens.append('SCREEN_NAME')
            continue
        else:
            lda_tokens.append(token.lower_)
    return lda_tokens

import nltk
nltk.download('wordnet')
from nltk.corpus import wordnet as wn
```

```
def get_lemma(word):
    lemma = wn.morphy(word)
    if lemma is None:
        return word
    else:
        return lemma

from nltk.stem.wordnet import WordNetLemmatizer
def get_lemma2(word):
    return WordNetLemmatizer().lemmatize(word)

nltk.download('stopwords')
es_stop = set(nltk.corpus.stopwords.words('spanish'))

def prepare_text_for_lda(text):
    text = accent_fold(text)
    tokens = tokenize(text)
    tokens = [token for token in tokens if len(token) > 4]
    tokens = [token for token in tokens if token not in es_stop]
    tokens = [token for token in tokens if token not in terms_dictionary]
    tokens = [get_lemma2(token) for token in tokens]
    return tokens

# Reading JSON file and executing preprocessing of every tweet
import random
import json
from pprint import pprint
text_data = []
with open('telecom-ar.json') as f:
    for line in f:
        data = json.loads(line)
        tokens = prepare_text_for_lda(data["text"])
        #print(tokens) #uncomment this line to show the generated tokens
        text_data.append(tokens)

#pprint(data["text"])

from gensim import corpora
dictionary = corpora.Dictionary(text_data)
corpus = [dictionary.doc2bow(text) for text in text_data]

pickle.dump(corpus, open('corpus.pkl', 'wb'))
dictionary.save('dictionary.gensim')
```

```
import gensim
import time
from gensim.models.coherencemodel import CoherenceModel

def compute_coherence_values(dictionary, corpus, texts, limit, start=2, step=3):
    """
    Compute c_v coherence for various number of topics

    Parameters:
    ----------
    dictionary : Gensim dictionary
    corpus : Gensim corpus
    texts : List of input texts
    limit : Max num of topics

    Returns:
    -------
    model_list : List of LDA topic models
    coherence_values : Coherence values corresponding to the LDA model with respective
number of topics
    """
    coherence_values = []
    model_list = []
    for num_topics in range(start, limit, step):
        #the random_state parameter is fundamental if you want to reproduce the training run, its
like a random seed
        s_time = time.time()
        model = gensim.models.ldamodel.LdaModel(corpus, num_topics = num_topics,
id2word=dictionary, passes=15, random_state=1)
        print("--- "+str(time.time() - s_time)+" seconds for model with k = "+str(num_topics)+" ---")
        model_list.append(model)
        coherencemodel = CoherenceModel(model=model, texts=texts, dictionary=dictionary,
coherence='c_v')
        coherence_values.append(coherencemodel.get_coherence())

    return model_list, coherence_values



# Execution of several models to compare the coherence value
import time
start_time = time.time()
```

```
model_list, coherence_values = compute_coherence_values(dictionary=dictionary,
corpus=corpus, texts=text_data, start=2, limit=50, step=6)
print("--- %s seconds for compute_coherence_values ---" % (time.time() - start_time))

import matplotlib.pyplot as plt
get_ipython().run_line_magic('matplotlib', 'inline')

# Show graph
limit=50; start=2; step=6;
x = range(start, limit, step)
plt.plot(x, coherence_values)
plt.xlabel("Num Topicos")
plt.ylabel("Valor Coherencia")
plt.legend(("coherence_values"), loc='best')
plt.show()

# Print the coherence scores
for m, cv in zip(x, coherence_values):
    print("Num Topics =", m, " has Coherence Value of", round(cv, 4))


# Visualization
import pandas as pd
optimal_model = model_list[6]
topics = optimal_model.print_topics(num_topics=-1, num_words=5)
t = []
for topic in topics:
    t.append(topic[1].split("+"))

#pprint(t)
sent_topics_df = pd.DataFrame(data=t,columns=["word1","word2","word3","word4","word5"])

sent_topics_df

import pyLDAvis.gensim

lda_display = pyLDAvis.gensim.prepare(optimal_model, corpus, dictionary, sort_topics=False)
#pyLDAvis.display(lda_display)
#uncomment next line if you want to make an html file with the visualization
pyLDAvis.save_html(lda_display, 'lda.html')
```