Predicción de Tendencias en Redes Sociales basada en características sociales y contenido [1]

Autor: Matías Gastón Silva.
Director: Dr. Martín A. Domínguez.
Co-director: Lic. Pablo Gabriel Celayes.
Facultad de Matemática, Astronomía, Física y Computación
Universidad Nacional de Córdoba

24 de abril de 2018

Resumen

En el marco del análisis de redes sociales éste trabajo busca capturar el comportamiento de los usuarios influyentes sobre una publicación determinada. Con esta información, la intención es generar un modelo de aprendizaje automático capaz de predecir si un determinado tweet será "popular" o no.

La construcción del conjunto de datos (*dataset*) fue realizada a través de la API pública de Twitter obteniendo un volumen final de más de 5,000 usuarios y 5,000,000 de publicaciones.

Con esta información se entrenaron y evaluaron diversos modelos de aprendizaje automático con múltiples configuraciones, con el objetivo encontrar así el mejor rendimiento.

En este sentido, en un primer experimento, se logró inferir un modelo de clasificación binaria basado en SVM (*Support Vector Machines*) sólo utilizando información social, qué obtuvo un 77 % de certeza, basado en la métrica F1, para predecir si una publicación es considerada "popular".

En una segunda etapa, se decidió agregar técnicas de Procesamiento de Lenguaje Natural aplicadas sobre el contenido de las publicaciones, logrando algunas mejoras significativas en los casos donde el modelo anterior se veía disminuido. Dicho análisis de los tweets fue realizado utilizando detección de tópicos, mediante algoritmos tipo LDA (Latent Dirichlet Allocation).

Palabras Clave: Análisis de Redes Sociales, Aprendizaje Automático, Detección de influenciadores, Detección de comunidades, Modelos de predicción, Twitter, LDA.

Clasificación (ACM CCS 2012):

- **Applied computing∼Sociology**
- **Computing methodologies∼Natural language processing**
- Computing methodologies∼Support vector machines
- Computing methodologies∼Latent Dirichlet allocation

Abstract

In the framework of social network analysis, this work seeks to capture the behavior of influential users about a specific publication. With this information, the intention is to generate an automatic learning model capable of predicting if a certain tweet is popular or not.

The construction of the dataset was made through the public Twitter API obtaining a final volume of more than 5,000 users and 5,000,000 publications.

With this information, different models of machine learning with multiple configurations were trained and evaluated, in order to obtain the best performance.

In this sense, in a database we can infer a classification model based on SVM (*Support Vector Machines*) only using social information, which obtained a 77 % certainty, based on the F1 metric, for predict whether a publication is considered "popular".

In a second stage, it was decided to add Natural Language Processing techniques, earning significant improvements in the cases where the previous model was reduced. This analysis of the tweets was done by detection of topics, through LDA(Latent Dirichlet Allocation) algorithms.

Agradecimientos

A mis directores Martin Dominguez y Pablo Celayes, por su paciencia y predisposición para realizar este proyecto.

A mis viejos, quienes con mucho esfuerzo siempre me apoyaron e incentivaron en el estudio dándome la oportunidad de hacerlo de manera exclusiva y dedicada haciendo qué nunca me falte nada.

A mis hermanos Gonzalo y Cristian, por los momentos compartidos y por ayudar a mis viejos a educarme desde los primeros pasos hasta hoy siempre remarcando la humildad y respeto.

A Laura Alonso i Alemany y Franco Luque, por su colaboración en la presentación de este trabajo cómo tribunal de evaluación.

A mis amigos de siempre, en especial los qué me acompañaron desde Neuquén en esta apuesta de estudiar en Córdoba, por los asados, mates, salidas y demás cosas qué nunca faltaron en los momentos donde hacía falta despejar la mente.. la segunda familia.

A mis amigos de la facultad que hicieron que cada una de mis actividades sea mas sencilla con un trabajo colaborativo destacable incluso en lo recreativo.

A mi colegio secundario, el *Instituto Técnologico del Comahue*, por la formación básica en muchas de las materias de la carrera, por mostrarme mi vocación y permitirme conocer el lugar donde hoy resido.

A la facultad y sus docentes, por la formación inigualable gracias a su dedicación y entrega hasta fuera del horario.

A mis compañeros de *Loghinet* e *Infoxel*, por compartir su conocimiento y ayudar en el crecimiento ajeno siempre con un nivel humano sobresaliente.

A las empresas anteriores por la oportunidad de desarrollarme profesionalmente aunque siempre priorizando mi carrera académica.

A la comunidad de software y los múltiples eventos relacionados, por permitirme asistir, pudiendo aplicar, afianzar y ampliar los conocimientos y conocer gente con las mismas pasiones.

A la educación pública, qué me dió la oportunidad de ser hoy un profesional y participar competentemente en el mercado laboral.

Índice general

Capítulo 1

Introducción

1.1. Idea central

No es novedad el impacto qué causó y causa constantemente Internet en la vida cotidiana. Junto con la evolución de la tecnología, el constante crecimiento de la infraestructura permite hoy estar conectados a nuestras redes sociales las 24 hs del día, estemos donde estemos. Debido a esta permanente comunicación, las redes sociales hoy son un gran reservorio de información valiosa. Por esta razón el análisis de estos datos puede ser importante para entender fenómenos sociales y las características de la gente qué las utiliza.

Un gran ejemplo, es el terreno del *marketing*, donde en la promoción de una marca, empresa o perfil personal, es fundamental entender los gustos y necesidades de la población, cómo así también contar con una buena imagen y aceptación. Es por esto qué anticipar y analizar el impacto qué pueda tener el contenido qué se publica puede ser de gran ayuda.

Siguiendo en esta vía, la idea principal es desarrollar modelos entrenados qué basándose en la aceptación de los *influenciadores* de la red, puedan anticipar de manera confiable el impacto de una publicación sobre el resto de la población, determinando si será de relevancia o no.

Como punto de partida para iniciar esta investigación se tomó el trabajo realizado por Pablo Celayes "*Recomendación de Información basada en Análisis de Redes Sociales y Procesamiento de Lenguaje Natural*"[1] el cuál propone un modelo de predicción para el comportamiento de un usuario frente a las publicaciones usando la información de su circulo de amistades.

Comenzando con la implementación de esta idea se partió de un dataset de tweets cómo base, el cuál fue complementado con información actualizada obteniendo así, prácticamente el doble de información de partida.

Dada la demora qué esta primer actividad demandó, simultáneamente fueron realiza-

das muchas de las demás tareas.

Evaluar las diversas alternativas para detectar aquellos usuarios más determinantes de la red denominados influenciadores fué una de ellas. Aquí se pudo ver qué para ser denominado de esta manera no sólo bastará con tener abundancia de contactos sino también ser un usuario activo en la red.

El análisis propio de los datos de nuestro dataset por su parte permitió determinar la heurística qué dió lugar a clasificación de un tweet cómo relevante siempre y cuando supere una cota de retweets conseguidos.

Finalizadas las tareas de análisis se implementaron alternativas de clasificadores binarios y su parametrización, para así determinar aquellos con mejor rendimiento bajo los distintos escenarios. No fue sorpresa aquí encontrar los mejores resultados bajo los clasificadores tipo *SVM*.

Como primer modelo se evaluó la performance de una alternativa puramente social donde no intervenga información del contenido sino sólo la referente a los participantes de la comunicación. Con resultados muy satisfactorios la predicción sobre aquellos ejemplos no vistos en el entrenamiento resultó ser muy acertada.

Se realizó también una incorporación de técnicas de Procesamiento de Lenguaje Natural y detección de tópicos mediante *LDA* sobre un modelo evolucionado qué reflejó mejoras significativas sólo en los casos con escasos *influenciadores* seleccionados cómo objetivo.

Como último experimento y puntapié para una futura entrega se planteó una solución qué incorpora la detección de comunidades para un análisis más clasista de los usuarios, aportando así más información al modelo en su entrenamiento. Los resultados poco reveladores de estos experimentos dejan abierto el análisis y evaluación sobre datasets de mayor tamaño para diagnosticar su rendimiento. Como resultado final vimos cómo información socialmente relevante, qué el comportamiento humano al menos en un entorno social virtual puede llegar a ser muy predecible.

1.2. Estructura

Concluida la sección introductoria del *capítulo 1*, a continuación se detalla la estructura del contenido siguiente.

En el *capítulo 2* se desarrollarán los conceptos teóricos aplicados. Las técnicas de análisis de grafos y análisis de lenguaje natural serán explicadas en ésta sección cómo así también los diversos métodos de clasificación utilizados en los modelos predictivos y algunos trabajos relacionados consultados.

Las herramientas y frameworks utilizados serán mencionados, junto a una breve descripción de su utilización dentro de la aplicación, en el *capítulo 3*.

Por su parte el *capítulo 4* describe el conjunto de datos y su procedencia, los usuarios objetivo y sus conexiones para obtener finalmente nuestros datos de muestra.

Durante el *capítulo 5* se abordará plenamente el proceso de desarrollo del modelo predictivo. No sólo se describirá el modelo final, sino también su evolución y alternativas. Se detallarán también los experimentos realizados y sus resultados pertinentes.

Por ultimo en el *capítulo 7* se concluirá con observaciones y detalles de trabajo futuro.

Capítulo 2

Fundamentos Teóricos

Para el desarrollo de una idea es imprescindible contar con una base teórica de conceptos y términos qué nos ayuden con la implementación y comunicación de la misma. Es por ello qué a continuación se introduce el vocabulario y las ideas conceptuales de cada sección útil en nuestra tarea.

2.1. Grafos

Los grafos son una estructura de datos muy popular en el ámbito matemático e informático. Compuestos por puntos llamados vértices o nodos y arcos o aristas qué los conectan buscan representar información relacional. Las características de sus aristas permiten clasificar a nuestros grafos en diferentes categorías, definiendo por ejemplo a un grafo direccional si sus aristas no sólo unen 2 vertices sino qué también explicitan el sentido de esta relación, en el caso contrario el grafo se definirá cómo no direccional. Por otro lado las posibles múltiples relaciones entre dos mismos vértices o incluso una relación de un vértice con el mismo (bucle) define a los grafos cómo multigrafos o pseudografos, en cuyo caso contrario se denomina al grafo cómo un grafo simple. Esta estructura de datos es ideal para el diagrama de una red de usuarios y sus conexiones "sociales". En nuestro caso se utilizará de esta forma en la cuál los vértices serán nuestros usuarios objetivo y las aristas indicarán la acción de seguir a otro usuario. Cómo la red social elegida para el trabajo considera qué la relación de seguimiento es unilateral, nuestro grafo será un grafo dirigido. El universo de los grafos permite seguir clasificando más allá de las características de sus aristas, es por eso qué surgen términos cómo centralidad, clustering o comunidades qué nos aportarán más información de los vértices y sus conexiones. Veamos entonces qué significan estos términos.

2.1.1. Centralidad e Influencia

La centralidad en un grafo, es una característica propia de los nodos y define la importancia relativa qué poseen dentro del mismo. Analizar esta característica puede ayudarnos a encontrar aquellos nodos referentes ya sea por su gran cantidad de nodos adyacentes, cómo así también por su ubicación estratégica qué permita la interconexión de grandes componentes conexas dentro del mismo. Dado qué cualquier acción qué realicen causará un impacto qué se reflejará en una gran porción de la red, cada uno de estos nodos significativos será denominado nodo influenciador y será fundamental su participación en el caso de querer causar un mensaje epidémico logrando cobertura de la forma más rápida. A continuación se detallan algunas de las medidas de centralidad más populares.

Valencia

Existen medidas qué nos pueden aportar información importante en la búsqueda de la centralidad. Una de ellas es la valencia ó grado de un vértice, definido cómo la cantidad de nodos qué tiene a su alcance en un sólo paso de distancia, es decir la cantidad de nodos adyacentes. Esta medición puede ser discriminada en grado ingresante, para la valencia teniendo en cuenta sólo los accesos al nodo o grado egresante contabilizando las relaciones salientes del mismo en un grafo dirigido.

Interconectividad (*betweenness*)

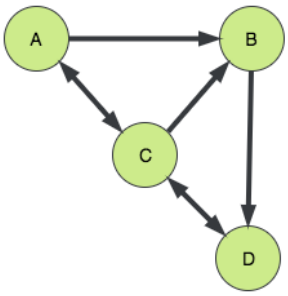
Definida formalmente por Linton Freeman[2], en teoría de grafos la interconectividad es una medida de centralidad basada en las trayectorias más cortas. Para cada par de vértices en un grafo conectado, existe al menos una ruta más corta entre los vértices, de modo qué el número de aristas por los qué pasa la ruta está minimizado. La interconectividad entonces para un vertice i será la proporción de trayectorias más cortas desde dos vértices distintos a i en las cuales participa este vértice sobre el total de trayectorias más cortas.

$$BET(i) = \sum_{s \neq i \neq t} = \frac{\sigma_{s,t}(i)}{\sigma_{s,t}}$$

donde $s, i, t \in vertices(G)$, $\sigma_{s,t}$ será el total de caminos más cortos desde s hacia t y $\sigma_{s,t}(i)$ será la proporción de los mismos donde encontraremos al vértice i en la ruta.

Pagerank

Otro algoritmo muy popular en la medición de centralidad es PageRank. Introducido por Google en 1999[3] causó revolución en su popular buscador web. No sólo involucra la información de adyacencia del propio nodo, sino también de sus nodos adyacentes. Es



	Iteración 1	Iteración 2	Iteración 3
A	1/4	1/12	1.5 / 12
B	1/4	1/8 + 1/12 = 2.5/12	2 / 12
C	1/4	1/8 + 1/4 = 4.5/12	4.5 / 12
D	1/4	1/4 + 1/12 = 4/12	4 / 12

Figura 2.1: Ejemplo de cálculo de pagerank mediante 3 iteraciones con $d = 1$

decir, tendremos información más allá de una arista de distancia para el nodo observado. PageRank es un algoritmo iterativo qué inicia otorgándole un puntaje de 1/N a todos los nodos donde N será la cantidad de vértices del grafo. Luego comenzando por algún nodo seleccionado al azar inicia su recorrido otorgándole su puntaje dividido la cantidad de sus aristas salientes a todos sus vértices adyacentes. El vértice receptor suma lo enviado por todos sus vecinos a su puntaje asignado. Una vez qué todos los vértices han cumplido con la tarea de enviar el puntaje adicional a sus nodos adyacentes, la primer iteración habrá concluido. Ordenando los nodos de forma descendente bajo este puntaje resultante luego de varias iteraciones nos otorgará el ranking de relevancia dentro de nuestro grafo. El algoritmo cuenta con un factor de amortiguación qué modifica ligeramente el resultado anteriormente mencionado quedando cómo resultante la siguiente fórmula final para el cálculo del PageRank de un nodo j donde $in(j)$ será el conjunto de conexiones entrantes hacia el nodo y $C(j)$ su valor de PR.

$$PR(j) = (1 - d) + d * \sum_{i \in in(j)} \frac{PR(i)}{C(i)}$$

En la Figura 2.1 podemos ver el calculo resultante de PageRank durante 3 iteraciones y un d fijado en 1, sobre un grafo de 4 nodos.

Centralidad de Katz

Introducida por Leo Katz en 1953[4] con similitud a Pagerank, la centralidad de Katz es una medida qué aporta información de los nodos adyacentes inmediatos cómo así también de otros nodos qué se conecten a través de ellos. Las conexiones hechas con vecinos intermediarios, sin embargo, están penalizadas por un factor de atenuación α el cuál es degradado exponencialmente mediante la cantidad de intermediarios qué participen.

Sea A la matriz de adyacencia y A^k la matriz de conexión entre vértices mediante k intermediarios, n la cantidad de nodos y α el factor de atenuación tenemos que:

$$KATZ(i) = \sum_{k=1}^{\infty} \sum_{j=1}^{n} \alpha^k (A^k)_{ij}$$

2.1.2. Comunidades

Un grafo es denominado **conexo** si todos sus nodos están conectados mediante un camino posible dentro del grafo. Pero pueden existir grupos de vértices qué compartan características muy similares o se agrupen en conglomeraciones mucho más interconectadas qué el resto. Es por esto qué la búsqueda de grupos o comunidades dentro de un mismo grafo es vital para reducir un problema, simplificándolo a aquellos nodos qué se comporten igual. Esta tarea va en contra de la generalización de la solución pero puede mejorar mucho en la precisión del resultado. Sectorizando, con una técnica de *divide y vencerás*, se puede evaluar una solución para cada comunidad y en el caso de poder analizar todos los problemas particulares de cada sector se podrá concluir en una solución general.

La tarea de detección de comunidades es un área de innovación y continuo desarrollo. Algunos de los algoritmos utilizados en la detección son los siguientes.

- **Infomap**: Este algoritmo fue propuesto por Rosvall et al[5]. Reconoce las comunidades empleando caminantes al azar qué analizan el flujo de información a través de una red[6]. En primer instancia los nodos y aristas son etiquetados en el grafo, luego se lanzan aleatoriamente los caminantes sobre el mismo, cuyos caminos distinguirán los nodos agrupándolos en clusters. Mientras menos caminantes persistan en el procedimiento, más información de la red habremos conseguido. Este algoritmo se ejecuta en el orden $\mathcal{O}(E)$ y es uno de los más rápidos en la tarea de detección de comunidades.[7].

- **Multilevel**: Algoritmo introducido por Blondel et al[8], evolución del método Fastgreedy. Este método primero asigna una comunidad diferente a cada nodo de la red, luego un nodo se mueve a la comunidad de uno de sus vecinos con el qué logra la mayor contribución positiva a la modularidad. El paso anterior se repite hasta qué sólo quede un nodo o la modularidad no pueda aumentarse en un sólo paso. Al basarse en la métrica de modularidad, las comunidades pequeñas pueden ser dificiles de detectar por este algoritmo. Su complejidad es $\mathcal{O}(N \log N)$[9].

- **Walktrap**: Este algoritmo propuesto por Pon y Latapy[10] es un algoritmo de agrupamiento jerárquico. La idea es qué las comunidades estén formadas por aquellos vértices qué estén comunicados por una corta distancia. A partir de un partición totalmente no agrupada, se calculan las distancias entre todos los nodos adyacentes.

En el paso siguiente, dos comunidades adyacentes son elegidas, se fusionan en una nuevo y las distancias entre las comunidades se actualizan. Dado qué se trata de un algoritmo iterativo, este paso se repite (N-1) veces, por lo qué la complejidad computacional de este algoritmo es $\mathcal{O}(EN^2)$. Para redes dispersas la complejidad computacional es $\mathcal{O}(N^2 \log N)$[11].

2.2. Procesamiento de Lenguaje Natural

El PLN es el campo de las Ciencias de la Computación qué comparte con otras áreas relacionadas cómo inteligencia artificial y lingüística. Juntas buscan resolver las problemáticas del lenguaje humano. Los modelos aplicados no sólo se enfocan en la comprensión del lenguaje, sino también en representar aspectos cognitivos humanos, de manera eficaz y confiable. Este tipo de algoritmos se pueden encontrar con inconvenientes qué el humano resuelve muchas veces de manera intuitiva cómo es el caso de la ambigüedad, la comprensión de información difusa o culturalmente adaptada, la detección de sentimientos cómo la ironía o burla entre otras problemáticas comunes qué cambian rotundamente el sentido del mensaje. Un modelo de PLN siempre parte de un conjunto de datos denominado corpus y cada palabra del texto se la menciona cómo token cuya detección en el corpus es denominada tokenización. El corpus de lenguaje natural puede contener demasiada información para un modelo computacional. La abundancia de conectores ó los múltiples signos de puntuación entre otros pueden entorpecer la búsqueda de patrones o reglas qué ayuden a entender las palabras y su contexto. A continuación veamos cómo hacer para evitar este inconveniente procesando previamente los datos.

2.2.1. Preproceso

Dentro de toda tarea de PLN existe un preproceso qué desarrolla una tarea de simplificación de los datos de entrada provenientes del corpus, lo cuál permite enfocar la tarea de un modo más analítico descartando aquella información qué no sea de interés. Algunas de las tareas desarrolladas en el preproceso son:

- **Remoción de palabras auxiliares**: Es muy común en el vocabulario humano el uso de conectores u otras palabras qué sólo tienen el fin de interconectar de manera ordenada aquellas palabras de la oración qué le aportan un sentido real a la misma. En la mayoría de los procesos de análisis computacional estas palabras, denominadas stopwords influyen negativamente entorpeciendo el algoritmo, es por ello qué se deciden retirar.

- **Remoción de símbolos**: Si bien para tareas cómo análisis de sentimientos los símbolos pueden aportar información, para el general suelen ser de poca utilidad. Es por

ello qué pueden ser removidos de las oraciones u adaptados para una simplificación donde los modelos sólo tengan en cuenta palabras.

- **Reemplazo o remoción de números**: Los números suelen ser tratados todos por igual. Debido a su gran cantidad y variedad a lo largo de los textos resulta dificultoso encontrar patrones similares. Es por ello qué en muchos casos todos los números son reemplazados por una palabra clave sin importar cuántos dígitos o cuál sea su cifra, con el objetivo de poder encontrar similitudes de contexto, cómo podría ser, si una palabra es acompañada por un número cómo antecedente o consecuente sin importar su valor.

- **Stemming**: Se trata de un método cuyo objetivo es agrupar aquellas palabras qué deriven de una misma raíz. Se dice qué una palabra tiene una raíz X si la misma comienza con X y continúa o no con demás letras. Preprocesar un corpus aplicando stemming, involucra el reemplazo de cada palabra por su raíz, de forma tal qué si dos palabras son distintas pero su raíz es la misma, serán tratadas de igual manera en el futuro. Cabe aclarar qué por características del lenguaje, no en todos los casos es una técnica saludable y dependerá mucho de la formación de las palabras y sus tiempos verbales. Con buenos resultados verificados sobre el inglés, se decidió experimentar esta técnica en el trabajo.

- **Lematización**: Tarea similar a la de stemming. La lematización es el reemplazo de una palabra por su forma normal. Es decir sin tener en cuenta genero, tiempo verbal, o número. De esta manera trataremos igual aquellas palabras qué signifiquen lo mismo pero estén afectadas por alguno de estos modificadores.

- **Minúsculas**: Pasar las palabras a minúsculas permite tomar por igual si una palabra esta al inicio de una oración cómo si estuviera en otra posición. También ayuda con los errores de tipeo qué incluyan mayúsculas en medio de la palabra. Las mayúsculas pueden ayudar en el reconocimiento de nombres o entidades, pero existen alternativas qué nos permiten descifrar esta información mediante otras vías.

- **Omisión de tokens con frecuencia despareja**: La falta de información o la abundancia puede producir un problema a la hora de analizar y comparar los datos. Tener mucha información de una palabra y su contexto, nos hace tratarla distinta al resto, de igual forma si no tenemos suficiente información, poco podremos decir de la misma. Por ello para normalizar los resultados, muchas veces se quitan de los análisis cualquier palabra qué aparezca escasas veces en el corpus, por no aportar suficientes ejemplos, cómo así también las palabras qué aparezcan demasiadas veces y nos opaquen la información general de las palabras de frecuencia media. Las cotas de frecuencia suelen denominarse Min df y Max df para la cota mínima y máxima correspondientemente.

Ejemplo de preproceso sin stemming con lematización:

¡Esta es una bueNa Oración para incluir en la tesis 1!!!.
esta ser buena oracion incluir tesis NUMBER

2.2.2. LDA

Finalizado el preproceso de la información, una tarea común de PLN es la fragmentación de una colección de documentos (*corpus*) en **particiones o clusters** qué muestren una relación entre aquellos qué tengan características similares.

La técnica de **clustering** [12], donde mediante una transformación vectorial del corpus podemos encontrar una relación entre los vectores qué nos indique similitudes textuales puede ser una alternativa, aunque no será suficiente en el caso de considerar documentos qué puedan pertenecer a más de un cluster, ya qué el clustering, determina una clase por documento. Incluso si quisieramos ver los términos en común o la razón de la clasificación resultante, el clustering no nos brinda dicha información de manera natural.

Una alternativa qué nos brinda estas posibilidades es Latent Dirichlet Allocation [13]. En este algoritmo cada documento ya no es etiquetado bajo una sola categoría sino qué puede verse cómo una mezcla de tópicos o categorías y su proporción de participación dentro del texto.

Los tópicos ahora no serán una simple etiqueta sino qué se tratará de una distribución sobre un vocabulario prefijado. De esta manera la conformación de los mismos está dada por un conjunto de palabras de características similares no en cuanto a su semántica, sino basadas en la coocurrencia dentro de los textos. Siendo el caso de la temática "Gatos", la misma tendrá probabilidad alta sobre los documentos en los cuales participen palabras cómo *miau*, *gatito*, *felino*, *leche*, mientras qué será potencialmente baja frente a las palabras relacionadas a términos cómo *ladrar*, *cachorro* más afines a un tópico de "Perros". Sin embargo dado el caso qué la palabra *mascota* pertenezca a un documento, ambos tópicos tendrán participación aunque con menor intensidad.

En la Figura 2.2 podemos ver el modelo LDA en su representación de platos donde se asume la distribución de tópicos θ_d precalculada cómo así también la distribución de palabras ϕ_k para cada tópico K, ambas *Distribuciones de Dirichlet* [14]. Por su parte $Z_{d,n}$ será el tópico para la *n-ésima* palabra del documento d y $W_{d,n}$ la propia palabra , única variable observada frente a las demás latentes[1]. En cuanto a las cardinalidades K aparecerá cómo número de tópicos, D cómo cantidad de documentos y N cantidad palabras en cada documento D_j. α y β serán parámetros para la conformación de las distribuciones de tópicos cómo de palabras dentro de los tópicos respectivamente.

[1]Latente o oculta no es una variable observable directamente aunque sino inferida a partir de las observadas.

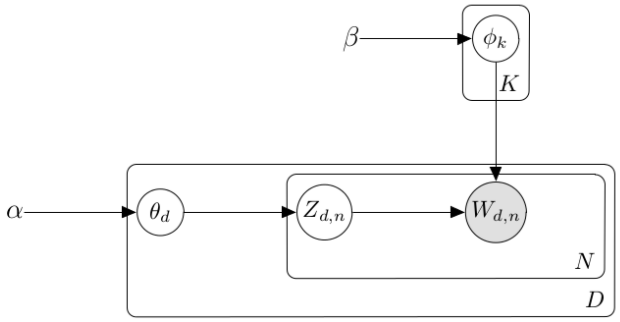


Figura 2.2: LDA en representación de platos.

2.2.3. Twitter-LDA

LDA puede ser de mucha utilidad y con resultados óptimos en la búsqueda de tópicos entre documentos de varios párrafos, pero puede ser una técnica con defectos en documentos de corta extensión cómo en nuestro caso. Es habitual pensar en una adaptación qué involucre agrupar todas las publicaciones de un usuario, en un mismo documento y luego evaluar la colección. Si bien esto puede dar mejores resultados en la formación de los tópicos, no parece ser útil en nuestro experimento ya qué no tendremos un tópico por tweet, sino una proporción de tópicos en un set de tweets del mismo usuario, lo cuál dificultaría la caracterización.

Para la resolución de esta problemática existe una implementación denominada Twitter-LDA[22] la cuál dentro de sus adaptaciones involucra la asignación de sólo un tópico por documento. Ya qué los textos cortos de microblogs tales cómo Twitter, ofrecen muy poca información, la adaptación de Twitter-LDA no buscará etiquetar a un tweet cómo una mezcla de tópicos sino qué le asignará sólo uno, similar a la tarea de clustering. Además en contraposición a LDA en su configuración típica, donde todas las palabras son involucradas en tópicos, en Twitter-LDA la participación en tópicos se restringe sólo a ciertas palabras, dejando de lado aquellas poco significativas formando una distribución propia de *background* con ellas. También, cómo se aprecia en la Figura 2.3, incorpora una variable Y con distribución de Bernoulli π, la cuál definirá si la distribución tomada para la palabra observada será la de las palabras de *background* o aquella referente al tópico pertinente.

Con una notación de plato muy similar a LDA, vemos qué α y β siguen siendo parámetros para las distribuciones θ^u y ϕ^t referentes a las palabras del usuario u y del tópico t respectivamente. También se mantienen Z cómo tópico ahora referente al usuario y W cómo variable observada. La incorporación de la distribución de las palabras de *background* θ^B y la distribución de Bernoulli π da lugar a $Y_{u,s,n}$ quien gobernará la selección de distribución para $W_{u,s,n}$ sobre el usuario u, tweet s en la palabra n, en cada caso. En cuanto a las cardinalidades, U y T representaran, la cantidad de usuarios y tópicos

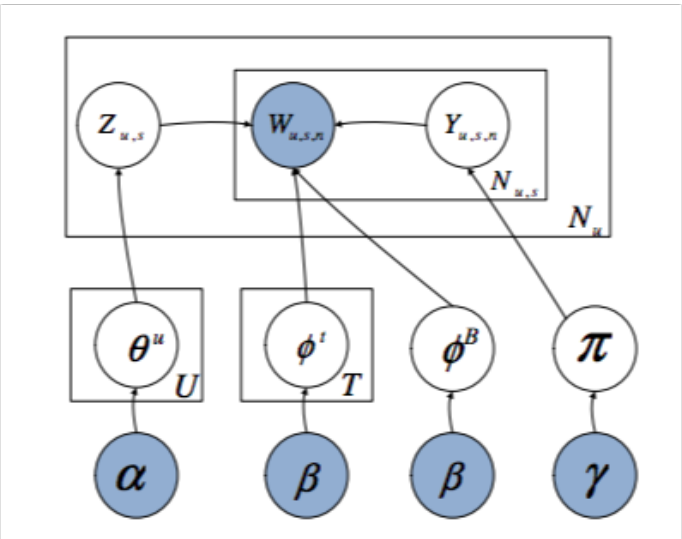


Figura 2.3: Twitter LDA en representación de platos.

respectivamente mientras qué N_u será la cantidad de tweets y $N_{u,s}$ las palabras de cada uno.

2.3. Clasificación

Clasificar es un problema qué involucra identificar a qué categoría pertenece una observación y se puede considerar una tarea de reconocimiento de patrones o reglas. En contra oposición a las técnicas de clustering, la clasificación es un procedimiento supervisado, ya qué requiere un entrenamiento previo donde será vital contar con ejemplos de observaciones y su categoría correcta para aprender los patrones qué caracterizan el dataset. De esta manera en una etapa de evaluación posterior se podrá clasificar con un criterio similar, tanto datos qué hayan sido observados, cómo así también inferir sobre aquellos qué no hayan sido vistos en el entrenamiento. Si bien es posible la clasificación azarosa o intuitiva, en tareas computacionales carece de sentido. Sin embargo es muchas veces utilizada cómo métrica de evaluación de un modelo entrenado. Es decir, será fundamental, para qué un modelo sea interesante, qué supere ampliamente una categorización aleatoria.

Comúnmente, cada observación es transformada en un vector de gran dimensión, proceso denominado caracterización o featurización, dónde cada columna será una característica qué describe una cualidad. Estas columnas llamadas *features* intentan describir

no sólo los detalles de la observación sino también las características de su contexto. Para una simplificación matemática en cuanto a la búsqueda de patrones, los vectores son transformados a su forma numérica, procedimiento denominado vectorización, y muchas veces son sometidos a procesos de normalización y modificados bajo reducción de dimensionalidad para su simplificación. Estos procesos ayudan a reducir el cálculo matemático necesario y eliminar detalles irrelevantes qué diferencien observaciones, descartando aquellas características qué sean poco informativas. Una vez entrenado el clasificador con las observaciones vectorizadas, ya estará disponible para clasificar una observación con las mismas dimensiones.

El corpus de entrenamiento de un clasificador influye directamente en la performance del modelo. Debe ser variado y mezclado para evitar caer en un modelo *subajustado*. Este tipo de modelos se caracterizan por entender muy poco sobre los detalles qué diferencian las observaciones y clasifican genéricamente. Sin embargo tampoco se puede abusar de la varianza, ya qué aprender todos los detalles de cada observación, no nos permitirá generalizar los casos similares. En este caso diremos qué nuestro modelo esta *sobreajustado*, dado qué funcionará de manera exitosa para los casos de entrenamiento pero poco podrá inferir de aquellas observaciones no conocidas.

2.3.1. Métricas

Evaluar la performance de un modelo clasificador requiere de un dataset de ejemplos anotados, es decir, cada observación con su resultado correcto. De esta manera el modelo clasifica este dataset y sus resultados son comparados con los reales. Intuitivamente uno diría qué un modelo es mejor qué otro si su cantidad de aciertos es mayor, pero no es la única métrica qué nos define la calidad de la clasificación. Las métricas más populares son:

Precision

La métrica más utilizada, refleja la cantidad de falsos positivos, ya qué se calcula para una categoría dada, cómo la división entre la cantidad de observaciones qué fueron correctamente clasificadas sobre el total de las qué fueron categorizadas bajo dicha categoría.

$$precision_{ci} = \frac{clasificados_{ci} \bigcap etiquetados_{ci}}{etiquetados_{ci}}$$

Exhaustividad o Recall

A diferencia de la precisión, recall es el nombre de la métrica qué refleja falsos negativos. Es decir para una categoría dada, se divide la cantidad de observaciones qué han sido

clasificadas correctamente bajo esta categoría, sobre la cardinalidad real de la categoría.

$$recall_{ci} = \frac{clasificados_{ci} \bigcap etiquetados_{ci}}{clasificados_{ci}}$$

F1 score

Cómo una combinación de ambas, ésta métrica es el objetivo de la mayoría de los modelos. Mediante la fórmula F1 se refleja el desbalance entre la Precisión y Recall de un clasificador, cualidad muy negativa para cualquier modelo. Su fórmula es conocida cómo Media Harmónica y se obtiene de la siguiente manera:

$$f1score = 2.\frac{1}{\frac{1}{recall} + \frac{1}{precision}} = 2.\frac{precision.recall}{precision + recall}$$

2.4. Clasificacadores

La tarea de clasificación depende particularmente de nuestros datos, es decir no existe una forma de clasificación genérica qué sea óptima para la totalidad de los casos. Por lo tanto dependiendo de nuestro dataset y tipo de clasificación, se deberá utilizar el más apropiado. En esta sección se detallan algunos de ellos y sus características principales.

2.4.1. Random Forest

Random Forest [15] es un clasificador qué involucra una decisión por votación de una serie de árboles creados por el propio algoritmo. Éstos árboles de decisión son creados a partir de los datos de entrenamiento, donde sus hojas finales indicarán una categoría de pertenencia. Simultáneamente las observaciones van *descendiendo* por los múltiples árboles hasta llegar a una categoría determinada. Cuando todos los caminos hayan concluido, se etiquetará la observación con aquella categoría qué sea mayoritaria en la votación, es decir, aquel resultado qué haya sido más veces encontrado cómo final del camino en los *descensos*. En esta sección se detallará el procedimiento y las ventajas de los clasificadores tipo Random Forest.

Creación de arboles de decisión

Los clasificadores de RF, crean a partir del dataset múltiples árboles de decisión. Pero poco sentido tendría si todos ellos fueran de características similares, ya qué al hacer una consulta en uno de ellos la respuesta sería idéntica al resto.

Por esta razón en la creación de los árboles de decisión se aplica una idea de *Bagging* (Bootstrap Aggregation). Esta técnica no es más qué la creación de múltiples conjuntos de entrenamiento a partir de uno solo, donde se aplicará un muestreo de los datos de

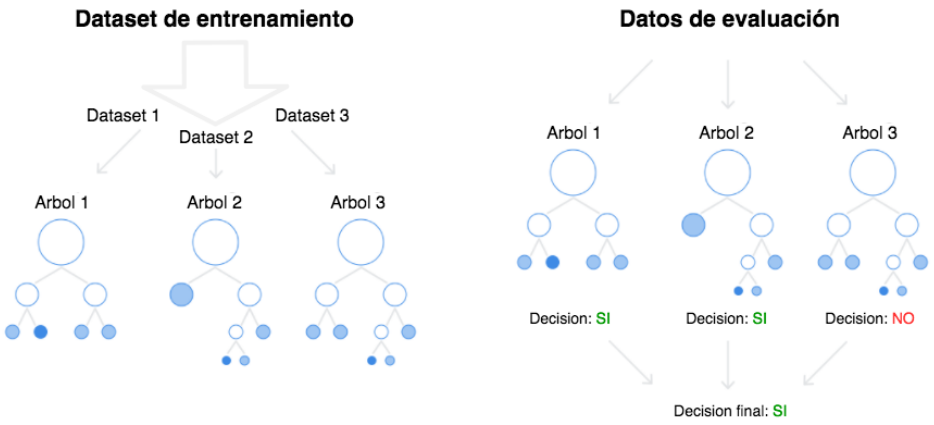


Figura 2.4: Creación y evaluación sobre arboles de decision de un clasificador RF.

manera aleatoria contemplando cobertura en la selección. Es decir, en el caso de qué quisiéramos obtener m árboles de decisión, dado un conjunto de entrenamiento estándar D de tamaño n, generaríamos m nuevos conjuntos de entrenamiento $D_1 ... D_m$, cada uno de tamaño $n' \leqslant n$, mediante muestreo uniforme y con reemplazo. De esta manera tendríamos m muestras distintas del dataset donde se estima qué el 63 % de los datos serán ejemplos únicos mientras qué el resto serán duplicados. Se logra así entonces llegar a construir los m árboles con información potencialmente distinta dado qué provienen de un conjunto distinto.

A la hora de realizar una clasificación otra técnica denominada Boosting es aplicada. Boosting es un meta-algoritmo de aprendizaje automático qué reduce el sesgo y varianza en un contexto de aprendizaje supervisado. Basado en el cuestionamiento planteado por Kearns y Valiant propone crear un clasificador robusto a partir de múltiples clasificadores débiles. Un clasificador débil está definido cómo un clasificador qué tiene mejor desempeño qué uno aleatorio aunque dista bastante de los resultados admisibles para tenerlo en cuenta. En contraste, un clasificador robusto será aquel qué tiene un mejor desempeño aproximandose, con su clasificación, fuertemente a las verdaderas clases. En RF los clasificadores débiles serán los árboles de decisión creados mediante Bagging, donde cada uno sabrá una porción de información. Dichos clasificadores darán lugar al clasificador general, quién dictará el resultado final mediante el análisis de cada árbol cómo podemos ver en la Figura 2.4.

Resumen

RF es ideal para dataset donde los datos son pocos pero muy variados. Cómo ventaja con respecto a otros clasificadores posee un buen manejo de los vectores de gran dimensión. No es necesaria la reducción de dimensionalidad o selección de features. Por otro lado también evita el *overfitting* ya qué el modelo de votación de varios árboles de decisión reduce la varianza y descarta los detalles insignificantes. Sin embargo esto puede llevar a un modelo muy libre o con *underfitting* qué puede afectar directamente en las métricas. Cómo una potencial desventaja posee escasa parametrización y tiene un bajo rendimiento en tareas de Regresión Logística.

2.4.2. Support Vector Machines

Dentro de los clasificadores lineales denominados así debido a su categorización mediante la separación del universo bajo un hiperplano, se encuentran las SVM [16]. Centrando su complejidad en la búsqueda del mejor plano qué divida de manera optima las categorías requieren de un universo de datos linealmente separables. Dado el caso qué esta condición no se cumpla en el universo original, será necesaria la transformación a un universo alternativo con mayor dimensionalidad. Esta transformación impacta directamente cómo una desventaja a la hora de vectorizar las observaciones, ya qué debemos seleccionar de manera inteligente los *features* o en caso contrario conllevará un computo mucho más extenso para conseguir la convergencia. La mutación de universos es conocida cómo *kernel trick* y las funciones qué realizan la conversión son denominadas *kernel*. En esta sección se explicará estos conceptos en detalle para entender el funcionamiento de este tipo de clasificadores. Se incluirá además información respecto a la configuración posible de los modelos para los distintos escenarios.

Búsqueda del mejor hiperplano

La búsqueda de cualquier hiperplano qué distinga los datos, no es suficiente ya qué esto podría determinar una mala generalización de la clasificación y decaer en un sesgo qué involucre subajuste. Por lo tanto es necesario encontrar el hiperplano qué mejor discrimine los datos con una frontera de decisión lo más alejada posible de las muestras. Dados datos de entrenamiento $\{x_i, y_i\}$ con i=1,..,j. Sea $x_i \in \mathbb{R}^{\kappa}$ el vector numérico proveniente de cada observación y cada $y_i \in$ -1, 1 la representación de la clase perteneciente. Diremos qué los datos son linealmente separables si existe un hiperplano separador qué los distinga, de la forma:

$$xw + b = 0 \ (x \in \mathbb{R}^{\kappa},\ b \in \mathbb{R})$$

El márgen del hiperplano se define cómo la suma de las distancias d_+ y d_- donde cada una se define cómo la distancia fronteriza de la muestra más cercana de cada clase.

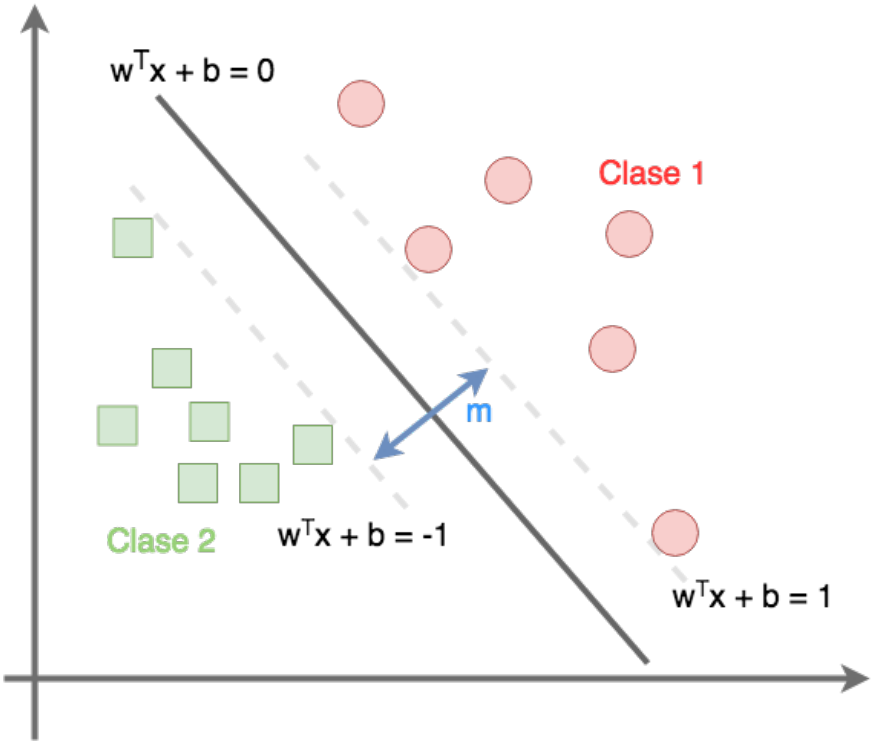


Figura 2.5: SVM y la mejor frontera de decisión, tan lejos de los datos cómo sea posible.

Quedan entonces caracterizados los hiperplanos tangentes por las siguientes ecuaciones.

$$\begin{aligned} \mathbf{x}_i\mathbf{w} + b \geq +1, \quad & y_i = +1 \\ \mathbf{x}_i\mathbf{w} + b \leq -1, \quad & y_i = -1 \\ \equiv & \\ y_i(\mathbf{x}_i\mathbf{w} + b) - 1 \geq 0, \qquad & \forall i \end{aligned} \tag{2.1}$$

$$m = d_+ + d_- = \frac{|(-b+1) - (-b)|}{\|\mathbf{w}\|} + \frac{|-b - (-b-1)|}{\|\mathbf{w}\|} = \frac{2}{\|\mathbf{w}\|} \tag{2.2}$$

Dado qué en los puntos más cercanos a la frontera de cada clase vale la igualdad, la distancia geométrica entre estos dos planos será $\frac{2}{\|w\|}$. Por lo tanto maximizar el márgen entre las fronteras quedará reducido a minimizar la norma euclidea $\|w\|$

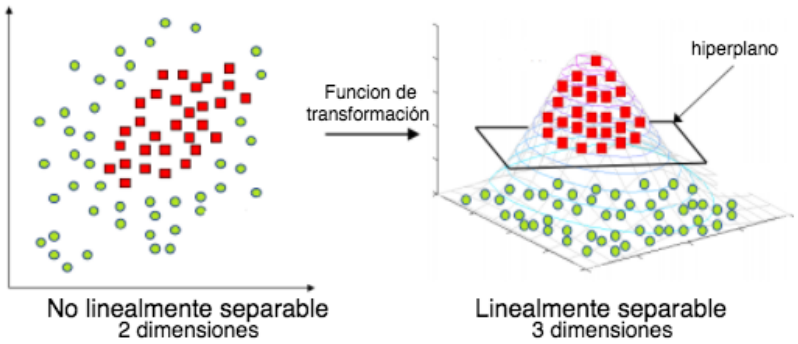


Figura 2.6: Ejemplo de Kernel-Trick de 2D a 3D

Truco de Kernel

El Truco de Kernel más conocido por su nombre en inglés *kernel-trick* [17] no es más qué una transformación de dimensión. Dada la necesidad de qué los puntos en el plano sean linealmente distinguibles por la condición qué tienen las SVM's, aplicar una transformación mediante una función puede darnos otra perspectiva de los datos qué nos permita lograr encontrar el hiperplano necesario. Kernel es el nombre qué se le da a este mapeo de cada punto a un nuevo universo de mayor dimensión. Poco sentido tendría si fuera una función constante, pues el problema seguiría persistiendo en el nuevo universo. Es por esta razón qué entre los kernel más populares encontramos funciones lineales, polinomios cómo también funciones de base radial (RBF). Más allá de las funciones genéricas, también es posible aplicar cualquier tipo de función personalizada para la resolución en particular de un problema de datos no separables. Vemos en la Figura 2.6 cómo es aplicada una función cuadrática cómo kernel sobre un plano de 2 dimensiones.

Parametrización

La gran variedad de aplicaciones en las cuáles se utilizan las SVM's cómo clasificador hace qué deban ser genéricas pero sin perder rendimiento. Para ello cuentan con parámetros de ajuste qué cambiarán sutilmente el algoritmo según la necesidad. La búsqueda de los parámetros puede llegar a ser una tarea compleja y poco intuitiva aunque existe la posibilidad de conseguir los mejores valores para cada uno de forma automática. Grid-search es un algoritmo qué se encarga justamente de realizar ésta búsqueda. Mediante una lista de posibles asignaciones para cada parámetro, implementa todas las combinaciones y aplicando la técnica de validación cruzada, devolverá cómo resultado el mejor modelo y sus parámetros de construcción.

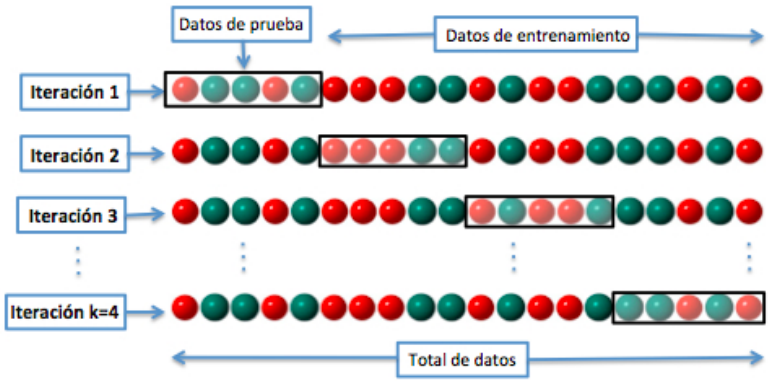


Figura 2.7: Procedimiento de validación cruzada con k iteraciones

La validación cruzada o cross-validation [18] es una técnica utilizada para evaluar los resultados de un modelo estadístico y garantizar qué son independientes de la partición entre datos de entrenamiento y prueba. Como podemos ver en la Figura 2.7 consiste en un procedimiento iterativo, en el cuál se calculan las métricas de evaluación sobre diferentes particiones, todas provenientes del mismo dataset. Los resultados de cada iteración son promediados en un resultado final y ésta será la métrica considerada para el modelo en cuestión.

Entre los parámetros qué gridsearch puede ajustar sobre este tipo de clasificadores de manera automática se encuentra el denominado Parámetro de Margen Débil representado cómo C en los argumentos. Cómo intuitivamente su nombre lo indica, este indicador, dejará la posibilidad de qué la frontera de decisión no sea tan estricta, tomando algunas observaciones etiquetadas con una clase, en el territorio de la otra.

Reduciendo de esta forma notablemente el *overfitting*. En las Figuras 2.9 2.10 y 2.11 se detalla la comparación de entrenamiento y evaluación de 3 modelos con la misma información y configuración, sólo variando este parámetro.

Vemos entonces qué reducir el sobreajuste es importante pero una frontera de decisión muy holgada, cómo sería en el caso de un C demasiado grande, tampoco es una buena opción. Cómo es de esperarse, los resultados obtenidos con gridsearch fueron para el C qué mayor rendimiento tuvo, disminuyendo al máximo posible el sobreajuste.

Resumen

Con resultados similares y muchas veces superadores a los de las redes neuronales, la gran adaptabilidad de las SVM's a los diferentes dataset y su sencilla configuración, lo hacen uno de los clasificadores más utilizados. El rendimiento de estos clasificadores

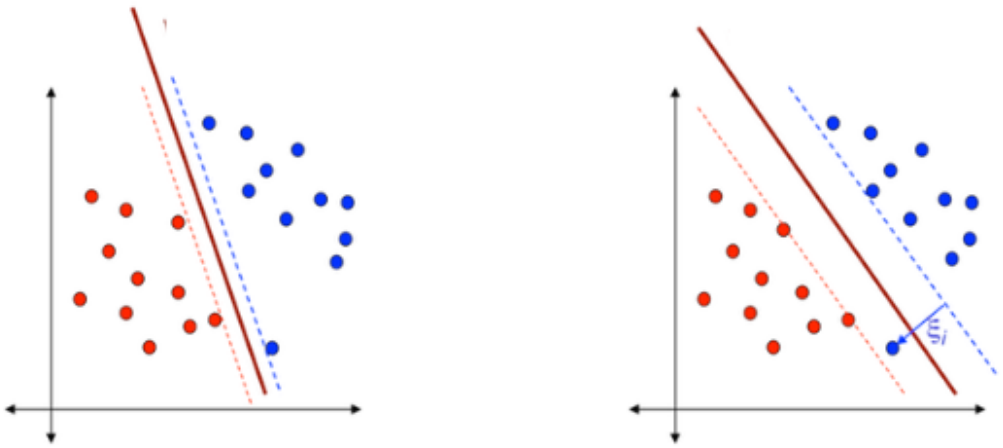


Figura 2.8: Muestra de cómo debilitando el margen se puede encontrar una frontera más amplia.

está muy ligado a la correcta selección del *kernel* qué mejor se adapte a la distribución del dataset. Sin embargo, también puede llegar a ser trascendental la mejora, ajustando cada uno de sus múltiples parámetros dependiente también de la distribución de los datos de entrada. Cómo desventaja de este clasificador se encuentra la complejidad de la clasificación multiclase. Si bien existen artilugios qué permiten efectuarla, la clasificación de las SVM's es naturalmente binaria, ya qué se trata de un algoritmo de clasificación lineal.

2.5. Trabajos Relacionados

Prediction of User Retweets Based on Social Neighborhood Information and Topic Modelling. [19]

Publicación íntimamente relacionada con este trabajo, propone la predicción de retweets con modelos particulares para cada usuario. Mediante el análisis del comportamiento de los usuarios allegados al objetivo, determina si éste hará *retweet* sobre una publicación o no. Cómo una evolución de esta idea, este trabajo pretende generalizar el modelo para todos los usuarios evaluando no sólo el comportamiento particular, sino de la población masiva.

Collective Influence Algorithm to find influencers via optimal percolation in massively large social media [20]

Implementación lineal del algoritmo de *Collective Influence*(CI) propuesto por Morone, Makse para encontrar el mínimo conjunto de influencers en una red vía *percolación óptima*. No fue aplicado en este proyecto aunque es una potencial mejora para la selección inteligente de la cantidad de influenciadores en trabajo futuro.

A Comparative Analysis of Community Detection Algorithms on Artificial Networks. [21]

Explicación y comparativa de los algoritmos de detección de comunidades más populares mediante el análisis de resultados tomando cómo benchmark el grafo de *Lancichinetti-Fortunato-Radicchi*, su tiempo de ejecución y efectividad. Determinando los mejores y peores escenarios para cada algoritmo, proponen una técnica para determinar cuál es el más adecuado para la resolución de un problema.

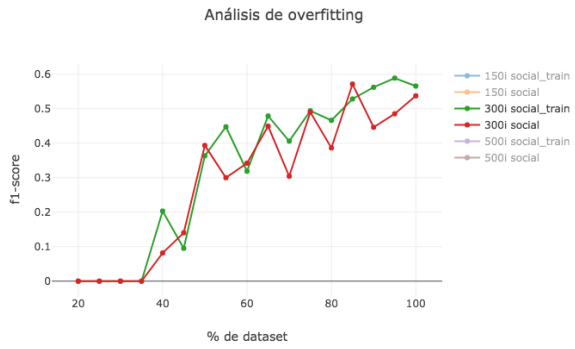


Figura 2.9: Comparación de evaluación demostrando underfitting con C=0.1

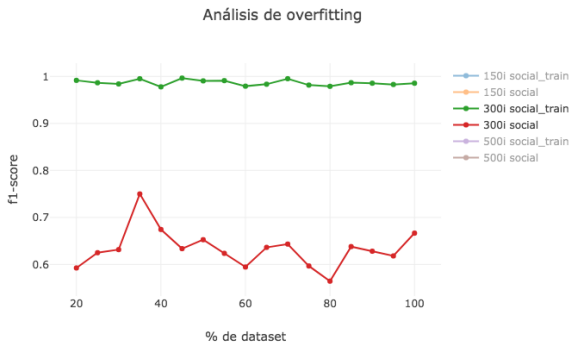


Figura 2.10: Comparación de evaluación demostrando overfitting con C=100

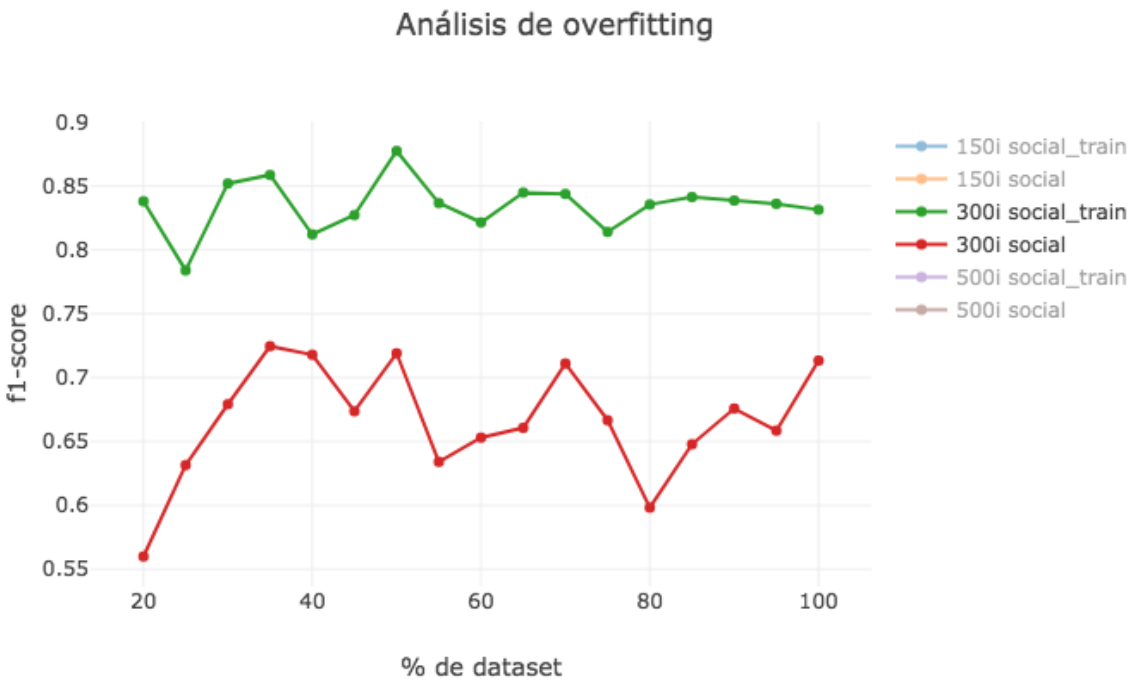


Figura 2.11: Comparación de evaluación de dataset de entrenamiento y dataset ajeno mediante un C=1, sobreajuste disminuido y buenos resultados.

Capítulo 3

Herramientas

Saber las herramientas qué se han utilizado puede ser útil tanto para la replicación de resultados cómo para una futura mejora. Por eso a continuación se detallan las herramientas de software qué han estado involucradas en los experimentos discriminadas por su utilidad en cada etapa.

Para mayor información se citará cada herramienta con su documentación oficial correspondiente.

3.1. Datos y análisis

3.1.1. Tweepy

Esta herramienta permite la extracción de información de Twitter. Se puede conseguir tanto los tweets generados por un usuario cómo también otros datos de utilidad cómo sus seguidores o sus seguidos y hasta su detalle de perfil cómo ubicación y demás. Cuenta con limitaciones de peticiones en ventana de tiempo y es necesaria la autenticación con credencial de desarrollador en twitter para usar la API.

3.1.2. SQLAlchemy

SQLAlchemy es el conjunto de herramientas de Python SQL y ORM(Object Relational Mapper) qué ofrece a los desarrolladores de aplicaciones un acceso simple y Pythonico a una Base de Datos de tipo relacional sin la necesidad de conocer la sintaxis de SQL.

3.1.3. Jupyter

Jupyter propone un entorno de desarrollo interactivo similar a ipython pero con la posibilidad de guardar las ejecuciones en un estado visible para ser presentado de forma

web. También permite la incorporación de texto Markdown y contenido multimedia cómo imágenes, lo qué mejora notablemente la presentación de la información.

3.2. Grafos

3.2.1. NetworkX

NetworkX es una librería de Python para el estudio de estructuras de datos tipo Grafo. De manera sencilla se pueden generar, almacenar y cargar modelos rápidamente para un futuro análisis mediante su amplio repertorio de algoritmos popularmente conocidos. Se utilizó en nuestro caso para la representación de la red de usuarios mediante un Grafo dirigido, el cuál fuera posteriormente analizado en cuanto a centralidad y afinidad.

3.2.2. Python-igraph

Adaptación a Python de una libreria popular para la manipulación de Grafos en el lenguaje R. Con una performance superior a NetworkX debido a su implementación y una utilidad y usabilidad muy similar fué nuestra opción elegida para la tarea de análisis de la red.

3.3. Aprendizaje automático y PLN

3.3.1. scikit-learn

Paquete de Python para la realización de tareas de Machine Learning. Ofrece algoritmos muy parametrizables con aplicación en Clasificación, Clustering, Preprocesamiento, Regresion, Reduccion de dimensionalidad entre otras. En este trabajo fue utilizado para la evaluación de los modelos mediante sus *métricas o scores*, *validación cruzada* y *ajuste de parámetros* guiado. También se utilizaron algunos de sus modelos de clasificación.

3.3.2. NLTK

NLTK(Natural Language Tool Kit) es una librería de Python con algoritmos qué facilitan las tareas cotidianas del procesamiento de lenguaje natural, fundamentalmente en el preproceso de los datos. La *tokenización*, *limpieza de caracteres especiales* y *stemming* son algunas de las funciones qué han sido de utilidad en este trabajo.

3.3.3. Gensim

Libreria open source para el modelado en espacio vectorial de tópicos. Usando SciPy y NumPy cómo base de cálculo y Cython para mejorar en performance, permite el análisis de una gran colección de textos mediante algoritmos incrementales de alto rendimiento. En este proyecto fue utilizado para la implementación de LDA tomando cómo documentos los textos de twitter.

3.3.4. Spacy

Libreria más qué completa en cuanto las tareas de PLN (Procesamiento del Lenguaje Natural). Complementando los algoritmos qué ofrece NLTK, suma la *Lematización*, y la posibilidad de diagramar *Pipelines* en su ultima versión. Con un modelo preentrenado muy aceptable en casi todos los idiomas predominantes incluyendo entre ellos el Español, fue utilizado en nuestro caso para aplicar *Lematización* sobre el vocabulario nativo.

3.4. Visualización

3.4.1. Gephi

Gephi es una aplicación open-source de escritorio desarrollada en Java, muy útil visualmente para el análisis de Grafos. Con algunos algoritmos de coloreo y muestreo fácilmente interpretable por el usuario permite el entendimiento visual de una red. A pesar de ser de gran utilidad, cuenta con limitaciones importantes a la hora de representar grandes grafos, por eso a pesar de los múltiples intentos fue muy poco utilizada en el desarrollo de este trabajo.

3.4.2. pyLDAvis

Librería de Python adaptada del paquete de R LDAvis. Facilita la tarea de presentación de los temas extraídos de un algoritmo tipo LDA, especialmente integrable y compatible con los resultados de la libreria gensim. Los temas se presentan en una modalidad tipo "bubblesz cuenta con ajustes qué permiten la interpretación profunda de los datos. Debido a su presentación tipo HTML es una gran herramienta para compartir los resultados y visualizarlos en cualquier navegador web.

3.4.3. Plotly

Plotly es la evolución de los gráficos en Python. Se trata de una libreria qué hace realmente sencilla y humanizable la tarea de generación y adaptación de graficas complejas con hasta 3 dimensiones. Además cuenta con un resultado tipo HTML de salida qué

permite una interactividad muy útil para aislar o comparar los datos. Plotly es también una plataforma web Plot.ly qué permite almacenar, evaluar, modificar y compartir de manera onlin los resultados.

Capítulo 4

Conjunto de datos

Sabemos qué las redes sociales hoy en día manejan grandes volúmenes de datos, muchos de ellos útiles para tareas de marketing y publicidad. Por esa razón pocas de ellas comparten su información al menos de manera gratuita. Twitter por su parte ofrece esta posibilidad mediante una API qué permite, aunque con sus pertinentes reestricciones, obtener información de su red.

Dado esto, y sumado la sencilla estructura qué presenta la red social, se eligió para realizar los experimentos competentes al proyecto. Si bien el dataset inicial fue heredado, es parte de este trabajo, complementarlo con mayor información. Veamos algunas características y cómo fue su evolución.

Conformación dataset inicial

Según la planificación de nuestro modelo deberíamos obtener de la red no sólo los usuarios y su relación (amistad, seguimiento, etc), sino también qué contenido fue aceptado por cada usuario cómo así tambien, cuál fue creado por el mismo.

Considerando la relación seguir cómo un vinculo qué une direccionalmente dos usuarios seguidor y seguido, tendremos resuelta la información vinculante entre los usuarios. En cuanto a la aceptación será tomada en cuenta la acción de retweet propuesta por Twitter, qué no es más qué compartir el contenido de otros en tu linea de publicaciones denominada *timeline*. Con respecto a la autoría de contenido, la API resuelve esto en un campo de la respuesta brindando la identificación del autor, quien puede o no coincidir, cómo en el caso de los retweets, con el dueño del *timeline* donde pertenece.

El dataset de partida fue creado mediante el uso de `Tweepy` a través 9 cuentas rotativas para evitar censuras de twitter. El universo de usuarios objetivo fue creado en dos pasos donde en una primera instancia se buscó lograr un gran volumen para luego ser reducido con el fin de obtener una mayor homogeneidad en las relaciones.

Para la conformación del universo inicial de usuarios se partió de un usuario "semilla" u_0 y un conjunto $U_0 = \{u_0\}$ realizando 3 iteraciones sobre el siguiente procedimiento:

- Buscar todos los usuarios seguidos por los usuarios en U_i
- De ese grupo, filtrar aquellos qué tengan más de 40 seguidores cómo así también más de 40 cuentas seguidas.
- Incorporar el grupo resultante junto con sus relaciones en un grafo extendido U_{i+1}.

Como resultado con $U := U_3$, se obtuvieron $2,926,181$ usuarios y más de $10,000,000$ de relaciones. Continuando con la idea de conseguir homogeneidad se aplicaron los filtros de la siguiente manera.

- Se creó un subset de usuarios S cómo semilla determinado por aquellos nodos pertenecientes a U qué tengan un grado de salida mayor a 50.
- Para cada uno de ellos se agregaron aquellos 50 usuarios más afines de entre sus seguidos, donde la afinidad entre dos usuarios estuvo dada cómo una medida de *ratio* entre el número de usuarios seguidos por ambos y los seguidos sólo por uno de ellos.
- Se repitió este último paso para cada nuevo usuario agregado, hasta qué no existieron más adiciones.

De esta manera se almacenó la información social en un grafo resultante con $5,180$ nodos representando usuarios y $261,005$ aristas qué representarán la relación entre ellos.

En cuanto al contenido, fueron descargadas las publicaciones entre el 25/2/2014 y 17/4/2017 provenientes del *timeline* de cada uno estos $5,180$ usuarios, exceptuando aquellas denegados por la API por restringir su perfil voluntariamente. Se obtuvo así un contenido resultante de $2,062,185$ tweets entre los cuales registraron $497,555$ retweets.

Dimensiones y características significativas

Es fundamental en cualquier tarea, en la cuál intervenga un dataset de información, realizar de manera previa , un análisis de los datos. Entender la distribución de los mismos puede influir en las heurísticas qué definan decisiones importantes dentro de los algoritmos. Por este motivo en esta sección incluimos algunos datos significantes en cuanto al dataset de partida.

Twitter posee una gran cantidad de usuarios, nuestro reducido dataset contiene sólo $5,180$ de ellos. De igual forma, entre ellos se puede encontrar un universo variado. Algunos usuarios, probablemente promocionales, con un flujo extremo de información con publicaciones constantes, y otros qué sólo usan la red a modo informativo cuyos timeline yacen vacíos o con escasas publicaciones. En las Figura 4.1 vemos la distribución de Tweets y Retweets del dataset completo respectivamente.

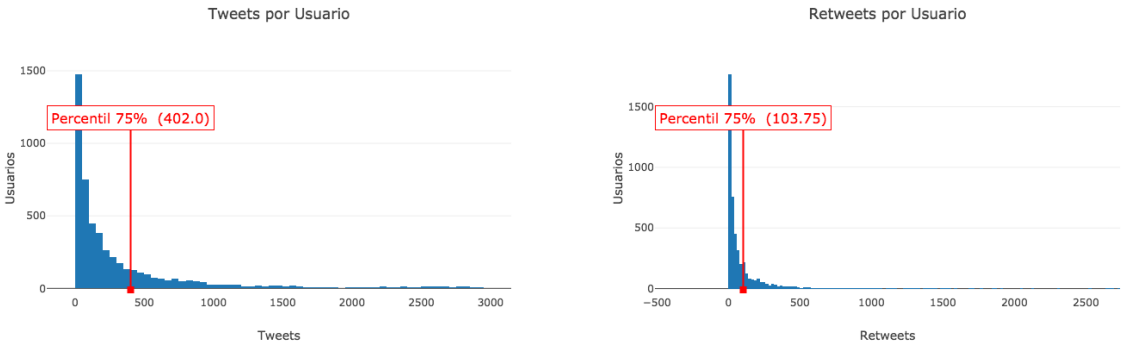


Figura 4.1: Histogramas de tweets y retweets por usuario

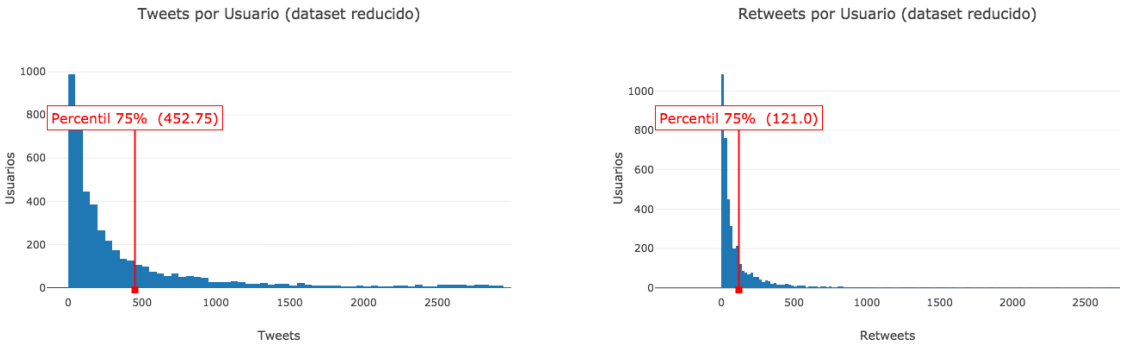


Figura 4.2: Histograma de tweets y retweets de usuarios con más de 10 tweets y retweets

En particular dado qué muchos de nuestros usuarios son inactivos se ha decidido exceptuarlos eliminando también su contenido relativo. La cota mínima de RT's para qué un usuario no sea descartado fue fijada en 10 unidades. Como resultado de esta operacion, se ve en la Figura 4.2 cómo se elevó el percentil 70 tanto en los TW cómo en los RT de cada usuario.

Dado qué la definición de las clases de relevancia explicadas en detalle en la siguiente sección, están relacionadas con la distribución de la aceptación qué tienen los tweets, analizamos su histograma y percentiles.

Como es de esperarse, mucho del contenido sólo pertenece al *timeline* de su autor, esto hace qué la distribución sea muy abundante cercana a 1. Cómo podemos ver en la Figura 4.3 el percentíl 90 no escapa a lo previsible y nos obliga a realizar un descarte de aquellos tweets insignificantes antes de cualquier análisis. Descartando aquellos con menos de 3 retweets, aumenta dicho percentil, llegando ahora hasta los 13 RT's, determinando así una potencial cota de división entre las clases de relevancia.

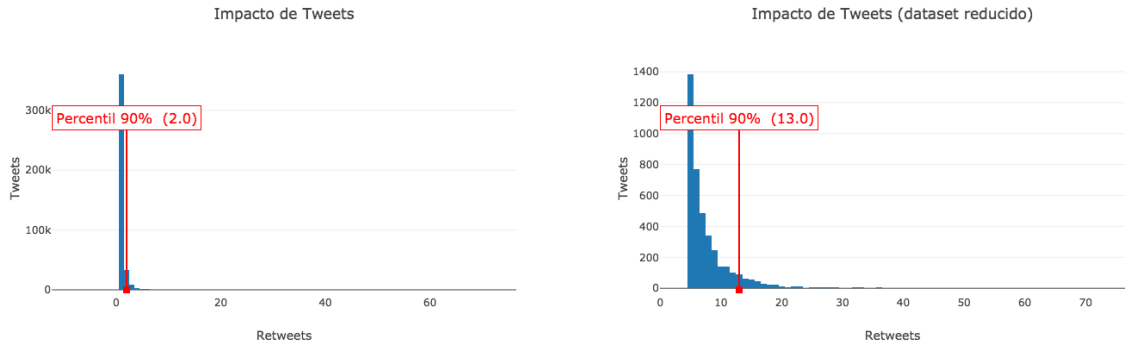


Figura 4.3: Histograma de retweets por cada tweet antes y después del filtrado de aquellos qué hayan sido considerados por más de 3 usuarios.

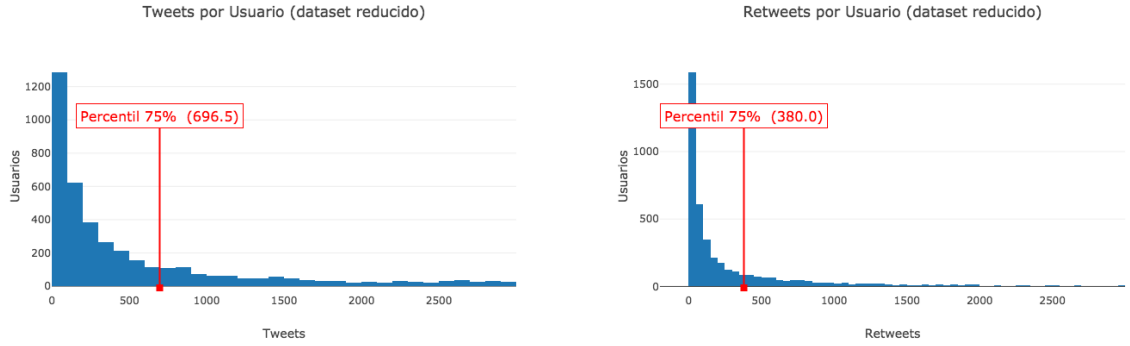


Figura 4.4: Tweets y Retweets por usuario sobre el dataset nuevo.

Complemento de información

Durante la tarea de complementación del dataset se decidió mantener la estructura social y sólo adicionar contenido de sus timelines. El scrapping de contenido fue reanudado entonces sólo sobre los 5, 180 usuarios registrados y sus relaciones se mantuvieron intactas. A partir del día consecutivo a la ultima fecha registrada 18/04/2017 hasta el 21/02/2018 se registraron nuevos tweets y su metadata.

Obtuvimos de esta manera una suma total de 5, 087, 862 tweets con 1, 519, 194 retweets sobre los mismos. Con un aumento de más de 3 millones de tweets y más de 1 millón de RT los histogramas anteriores se vieron afectados cómo se muestran en las figuras 4.4 y 4.5

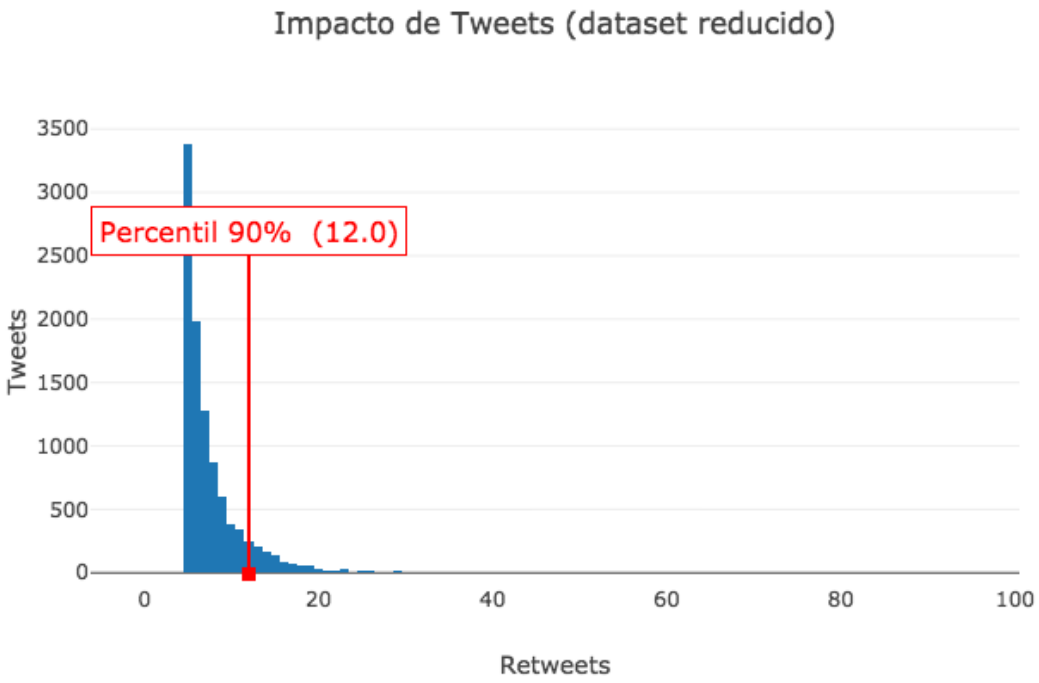


Figura 4.5: Histograma de retweets por cada tweet filtrado sobre el dataset nuevo.

Capítulo 5

Modelo predictivo de relevancia

Con el objetivo de realizar una predicción del impacto qué causa un tweet en el espacio reducido de nuestro dataset, se entrenará un modelo basado en la aceptación qué causó sobre los influencers para determinar si será relevante o insignificante para el resto de los usuarios. Basando esta idea en un primer modelo social puro, luego se evaluará la alternativa de agregar técnicas de PLN en un modelo con análisis de contenido dónde no sólo se tenga en cuenta la actividad de los influencers. Durante el desarrollo de este trabajo fueron probadas varias alternativas para la conformación del modelo final qué nos permita realizar la clasificación o predicción de la mejor manera.

Se puede visualizar en la Figura 5.1 la arquitectura general del proyecto, empezando desde un scrapping rotativo el cuál almacene los datos sociales ordenados en una base de datos propia la cuál será consultada tanto para la detección de influenciadores cómo así también por el clasificador para determinar la relevancia de los tweets. A su vez se integra opcionalmente un módulo de PLN para determinar la información social de importancia para el clasificador.

A continuación se detallarán los diversos ajustes y modificaciones qué se fueron aplicando para obtener una mejora relativa de rendimiento sobre los experimentos, cuyos resultados serán presentados en la siguiente sección. Se desarrollarán los métodos aplicados para la selección de influencers y detección de comunidades. También se explicitará qué características fueron tomadas en cuenta y bajo qué clasificadores fueron evaluadas cómo así también el preproceso utilizado en el modelo con análisis de contenido y los criterios para la determinación de heurísticas de relevancia.

5.1. Selección de influenciadores

La estructura de datos de tipo grafo, busca abstraer una red de relaciones compleja con el objetivo de poder ser interpretada para extraer la mayor información posible de ella. Junto con la información relacional surgen ideas de optimizar la propagación de mensajes

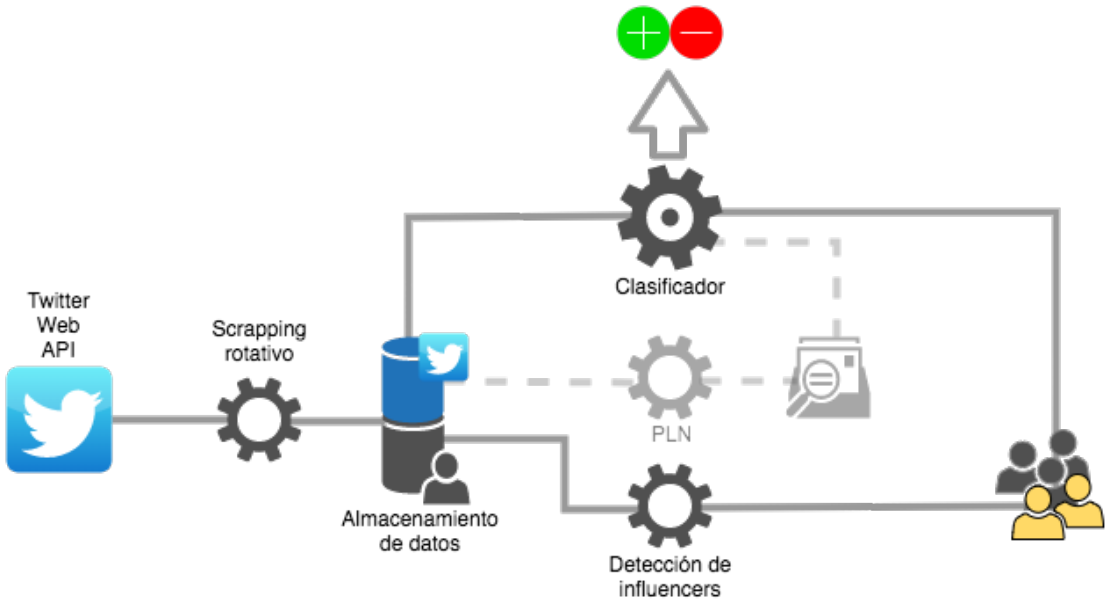


Figura 5.1: Arquitectura del proceso con la posibilidad de la incorporación de un modulo de PLN en el análisis de contenido

con fenómenos tipo *epidemia*. Este tipo de análisis busca determinar la menor cantidad de pasos necesarios para lograr la cobertura del grafo. Para la propagación óptima es vital comprender qué nodos son determinantes para el resto de la población. Estos nodos influenciadores tienen características qué hacen qué su actividad dentro del grafo tenga repercusión en gran porción del total de los nodos. Será entonces fundamental contar con su colaboración en el caso de querer llevar un mensaje a la totalidad de la población. Como se puede ver en la Figura 5.2 la tarea de detección de influenciadores puede ser compleja y dependiente del problema en particular. Por este motivo aquí se detallan los métodos utilizados en este trabajo.

5.1.1. Aleatoria

Como mencionamos antes, la aleatoriedad es una técnica utilizada para determinar la cota menor de rendimiento y así poder comparar nuestro modelo objetivo esperando obtener resultados superiores. La detección aleatoria de influenciadores no es más qué elegir al azar si un nodo será o no tenido en cuenta cómo influenciador. Este tipo de selección devendrá en un modelo aleatorio qué ocasionalmente puede tener una performance aceptable aunque en la mayoría de los casos se espera bastante baja o inclusive

nula. Por este motivo, para considerar las métricas de este modelo se promediará entre las métricas de 10 selecciones de usuarios al azar por cada experimento qué se realice. De esta manera se obtiene un modelo aleatorio más realista.

5.1.2. Por centralidad

Una selección más inteligente de los nodos nos lleva intuitivamente a pensar en aquellos nodos con mayor grado o valencia, es decir, aquellos nodos qué tengan mayor cantidad de enlaces en un paso a otros. Esta selección puede no ser suficiente, dado qué existen casos qué un nodo interconecta cómo único camino 2 componentes conexas del grafo mediante su nodo más central. Vemos qué el nodo en principio no es influenciador porque posee escasas relaciones. Sin embargo si lo vemos desde el punto de vista de su alcance de mediante dos pasos, posiblemente tenga una influencia considerable en la población debido a qué se relaciona con los influenciadores más importantes de cada componente conexa. Algoritmos cómo Katz o el popular PageRank de Google, explicados en la sección 2, resuelven este tipo de desajustes con un método iterativo de ponderación de relevancia. De esta manera tendremos entonces aquellos nodos qué conocen a una gran porción del grafo, o aquellos qué conozcan nodos de alta relevancia cómo nuestros influenciadores.

5.1.3. Por actividad y centralidad

Es vital para obtener los mejores resultados entender los detalles de cada problema. Analizando la influencia en una red social cómo Twitter, vemos qué no sólo importa la ubicación dentro del grafo, característica qué ya queda resuelta por la centralidad del nodo sino qué también es importante la actividad o aceptación qué tenga un nodo en el aprovechamiento de la red. Aquel nodo qué tenga una volumen considerable de nodos adyacentes pero nunca publique contenido o cuando lo haga no cause ningún impacto, no será indispensable para la comunicación. Por otro lado tenemos aquellos puntos de popularidad promedio pero con impacto frecuente y asegurado en sus pocos seguidores. Por esto el ajuste del índice de actividad es importante y debe ser contemplado cómo una medida más de influencia. En nuestro caso la actividad o aceptación, será determinada por la cantidad de tweets qué generó el usuario y fue *retweeteado* por algún seguidor. Quedará determinada la influencia en nuestro modelo cómo una combinación de un 50 % de aceptación y un 50 % relacional mediante el cálculo de PageRank y la valencia del nodo, y se eligirán cómo influenciadores los qué posean mayor exponente en estos valores.

5.2. Predicción social pura

Solo con información social mediante el análisis de grafos, el modelo social puro fué el primer experimento. Aquí se aplicaron los ajustes en reducción de dataset, mencionados

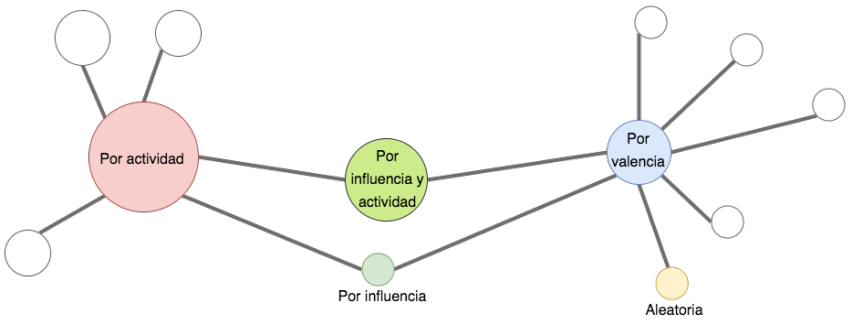


Figura 5.2: Posible selección de influenciadores con los diversos métodos donde la dimensión del nodo indicará la actividad del mismo.

en el capítulo 4, y determinadas heuristicas para la definición de las clases de relevancia de un tweet. Estos procesos cómo la extracción de características o detección de comunidades serán explicados en esta sección. Finalmente veremos los clasificadores utilizados y sus rendimientos respectivos.

5.2.1. Ajuste de dataset

Los clasificadores son muy sensibles al *ruido* en la información, por eso es fundamental intentar qué los datos de entrenamiento sean los más claros y reveladores posibles. Luego de un muestreo de los datos, se vió un desbalance en la información de cada clase. Esto afecta directamente en el entrenamiento muy sesgado y posteriormente en su rendimiento. Es por esto qué se decidió reducir el conjunto de datos descartando aquella información poco informativa de la siguiente manera:

En cuanto a la selección de usuarios fueron descartados del corpus los usuarios y toda su información relacionada, de aquellos qué no tenían cómo mínimo 10 retweets en su timeline. A su vez también fue aplicada una selección de tweets dejando sólo aquellos qué sean informativos en el análisis. Consideramos poco informativos aquellos sin repercusión, en nuestro caso deberán sido retweeteados al menos 3 veces por cualquier usuario para ser tenidos en cuenta. También fue descartado cualquier tweet qué no estuviera escrito en idioma español. Esto en un modelo social puede llegar a ser de poca relevancia pero para la comparativa con un modelo con análisis de contenido fueron descartados en este entrenamiento también.

5.2.2. Heuristica de *relevancia* de un tweet

El objetivo de clasificación en dos clases *relevante* o *insignificante* para un tweet hace qué debamos determinar una cota en la cantidad de retweets para considerarlo relevante si

la superara o insignificante en caso contrario. La cota se obtuvo mediante un muestreo de los tweets agrupados en cantidad de retweets. Tomando el percentíl 90 %, sobre el dataset normalizado(Figura 4.3), cómo la cota de división de clases se iniciaron los experimentos donde la relevancia se determinará superando los 13 retweets.

5.2.3. Detección de comunidades

En este proyecto se utilizaron dos algoritmos de selección, *infomap* con una clusterización más dispar y *multilevel* donde la cardinalidad de los grupos resultó ser más equitativa.

Dichos agrupamientos fueron utilizados en dos experimentos modificando el proceso de selección de influenciadores.

En el primero con el objetivo de una selección más equitativa entre comunidades, ya no se tomaron los usuarios con mayor influencia desde el ranking general, sino una cantidad sobre cada comunidad proporcional a su cardinalidad, asegurando al menos un influenciador de cada agrupación.

Otro experimento qué resulta de interés es determinar el comportamiento del modelo sobre un subconjunto de usuarios, sectorizando de esta forma el problema general. Las distintas comunidades determinadas bajo estos algoritmos de detección, fueron evaluadas cómo subconjuntos de observación en este experimento.

5.2.4. Extracción de *features*

Las características o features pueden mejorar sustancialmente el rendimiento de un clasificador. Cómo en cualquier aprendizaje, la información qué se brinde qué detalle con mayor precisión de cada ejemplo puede colaborar con la discriminación de los casos, aunque en abundancia puede ser difícil identificar las similitudes. De esta manera en este modelo se decidió incluir cómo features a la aceptación de los influencers con respecto a un tweet es decir, si lo retweeteó o no. Así un tweet s quedará determinado por un vector booleano $I_{s1}, .., I_{sn}$ de dimensión equivalente a la cantidad n de influencers qué el modelo haya seleccionado, donde cada I_{jk} representará si el tweet j se encuentra en la linea de tiempo del influencer k con 1 en caso afirmativo, o 0 en caso contrario.

$$Ts := \begin{bmatrix} i_{s1} & i_{s2} & \dots & i_{sn} \end{bmatrix}$$

Quedando entonces los vectores de cada tweet comprendidos en una matriz $\in \mathbb{R}^{m \times n}$ a de características de la siguiente manera:

$$features := \begin{bmatrix} T_1 \\ \dots \\ T_s \\ \dots \\ T_m \end{bmatrix} = \begin{bmatrix} i_{11} & i_{12} & \dots & i_{1n} \\ \dots & \dots & \dots & \dots \\ i_{s1} & i_{s2} & \dots & i_{sn} \\ \dots & \dots & \dots & \dots \\ i_{m1} & i_{m2} & \dots & i_{mn} \end{bmatrix}$$

donde

$$i_{sj} = \begin{cases} 1 & s \in \texttt{timeline}(j) => \text{RT de j} \\ 0 & \text{caso contrario} => \text{no RT de j} \end{cases}$$

Asi obteniendo desde los datos de entrada la matriz de características veamos cómo sería el vector objetivo el cuál primero será el blanco de entrenamiento y luego será comparado con la predicción. Dado entonces T_s, con $s \in 1, .., n$ tendremos su clasificación relevante si supera la cota de RT's necesarios para serlo y se le asignará la clase 1 en el vector resultante. Dados $T_1, .., T_m$ tendremos entonces el siguiente vector objetivo.

$$clasificacion = \begin{bmatrix} r_1 \\ \dots \\ r_s \\ \dots \\ r_m \end{bmatrix}$$

donde

$$r_s = \begin{cases} 1 & \texttt{retweets}(s) > \text{cota de relevancia} \\ 0 & \text{caso contrario} \end{cases}$$

5.2.5. Clasificadores y sus parámetros

Dada la partición de datos de entrenamiento y evaluación en 75 % y 25 % sobre el corpus ya reducido, se evaluaron alternativas de clasificadores de `scikit-learn`. En una primer instancia se realizaron los experimentos con `RandomForestClassifier`[1] con una configuración de parámetros asistida por `GridSearchCV`[2] con objetivo en optimizar el puntaje F1, mediante la siguiente grilla:

```
{
  "n_estimators" : [9, 18, 27, 36, 45, 54, 63],
  "max_depth" : [1, 5, 10, 15, 20, 25, 30],
  "min_samples_leaf" : [1, 2, 4, 6, 8, 10],
  "class_weight": ['balanced', None]
}
```

Obteniendo cómo la mejor configuración los siguientes parámetros.

```
{
  'n_estimators': 63,
  'min_samples_leaf': 2,
  'max_depth': 10,
  'class_weight': 'balanced'
}
```

[1] Clasificador de tipo Random Forest de sklearn.

[2] Libreria de sklearn para la búsqueda de hiperparametros tipo gridsearch.

Con un puntaje F1 de 41 % sobre la clase relevante con un 10 % de influenciadores sobre el total de usuarios no se presentaron resultados satisfactorios.

Por esta razón cambiando el clasificador por uno de tipo `SVC`[3] se repitió el experimento una vez más utilizando `GridSearchCV` con la siguiente grilla:

```
{
  'kernel': ['rbf', 'poly', 'linear'],
  'gamma': [0.1, 1, 10, 100],
  'C': [0.01, 0.1, 1, 10, 100],
  'class_weight': ['balanced', None]
}
```

Obteniendo cómo mejor configuración:

```
{
  'kernel': 'rbf',
  'gamma': 0.1,
  'C': 1,
  'class_weight': None
}
```

Con una performance superior al RF, se obtuvo un score de 77 % con los mismos influencers. Las SVM parecen ser la mejor opción para nuestro problema de clasificación binaria.

5.3. Predicción con análisis de contenido

La información social aportada por las relaciones de los usuarios dieron resultados interesantes, ¿Que pasará si añadimos información del contenido?. En esta sección veremos cómo aplicando una detección de temática podemos incluir mayor información en los ejemplos de entrenamiento pretendiendo aprender detalles de situaciones particulares, tales cómo un tweet qué publique un usuario *popular* pero sobre un tema con baja aceptación. Se pretende mejorar el puntaje F1 superando la alternativa puramente social. Para ello se utilizara un preproceso de los textos de cada tweet y se aplicará alguna variante de LDA para distinguir temáticas, explicados a continuación en detalle.

5.3.1. Modificación de *features*

En este modelo los features no serán sólo provenientes de la actividad de los influencers, sino también de la temática qué desarrollan los tweets. Cada vector T'_s representante del

[3] *Support Vector Classifier* Nombre del clasificador de tipo SVM de sklearn

tweet s será idéntico a los T_s sociales añadiendo k columnas más, donde k representa la cantidad de tópicos discriminados entre el corpus. Y cada T'_s en su columna j con $j > n$ donde n representa la cantidad de influencers, será si el tópico j *tiene participación*[4] en el tweet s.

$$T'_s := \begin{bmatrix} i_{s1} & i_{s2} & \dots & i_{sn} & t_{11} & \dots & t_{1k} \end{bmatrix}$$

De esta manera se modifica la matriz de features quedando de la siguiente manera:

$$features := \begin{bmatrix} T'_1 \\ \dots \\ T'_s \\ \dots \\ T'_m \end{bmatrix} = \begin{bmatrix} i_{11} & i_{12} & \dots & i_{1n} & t_{11} & \dots & t_{1k} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ i_{s1} & i_{s2} & \dots & i_{sn} & t_{s1} & \dots & t_{sk} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ i_{m1} & i_{m2} & \dots & i_{mn} & t_{m1} & \dots & t_{mk} \end{bmatrix}$$

donde

$$i_{sj} = \begin{cases} 1 & s \in \texttt{timeline}(j) => \text{RT de j} \\ 0 & \text{caso contrario} => \text{no RT de j} \end{cases}$$

$$t_{sj} = \begin{cases} 1 & j \in \texttt{topics}(s) => \text{la temática j esta presente en s} \\ 0 & \text{caso contrario} => \text{la temática j no esta presente en s} \end{cases}$$

En cuanto a la matriz de clasificación sigue manteniendo su dimensión y su conformación sigue siendo de la misma forma, dado qué no interviene en el proceso de caracterización.

5.3.2. Preprocesamiento adoptado

Aplicar técnicas de detección de tópicos sin un preprocesamiento de los datos sobre todo en textos provenientes de redes sociales puede llegar a dar resultados no muy alentadores. Por esta razón decidimos aplicar un preproceso a cada texto de los tweets objetivo tanto para el entrenamiento cómo para la evaluación. Estas son las etapas:

- **Remoción de palabras auxiliares, símbolos y números**. Considerando estás pseudopalabras cómo carentes de información debido a su uso cómo auxiliares, conectores, expresión, o en el caso de los números para contabilización, se decidió retirarlas en su totalidad.
- **Remoción de tíldes**. El uso de tíldes sobre todo en dispositivos móviles no está tan resuelto debido a la variedad de teclados y configuraciones. Cómo nuestro foco es una red social en la cuál muchas publicaciones son provenientes de este medio, podemos encontrar palabras con el mismo significado a veces escritas con los tildes correctos y muchas otras no. Por ello decidimos tomar las palabras sin tildes asumiendo los errores qué esto podría conllevar cómo mínimos, relacionando así las apariciones de una misma palabra ya sea con el tilde o no.

[4] Tomaremos en el caso de LDA típico cómo participación si el texto contiene más del 25 % con respecto a la temática.

- **Lematización**. La familia de palabras con una misma raíz sólo suele ser distinguida en cuestiones de temporalidad o conjugación del lenguaje. Sin embargo la temática qué desarrollan puede ser en muchos caso idéntica. Tanto la opción de Stemming cómo Lematización pretenden resolver estas disparidades considerando cada palabra cómo su raíz o palabra en infinitivo respectivamente. Dado qué Stemming es una técnica qué fue pensada en un idioma con una estructura similar al inglés, se aplicó a la técnica de Lematización con mejores resultados en el idioma Español, seleccionado en nuestro proyecto.

- **Pasaje a minúsculas** Desde ya qué sin importar si una palabra esta en principio de una oración o escrita en *Camelcase* por ejemplo, tendrá el mismo significado. Sin embargo si no hacemos la transformación a minúsculas de la palabra la distinción será inminente.

5.3.3. Modelo con LDA

LDA, cómo detallamos en la sección 2 es una técnica utilizada para la detección de tópicos en documentos y de esta manera poder asociar aquellos qué sean similares en cuanto a sus temáticas relacionadas.

En nuestro caso fue utilizada para esta tarea analizando cómo documentos los tweets de los usuarios. Mediante esta clasificación de tópicos, fue modificada la matriz de *features* cómo se explicó en la sección anterior donde se incorporó de esta manera información referente al contenido.

Dado qué LDA no identifica la participación de un sólo tópico en el documento sino la proporción de cada uno sobre el mismo, combinada con la escasa longitud de los documentos, hizo qué los resultados no sean los esperados. Los tweets tenían demasiados tópicos asignados aunque muchos con poca participación. Para combatir este fenómeno se determinó qué los tópicos con una participación menor a un 25 % no serían tenidos en cuenta cómo temática involucrada en el texto y fueran descartados.

Sin embargo esto no parecería ser suficiente ya qué podríamos tener todavía más de un tópico por tweet, escenario dificil, ya qué en un texto de 140 caracteres cómo máximo[5], es poco frecuente hablar de más de un tópico. Agregando a la matriz presentada en la sección 5.3.1 la siguiente condición sobre los T'_s

$$\forall s \in 1..m : (\sum_{y=1}^{k} t_{sy} >= 1)$$

no parecería ser suficiente. Por este motivo decidimos probar una mutación de esta idea, qué sólo asigne una temática por cada documento corto denominada Twitter-LDA.

[5]Límite aumentado, a fines de 2017, permitiendo publicaciones de hasta 280 caracteres.

5.3.4. Adaptación con Twitter-LDA

Compartiendo la idea general de tópicos en documentos, esta adaptación del modelo natural de LDA hacia documentos cortos, se diferencia en la asignación de sólo un tópico a cada documento, asumiendo de esta forma qué no es posible hablar de 2 temas distintos en una longitud de 140 caracteres.

Tenemos entonces una modificación de nuestra matriz presentada en la sección 5.3.1 donde ahora se cumplirá en todo T'_s

$$\forall s \in 1..m : (\sum_{y=1}^{k} t_{sy} = 1 \wedge \exists y \in 1..k : t_{sy} = 1)$$

5.4. Resultados

Concluida la descripción de los procedimientos realizados, veremos algunos de los experimentos y sus resultados obtenidos mediante las diferentes configuraciones.

Experimento puramente social

Con el objetivo de una primera evaluación del modelo más simple se buscó obtener resultados alentadores obviando los detalles. De esta forma dos métricas de centralidad fueron evaluadas frente al modelo con selección aleatoria[6]. En la figuras 5.3 y 5.4 se puede ver qué la selección de influencers mediante el algoritmo de Katz resulta en un puntaje f1 menor en comparación con la alternativa de PageRank, aunque en ambos casos los resultados excedieron ampliamente los valores esperados y fueron satisfactorios para nuestra hipótesis del experimento.

[6]Explicada en detalle en la sección 5.1.1, la métrica F1-score del modelo es tomada cómo el promedio de resultados de múltiples repeticiones del experimento con la misma cantidad de influencers pero con nuevas selecciones.

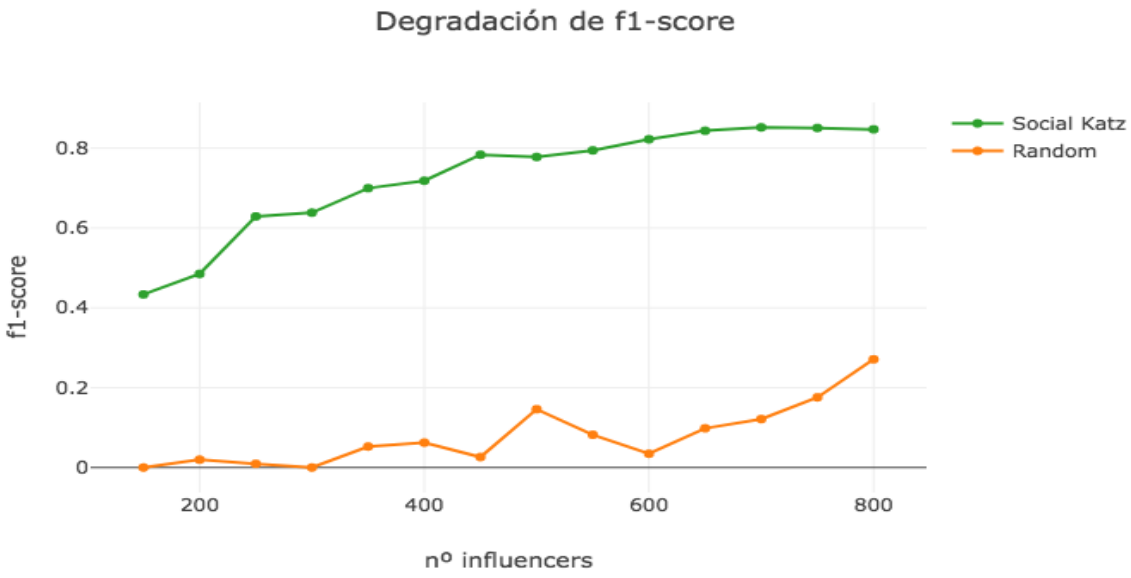


Figura 5.3: Puntaje f1 del primer modelo social. Selección de influencers por centralidad de **Katz** y actividad en contraste con una selección aleatoria.

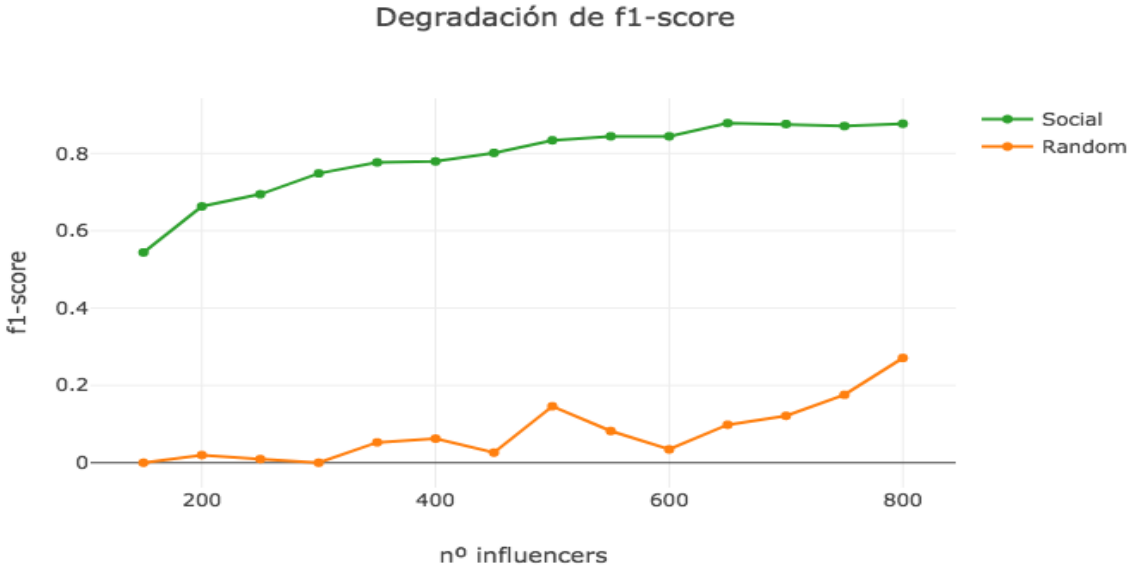


Figura 5.4: Puntaje f1 del primer modelo social. Selección de influencers por centralidad de **PageRank** y actividad en contraste con una selección aleatoria.

Adicionando LDA

Como mencionamos anteriormente el modelo LDA involucra la obtención de tópicos sobre el corpus de tweets de entrenamiento. Buscamos en este experimento aumentar nuestras métricas obtenidas con el experimento anterior de menor complejidad. En la Figura 5.5 podemos ver mediante la visualización de pyLDAVis cómo se realizó el análisis de los tópicos resultantes.

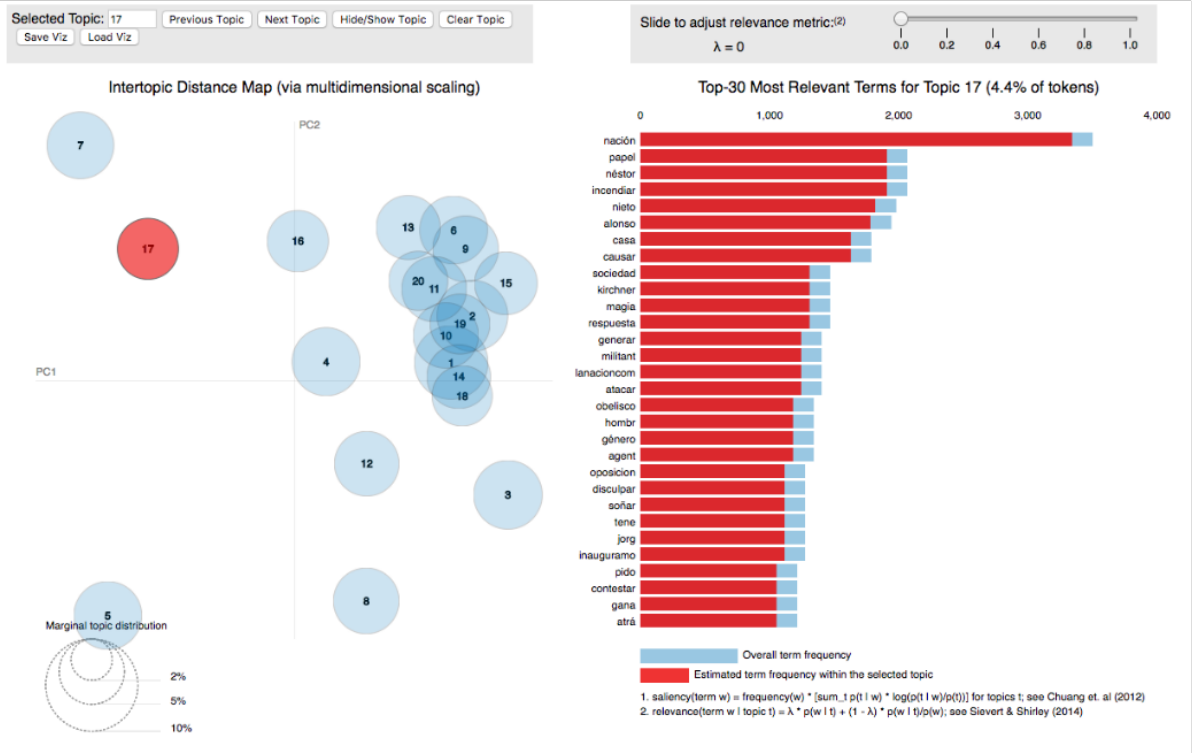


Figura 5.5: Visualización pyLDAVis de los tópicos generados por el modelo lda_20

Por su parte la Figura 5.6 muestra los resultados sobre el experimento, incorporando la información temática del contenido cómo *features* para el clasificador. Se obtuvieron así los resultados de la métrica f1 oscilantes a la curva puramente social sin una mejora interesante del modelo independientemente de la cantidad de tópicos qué se hayan seleccionado. Por este motivo se descartó seguir con el análisis de este modelo con LDA.

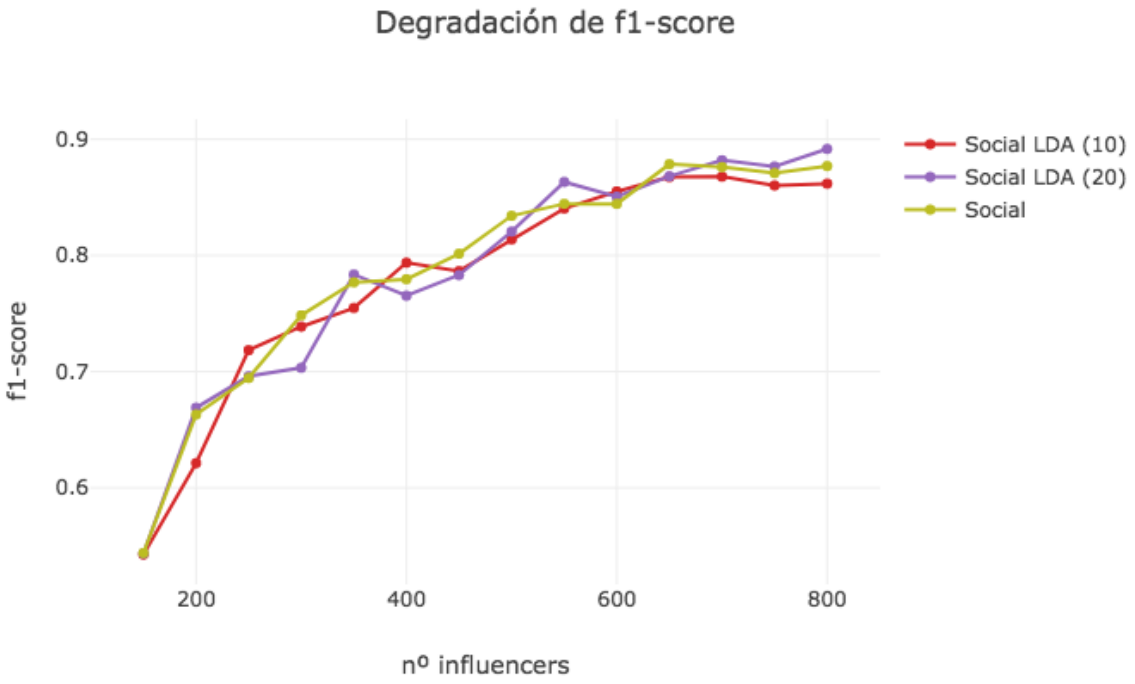


Figura 5.6: Puntaje f1 de modelo con análisis de contenido sin mejora aparente.

Ajuste de LDA a Twitter-LDA

Dados los insatisfactorios resultados del experimento previo, se analizó la modificación del método para asignar temáticas. Sustituyendo LDA por una adaptación especialmente ajustada para textos cortos. Con el mismo objetivo pretendiendo qué la información del contenido adicione conocimiento en el entrenamiento buscamos mejorar los resultados del modelo puramente social. En este caso, cómo se muestra en la Figura 5.7, una vez más la incorporación de éstas características no dió los resultados esperados sobre todo el espectro de los influencers pero es notable destacar la diferencia ganada en los casos donde la cardinalidad de influencers es baja. Es posible qué el escaso número de características de contenido frente a la abundancia de características sociales pueda anular su aporte a partir de los 200 influencers. De igual forma parece un resultado interesante a tener en cuenta en los casos de *Cold Start* o pocos datos.

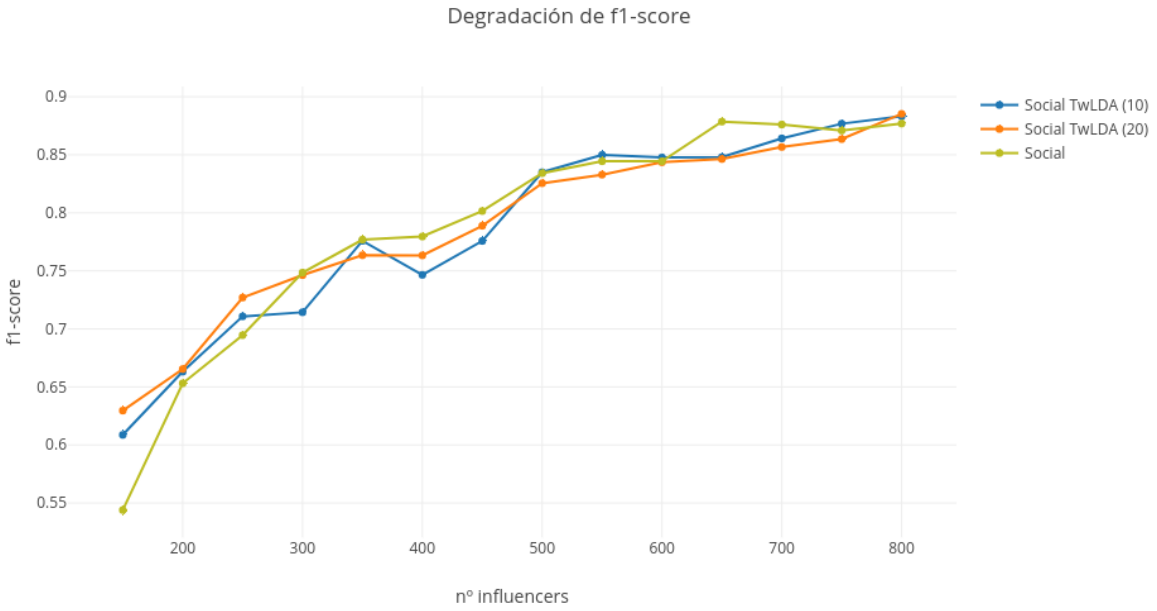


Figura 5.7: Puntaje f1 del modelo evaluado con contenido mediante la adaptación de Twitter-LDA

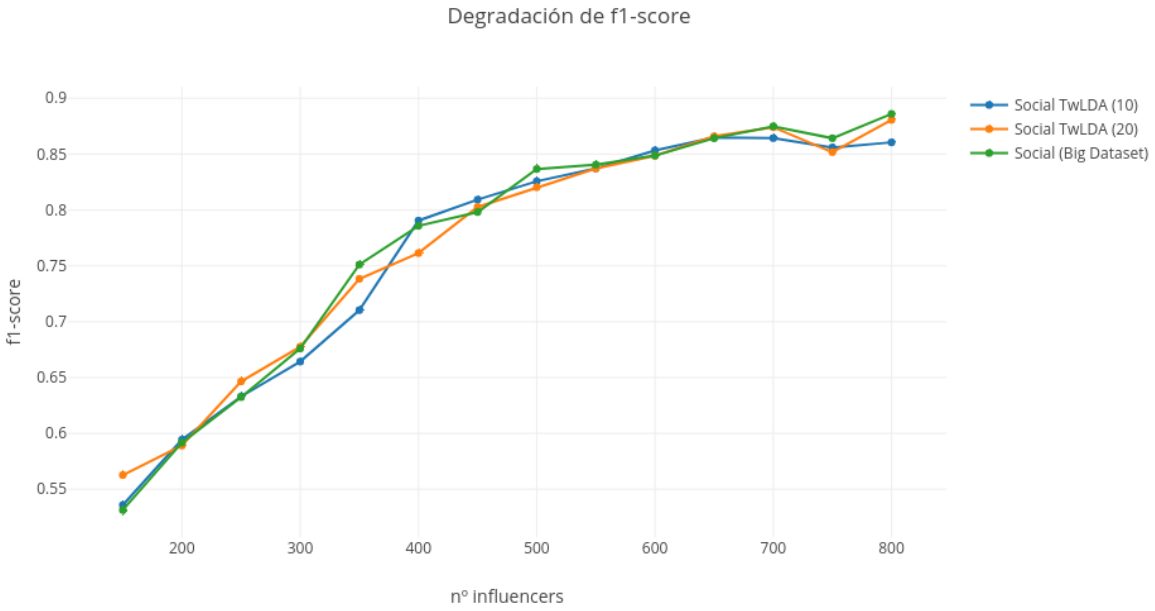


Figura 5.8: Puntaje f1 del modelo evaluado con contenido mediante la adaptación de Twitter-LDA sobre el dataset completo

Experimento puramente basado en contenido

La incógnita generada por los anteriores 2 experimentos generó la necesidad de analizar cuanta información aporta el contenido a la hora de clasificar. En la Figura 5.9 podemos ver qué realmente la información del tópico del contenido parece ser muy poco informativa e incluso confusa para el clasificador. Sin embargo analizando la particularidad de nuestro problema, es esperable qué poco se pueda predecir del impacto del contenido sin evaluar el alcance. Una publicación de un tema importante por un usuario aislado o de un tema irrelevante por un usuario central son casos ejemplificadores qué validan esta idea donde el alcance puede afectar tanto cómo el contenido. Por otro lado los comentarios generalistas o poco informativos de esta red social, junto con la corta longitud del mensaje parecen ser razones suficientes para no conseguir un buen agrupamiento al menos con algoritmos del tipo LDA.

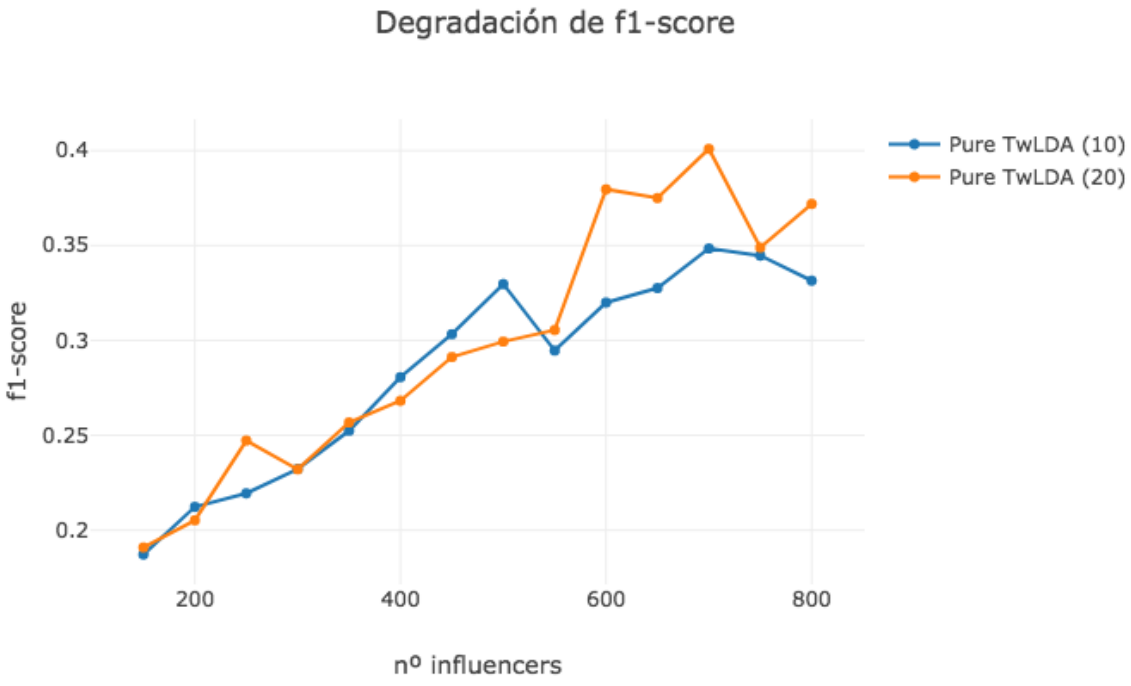


Figura 5.9: Modelo sólo basado en *features* de contenido

Influencers distribuidos por comunidades

La suposición de sobreajuste sobre un grupo particular de usuarios en la detección de los influenciadores, dió lugar a un nuevo experimento, en el cuál dichos influenciadores sean seleccionados de una manera más justa entre las comunidades dentro del dataset. En la Figura 5.10 se pueden apreciar los resultados de este experimento, tomando una cantidad de influenciadores sobre cada comunidad relativa a su tamaño. Este experimento mostró una performance siempre por debajo del experimento social puro, y aunque no fue satisfactorio su resultado, nos permitió asegurar qué la selección de influenciadores tomando el dataset cómo una comunidad es la mejor opción.

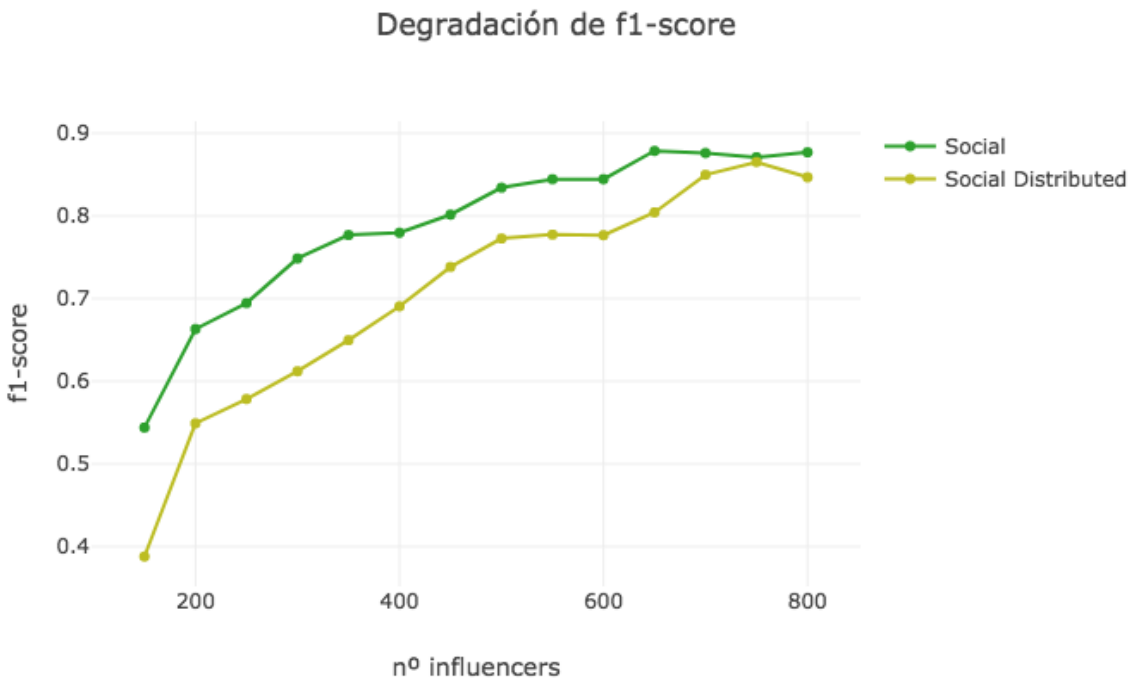


Figura 5.10: Puntaje f1 con una distribución de influenciadores relativa a la proporción de la comunidad.

Experimento aislando subconjunto de usuarios de una comunidad

Como ultima alternativa se buscó un análisis más sectorizado evaluando las comunidades aisladas formadas con el algoritmo de `infomap` y `multilevel`. En ambos experimentos los resultados sólo pudieron ser evaluados sobre aquellas comunidades qué contaran con un volumen admisible de actividad y sólo en una comunidad con 367 individuos formada mediante el algoritmo de `infomap`, los resultados fueron similares a los del experimento general reflejados en la Figura 5.11. A pesar de los resultados poco satisfactorios de este experimento cómo el anterior mencionado, creemos qué el motivo de este problema podría deberse a qué el dataset en sí puede ser considerado cómo una comunidad, intentar minimizar aún más su cardinalidad es evidentemente una mala decisión ya qué se reducen en gran proporción la cantidad de ejemplos de entrenamiento dejando una enseñanza escasa para una buena evaluación.

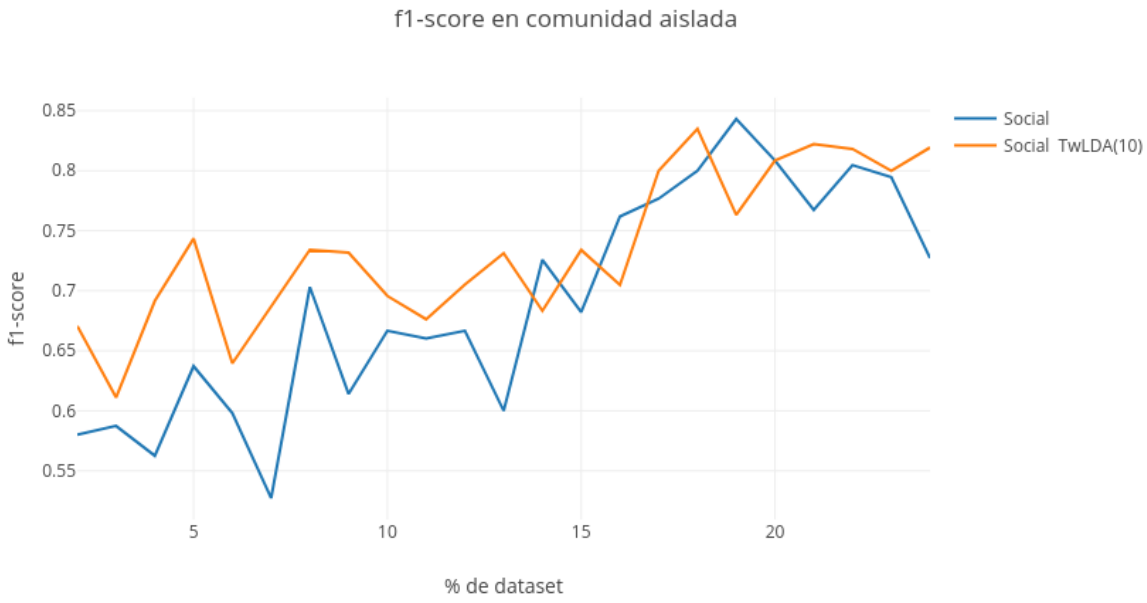


Figura 5.11: Experimento social aislando los datos de la comunidad más grande del dataset.

Capítulo 6

Conclusiones y Trabajo futuro

6.1. Conclusiones

Como conclusión general del trabajo y con respecto a lo planteado en el inicio del mismo comprobamos qué la información social qué voluntariamente aportan los usuarios de las redes populares puede ser de importancia para hacer una recomendación o incluso anticipar eventos del futuro.

El análisis de los resultados muestra qué los modelos sociales gozan de una alta calidad, pero sorprendentemente la temática del contenido de una publicación parece ser menos relevante qué su origen. Aquel contenido publicado por los referentes será fácilmente popular en la mayoría de los casos, mientras qué el mismo contenido por los usuarios relegados, aislados de la gran masa será insignificante.

La experimentación en comunidades se vió reducida dado el tamaño del dataset insuficiente para aplicar a estas técnicas. De igual forma los resultados arrojados sobre estos experimentos invitan a profundizar en una futura investigación ya qué se presentaron en algunos casos similares a los valores más altos de esta entrega. Creemos con convicción qué mediante un análisis qué permita descubrir y corregir los detalles de este modelo, podría superarse la performance actual y resolver las problemáticas de pequeños conjuntos de datos.

6.2. Trabajo futuro

6.2.1. Modelo sobre dataset en vivo.

Como una tarea bien técnica, adaptar los modelos a un entrenamiento dinámico y predicción sobre la actualidad relativa puede dar lugar a una herramienta interesante. La predicción se debería dar sin qué exista el resultado de comparación y en una segunda iteración, tiempo después determinar si fue correcta o no para medir la calidad del modelo.

6.2.2. Temporalidad en los tweets

El modelo actual de evaluación selecciona tweets de entrenamiento o evaluación independientes de temporalidad. Es decir podemos predecir sobre la actualidad cómo así también sobre el pasado teniendo un entrenamiento potencialmente mixto. Ajustar el entrenamiento sólo referente al pasado nos dará información más realista si podemos aprender del histórico.

6.2.3. Adicionar *features* al modelo.

Son infinitas las características qué se pueden adicionar en los tweets en el modelo. Entre las qué quedaron pendientes adicionar en nuestro caso se encuentran

- Afinidad del autor con cada uno de los influenciadores: incorporar más información de la relación del autor del tweet con el influencer evaluado. Pueden ser medidas de distancia en el grafo, pertenencia a la misma comunidad o demás medidas qué los relacionen.
- Sentimiento del tweet: Analizar si el tweet es violento de protesta o pretende contagiar alegría puede ser fundamental a la hora de influenciar un retweet.
- Entidades y *hashtag*: La posibilidad de incorporar mediante @ o # a otros usuarios o temáticas en común puede aumentar el alcance o impacto de una publicación y podría ser también determinante.

No descartamos tampoco la posibilidad de sustituir las características generadas por el proceso LDA por la información de contenido directamente. Es decir en reemplazo de la categoría o tópico qué resulta de una evaluación mediante un modelo LDA, incluir las características de las palabras del contenido o utilizando un *embedding* tipo Doc2Vec.

6.2.4. Profundizar el análisis por comunidad.

La detección de comunidades sobre grafos sigue siendo una tarea de vanguardia. No resuelta completamente, sobre todo en grafos pequeños. En esta entrega se inicio la exploración de algunos algoritmos aunque con resultados mucho más bajos de lo esperado. Continuar y profundizar sobre la tarea puede develar algunas características interesantes de los agrupamientos.

6.2.5. Ampliar comunidades

Ampliar las comunidades con un nivel más de seguidores de los integrantes y usar la ampliación también cómo características de nuestro modelo puede generar información de valor, al menos en comunidades tan pequeñas cómo las resultantes en nuestro caso.

Por otro lado esta tarea puede darnos información de cohesión qué caracterice cada comunidad. Caracterizar las comunidades puede ser una tarea reveladora para determinar las condiciones bajo las cuáles los modelos tienen una mejor performance.

6.2.6. Clustering de tweets.

A pesar de qué esta tarea fue descartada en principio y reemplazada por LDA, teniendo en cuenta qué en definitiva adaptamos el algoritmo a la clasificación de una sola categoría mediante Twitter-LDA podría ser reconsiderada. Una vez más la propuesta de transformar un Tweet a vector es mediante *embeddings* tipo Doc2Vec. Teniendo agrupados los tweets por características similares incluir esta información cómo *features* de modelo.

Bibliografía

[1] Celayes Pablo, Dominguez Martin Recomendación de Información basada en Análisis de Redes Sociales y Procesamiento de Lenguaje Natural

[2] Freeman, Linton A set of measures of centrality based on betweenness

[3] Brin, Sergey and Page, Lawrence The Anatomy of a Large-Scale Hypertextual Web Search Engine

[4] Katz, L. (1953). A New Status Index Derived from Sociometric Analysis. Psychometrika, 39–43.

[5] Rosvall, M. and Bergstrom, C. T. An information-theoretic framework for resolving community structure in complex networks. Proceedings of the National Academy of Sciences 104, 7327–7331 (2007).

[6] Peel, L. Estimating network parameters for selecting community detection algorithms. In 13th Conference on Information Fusion 1–8 (IEEE, 2010).

[7] Mukherjee, A., Choudhury, M., Peruani, F., Ganguly, N., Mitra, B. Dynamics On and Of Complex Networks, Volume 2: Applications to Time-Varying Dynamical Systems (Springer Science and Business Media, 2013)

[8] Blondel, V. D., Guillaume, J.-L., Lambiotte, R. and Lefebvre, E. Fast unfolding of communities in large networks. Journal of Statistical Mechanics: Theory and Experiment 2008, P10008 (2008).

[9] Xie, J., Szymanski, B. K. Community detection using a neighborhood strength driven label propagation algorithm. In Network Science Workshop 188–195 (IEEE, 2011).

[10] Pons, P., Latapy M.: Computing communities in large networks using random walks. In Computer and Information Sciences-ISCIS 2005, 284–293 (Springer, 2005).

[11] Xie, J., Szymanski, B. K.: Community detection using a neighborhood strength driven label propagation algorithm. In Network Science Workshop 188–195 (IEEE, 2011).

[12] Rousseeuw, P.J.; Kaufman, L. (1990). Finding Groups in Data: An Introduction to Clúster Analysis. Wiley.

[13] David M. Blei, Andrew Y. Ng, Michael I. Jordan Latent Dirichlet Allocation

[14] `https://es.wikipedia.org/wiki/Distribuci%C3%B3n_de_Dirichlet`.

[15] Breiman, Leo (2001) Random Forests

[16] Cortes, Corinna, Vapnik, Vladimir N. (1995). Support-vector networks

[17] Shawe-Taylor, J., Cristianini, N. (2004). Kernel Methods for Pattern Analysis.

[18] https://en.wikipedia.org/wiki/Cross-validation_(statistics)

[19] Pablo Celayes, Martin Dominguez: Prediction of User Retweets Based on Social Neighborhood Information and Topic Modelling

[20] Flaviano Morone: Collective Influence Algorithm to find influencers via optimal percolation in massively large social media". *Nature, 2016*

[21] ZhaoYang: A Comparative Analysis of Community Detection Algorithms on Artificial Networks. Prediction of User Retweets Based on Social Neighborhood Information and Topic Modelling

[22] Wayne Xin Zhao, Jing Jiang et al. Comparing Twitter and traditional media using topic models.

[23] Tweepy http://tweepy.readthedocs.org/en/v3.2.0/.

[24] SQLAlchemy http://docs.sqlalchemy.org/en/latest/.

[25] Jupyter https://jupyter.readthedocs.io/en/latest/index.html.

[26] NetworkX http://networkx.readthedocs.io/en/stable/.

[27] Python-igraph http://igraph.org/python/

[28] Scikit-learn http://scikit-learn.org/stable/documentation.html.

[29] NLTK http://www.nltk.org/.

[30] gensim https://radimrehurek.com/gensim/tutorial.html.

[31] Spacy https://spacy.io/

[32] Gephi https://gephi.org/users/.

[33] pyLDAvis http://pyldavis.readthedocs.io/en/latest/

[34] PlotLy https://plot.ly/python/user-guide/

Los abajo firmantes, miembros del Tribunal de Evaluación de tesis, damos Fe que el presente ejemplar impreso, se corresponde con el aprobado por éste Tribunal