Universidad de Buenos Aires
Facultad de Ingeniería

Propuesta de Trabajo Profesional

Análisis de sentimientos más allá de la polaridad sobre el stream de Twitter

Martinez, Gaston Alberto – 91383

Índice

1. Introducción

El siguiente documento presenta la propuesta de Trabajo Profesional de Ingeniería en Informática del estudiante Gaston Alberto Martinez (padrón 91383).

El objetivo es aplicar los conocimientos adquiridos durante la carrera, para lo cual el tema elegido es **"Análisis de sentimientos más allá de la polaridad sobre el *stream* de Twitter"**.

El análisis de sentimientos es un proceso que se utiliza para determinar si un texto contiene una opinión del escritor. Usualmente, se traduce en una clasificación positiva o negativa del mismo y suele emplearse en reseñas o redes sociales. El análisis aplicado a estas últimas resulta de gran interés, a tal punto de ser motivo de competencias.[1] En la práctica, estos análisis abarcan desde el conocimiento de la opinión de un conjunto de personas sobre un tópico en particular hasta determinar problemas con algún servicio de forma completamente automática. El creciente volumen de datos y aplicaciones posibles sirven de motivación para el presente trabajo práctico.

Por otra parte, la realización de un aporte de software libre responde a la intención de contribuir con la comunidad que a diario provee de herramientas de uso académico y profesional.

2. Descripción del problema

Twitter es uno de los servicios de *microblogging*[2] más conocidos y utilizados en la actualidad. En él los usuarios plasman situaciones de sus vidas diarias, comunican emociones e ideas y transmiten espontáneamente cualquier pensamiento que les surja en el momento.

> Increíble Egipto, impresionante, locura, se cae el estadio, están en Rusia 2018 con gol al minuto 95, se los habían empatado. Te amo futbol.
>
> -**@RiverettiMX** (*8 oct. 2017*)

De esta manera, Twitter resulta una fuente de conocimiento acerca del estado emocional de las personas, especialmente en lo que respecta a los

[1] http://alt.qcri.org/semeval2017/task4/

[2] Sitio web que incluye, a modo de diario personal de su autor o autores, contenidos de su interés, actualizados con frecuencia y a menudo comentados por los lectores[1]. Los sistemas de *microblogging* se caracterizan por limitar el número de caracteres de cada entrada, usualmente a no más de 140.

acontecimientos de carácter masivo. Es sencillo inferir la opinión de un conjunto de la población sobre un tópico en particular (o la manera en la que los afecta) solo leyendo los tweets relacionados con el mismo.

No obstante, Twitter ya no se utiliza únicamente como red social de *blogging*. Se le da uso, también, como herramienta de publicidad, de noticias o, simplemente, de compartición de fotos.

> Mendoza: La reacción de un grupo de vecinos con un auto estacionado sobre la senda peatonal https://goo.gl/GuEDrg
>
> -**@C5N** (*8 oct. 2017*)

Este tipo de tweets son los que se denominan tweets neutros, los cuales no permiten inferir información sobre el emisor o, al menos, no sólamente en base al texto.

El trabajo constará, en primera instancia, en separar los tweets neutros de los restantes. Luego, en clasificar los no filtrados de forma automática entre las diferentes clases definidas. Además, como este tipo de análisis tiene un valor mayor si se lo realiza en tiempo real, es necesario transformarlo en un requerimiento necesario para darle un valor agregado al trabajo.dotarlo de un beneficio

Las formas más comunes de lograr una buena performance en el procesamiento de datos infinitos suelen ser a través de la utilización de arquitecturas del tipo pipeline, como las que se pueden encontrar en los microprocesadores.

Las arquitecturas pipeline se caracterizan por separar el proceso en etapas que funcionan de forma independiente y que sólo dependen del resultado de la etapa anterior. Este tipo de construcciones es naturalmente paralelizable y permite procesar una mayor cantidad de datos en simultáneo, impidiendo que las partes más lentas del proceso ralenticen a todo el procesamiento global. Las mismas ya se han utilizado con éxito en el campo del procesamiento del lenguaje natural como, por ejemplo, la denominada TANL Pipeline de Giuseppe Atari [2].

Dichas arquitecturas, al ser partes independientes que únicamente consumen datos y producen nuevos, son fácilmente distribuibles en distintos equipos. Su capacidad de escalar horizontalmente, utilizando los mecanismos indicados, permite mejorar significativamente el volumen de datos procesados en simultáneo.

3. Objetivos

El Trabajo Profesional consta de 3 objetivos principales:

- Desarrollar un sistema capaz de clasificar textos obtenidos a través de un stream infinito, de forma eficiente.
- Diseñar y analizar una alternativa viable a los clasificadores polares.
- Mostrar los resultados de forma intuitiva y en tiempo real.

4. Características del trabajo

4.1. Análisis del problema

El punto de partida es la propuesta de tesis de Phillip Smith[3], en la cual se hace un análisis de sentimientos multivaluado. El enfoque propuesto por la misma es utilizar clustering para la clasificación y difiere en nuestro objetivo en varios puntos clave:

- Es un solo lenguaje el que se analiza.
- Es un clasificador estático, lo cual implica que no importa cuándo estén los resultados sino que estén.
- Asume buena redacción/sintaxis, de lo cual no existe garantía en Twitter.
- Se ciñe únicamente a notas de suicidio (tópico uniforme).

En nuestro caso, el ambiente multilenguaje de por sí supone un nivel de datos muy grande como para poder manejarlos efectivamente mediante clustering; sin contar el hecho de que el volumen de features es exponencialmente grande y que vuelve la tarea mucho más engorrosa. Por estas razones se descarta ese enfoque a priori, pero se continúa considerando el análisis hecho en el paper ya que provee un esquema útil para la conformación de las clases de sentimientos.

Existen al menos dos cuestiones claves: el qué y el cómo. No siempre resulta tan importante lo que se dice sino también cómo se lo dice. Esto último ayuda a distinguir ciertas subjetividades en el discurso. Por ejemplo, se puede enfatizar un estado de emoción violenta a través de las mayúsculas mientras que, la misma frase en minúsculas, atenúa la intensidad de ese sentimiento puntual para el lector. Además de las dos cuestiones mencionadas,

existen otros mecanismos que le dan connotaciones subjetivas a la misma construcción sintáctica. El con qué es parte de ello: el uso de emojis puede ser indicador de ironía en una oración ó restarle importancia a su significado literal. "Hoy me robaron" y "hoy me robaron =P" son drásticamente distintas. Este tipo de construcciones son dependientes del idioma, del entorno sociocultural del que escribe y del público al que quiere dirigirse. En un ambiente multilenguaje resulta imposible realizar a mano dicha clasificación ya que, en un entorno donde el espectro de emociones es más amplio y específico, apuntar simplemente a la sintaxis es un abordaje insuficiente, aunque, por otra parte, sirve para asociar fácilmente un discurso a una escala de solo positivo-negativo. Entonces, cuando se intenta abarcar un mayor espectro de matices, resulta más difícil lograr un buen resultado sin agregar otros indicadores. Un claro ejemplo se da en el empleo de una lengua extranjera: un estadounidense diciendo "taco taco taco" posee una connotación distinta a la que podría guardar si lo hiciera un mexicano (es comúnmente utilizado como burla cuando alguien habla en español en un ambiente donde predomina el inglés).

De esta manera, se puede concluir que un enfoque clásico basado en lexicones no resulta un buen approach: por un lado, porque realizarlo a mano es físicamente imposible y, por el otro, porque si se emplearan técnicas de construcción automática de lexicones se dejaría de lado todo el significado implícito de cada palabra. En línea con lo anterior, un enfoque orientado a información contextual, como los basados en word2vec y en redes neuronales, se ve mucho más razonable.

Sin tener en cuenta el word2vec, los enfoques de redes neuronales tienen un cuello de botella muy importante: la entrada de datos. Si uno no puede garantizar que un feature map sea acotado, no se puede crear y entrenar una red con ello. Por lo tanto, si se acepta todo como un feature, los feature maps son infinitos. Más aún, con el tiempo aparecen dinámicamente nuevas features a medida que aumenta el número de muestras disponibles. Claramente se hace imposible elaborar un sistema escalable, acorde a las necesidades, usando este enfoque. A primera vista el hash trick sería una solución, pero la realidad es que tiene al menos un problema, y no menor: a medida que se agranda el universo de elementos que se pueden analizar (es un stream) llega indefectiblemente el momento en el que surge alguno que no puede ser clasificado. Utilizando el hash trick, al perder información de qué feature es en el featuremap, se pierde la capacidad de conocer eso de antemano y, más aún, puede matchear con algo que no corresponde. Descartando el hash trick se termina en una situación en la cual no se puede avanzar, dado el infinito número de features. Entonces, alternativas a eso utilizarían clustering para matchear cada feature a un índice distinto de la entrada de la red, indicando

la presencia de un grupo de features o no. Si bien esto parece una buena opción, el clustering clásico es lento y el agrupar según frecuencias de aparición en cada una de las clases supone un bias importante en la entrada de la red. Por lo tanto, word2vec surge como una opción más útil para reducir las dimensiones a la entrada de la red. Aún así, no soluciona a priori el problema de las features infinitas, es decir, lo acota pero no lo resuelve por sí mismo.

Ahora bien, con una pequeña modificación en la forma de trabajo se puede utilizar este mecanismo efectivamente. Word2vec cumple una doble función: por un lado, da una representación acotada de cada feature, con lo que el tamaño de la entrada se puede fijar en una de sus dos dimensiones (ancho). Por el otro, codifica dentro de esa representación la información contextual del feature. Estas dos características son ampliamente buscadas para el enfoque a adoptar. Aún así, por si solo sigue sin ser la solución alternativa al problema de las features infinitas. Recordando, se busca usar features que vayan más allá del simple significado de la palabra, pero aún se continúa sin poder acotar el número total de features a la entrada. Más todavía, añade un nuevo problema: si se planteara utilizar el mismo word2vec para codificar palabras y otros elementos, por un lado el algoritmo tardaría demasiado y, por el otro, encontraría relaciones inútiles, que podrían afectar el resultado final. Por ejemplo, siempre que se encuentre HOLA, se va a encontrar hola (suponiendo 2 tokenizers, uno sin preprocesamiento y otro que traslada todo a minúsculas). Ahora, es claro que hola todo en mayúsculas no se encuentra necesariamente relacionado con el significado de hola.

Asimismo, se puede pensar lo siguiente: en un tweet no puede haber más de 70 palabras. Las palabras en sí codifican el sentido semántico del tweet; mientras que las palabras mas el idioma de origen del tweet, las palabras mas si son hashtags y demases, codifican la información meta sintáctica del tweet. La cantidad de cada tipo de feature está acotada completamente debido a la existencia de una longitud máxima en los tweets. Entonces, se puede separar el embeding de cada grupo de features según su rol, con lo que, de esta forma, se achican los espacios y las features guardan real relación entre sí. Así también, se puede determinar la cantidad máxima de features de cada tipo que pueden aparecer, con lo que finalmente se simplifica el feature map.

Es más, incluso se pueden agregar otras features de soporte, así como embeddings a nivel carácter. A pesar de que se demostró que por sí mismos los embeddings a nivel carácter no producían buenos resultados, se los puede mantener como información de respaldo. En un contexto donde la sintaxis está lejos de ser perfecta, se puede utilizar cierta información de los caracteres utilizados para aproximar a la clasificación esperada.

A través de esta metodología, se puede utilizar un conjunto arbitrario de tokenizers y tener siempre un feature map de tamaño fijo y un embeding

representativo de cada feature generada. Para poder explotar word2vec de esta forma, es necesario poder representar cada feature como una cadena de forma inequívoca dentro de su espacio de features. El resto es simplemente una variación de ajustes de los metaparametros del word2vec, como, por ejemplo, el tamaño de la ventana.

4.2. Evaluación de performance

Para este proyecto se utilizará como baseline el aproach estándar basado en redes neuronales. Es decir, se evaluara como base la clasificación generada utilizando únicamente *word embedings* y la estructura de red propuesta en el paper de Giuseppe Attardi [4]. A continuación, se valorarán las modificaciones propuestas, intentando mejorar, en un principio, la efectividad sobre el set de pruebas.

5. Tecnologías

5.1. Bibliotecas

5.1.1. Gensim

Gensim[5] es una biblioteca escrita en Python para el modelaje de tópicos y la indexación de documentos, diseñada para trabajar con conjuntos de textos de gran tamaño. Su uso se ha expandido tanto en el ámbito comercial como en el académico, apuntando especialmente al área de procesamiento del lenguaje natural y a la búsqueda y recuperación de la información.

Sus características principales son:

- Utiliza las extensiones científicas de Python NumPy[6] y SciPy,[7] y se encuentra optimizado a través Cython[8].
- Todos sus algoritmos están diseñados para procesar entradas mucho mayores que la memoria RAM disponible.
- Procesamiento distribuido.

De Gensim se utilizará su módulo de Word2Vect para generar los embedings necesarios para la clasificación.

5.1.2. Tensorflow

TensorFlow[9] es una biblioteca de código abierto para aprendizaje automático desarrollada por Google. Permite crear sistemas capaces de construir y entrenar redes neuronales. Actualmente es utilizado tanto para la investigación como para la elaboración de productos de Google.

TensorFlow se utilizará como servicio para la clasificación mediante redes neuronales que requiere el sistema. Las capacidades de montarlo como un servicio independiente y la de levantarlo sobre un cluster lo hacen ideal para permitir el escalado del sistema.

5.1.3. Rabbit MQ

Rabbit MQ[10] es un software de negociación de mensajes (*broker*) de código abierto, y se inserta dentro de la categoría de middleware de mensajería. Implementa el estándar Advanced Message Queuing Protocol (AMQP). El servidor RabbitMQ está escrito en Erlang y utiliza el framework Open Telecom Platform (OTP) para construir sus capacidades de ejecución distribuida y conmutación ante errores.

Rabbit MQ se utilizará como mecanismo de comunicación entre los distintos workers del sistema. Sus características permiten vincular workers de forma local o de forma distribuida, tanto mediante una red local como por Internet.

5.1.4. Docker

Docker[11] es una herramienta capaz de automatizar el despliegue de aplicaciones dentro de contenedores de software, proporcionando así una capa adicional de abstracción y automatización en el nivel de virtualización de sistema operativo sobre Linux. Docker hace uso de las utilidades de aislamiento de recursos del núcleo de Linux como los cgroups y espacios de nombre del kernel, para permitir que çontenedoresïndependientes se ejecuten como una instancia única de Linux, evitando así la elevada sobrecarga en el arranque y mantenimiento de máquinas virtuales convencionales como virtual box.

Docker podría considerarse solo una herramienta de desarrollo, ya que durante la realización del presente se empleará mayoritariamente para poder levantar los servicios de forma amigable y de proveer un entorno aislado para las pruebas. Pero este, a su vez, juega un papel importante en la etapa de deploy. Utilizándolo se puede agregar fácilmente más workers a la red, sin necesidad de tener que configurar el entorno en las nuevas máquinas.

5.2. Herramientas y otras tecnologías

Categoría	Herramientas
Lenguajes de programación	• Python[12]
Herramientas web	• Boostrap[13] • FusionCharts[14] • Flask[15] • Gunicorn[16]
Entornos de desarrollo	• Pycharm[17]
Servidor web	• Heroku[18]
Control de versiones	• Git[19]

6. Alcance

El alcance del presente Trabajo Profesional comprende:

- Desarrollo del sistema de clasificación distribuida.
- Desarrollo e implementación de modificaciones con su correspondiente evaluación de desempeño en base a métricas preestablecidas.
- Interfaz web para muestra de resultados.
- Interfaz web para clasificación y validación de las clasificaciones automáticas.
- Sistema de re-entrenamiento.

7. Plan de trabajo

7.1. Equipo de trabajo

El equipo de trabajo estará conformador por:

- Tutor: Lic. Luis Argerich.
- Desarrolladores: Gaston Alberto Martinez.

7.2. Metodología

Para la ejecución del proyecto se usará una metodología ágil basada en SCRUM. La misma consistirá en definir una serie de iteraciones con fechas pautadas de reuniones de avance y entregas.

Al inicio de cada iteración se conformará una priorización de los requerimientos pendientes de desarrollo. Posteriormente, se procederá a su implementación y, para finalizar cada iteración, se hará una presentación de los entregables pautados.

Las reuniones, entregas, presentaciones y la priorización de los requerimientos para cada iteración se realizará en con el tutor del trabajo.

7.3. Estimación

Se muestra a continuación el listado de tareas necesarias para alcanzar los objetivos descritos anteriormente. Todos los esfuerzos se encuentran expresados en horas.

# It.	Descripción	Esf.
–	Propuesta	20
1	Investigación previa	60
	Interfaz web para el Tagger	20
2	Clasificador (pipeline): diseño de la estructura	15
	Clasificador (pipeline): implementación de los workers	35
	Clasificador (pipeline): integracion con Word2Vect de Gensim	20
3	Clasificador (clasificador): Implementacion de Tensorflow	10
	Clasificador (clasificador): Diseño e implementacion de la red	30
4	Pruebas de rendimiento: análisis de los embedings	15
	Pruebas de rendimiento: análisis de los resultados	15
	Pruebas de rendimiento: modificaciones	30
5	Interfaz web para muestra de resultados	30
6	Sistema de re-entrenamiento automatizado	40
7	Pruebas del sistema en funcionamiento	28
8	Análisis de datos e informes de resultados	20
–	Reuniones	15
–	Presentación	24
–	Clasificación manual del set de entrenamiento	30

Además se debe tener en cuenta el tiempo dedicado a la administración del proyecto, estimando un 15 % del tiempo de desarrollo, lo cual resulta en:

Descripción	**Esfuerzo**
Iteración 1	80
Iteración 2	70
Iteración 3	40
Iteración 4	60
Iteración 5	30
Iteración 6	40
Iteración 7	28
Iteración 8	20
Otros	89
Administracion	60
Esfuerzo total	517

Referencias

[1] *Rae*. Diccionario de la lengua española.
http://dle.rae.es/

[2] *The Tanl Pipeline*. G. Attardi, S. Dei Rossi, M. Simi. The Tanl Pipeline. Proc. of LREC Workshop on WSPP, Malta, 2010.
https://www.di.unipi.it/ attardi/Paper/LREC10%20Pipeline.pdf

[3] *Sentiment Analysis: Beyond Polarity*. Thesis Proposal Phillip Smith October 2011.
https://www.cs.bham.ac.uk/ pxs697/publications/documents/rsmg3.pdf

[4] *UniPI at SemEval-2016 Task 4: Convolutional Neural Networks for Sentiment Classification*. Giuseppe Attardi, Daniele Sartiano, Proceedings of SemEval-2016, Junio 16-17, 2016.
http://www.aclweb.org/anthology/S16-1033

[5] *Gensim*. Biblioteca de modelado de tópicos.
http://radimrehurek.com/gensim/

[6] *NumPy*. Paquete de Python para procesamiento numérico.
http://www.numpy.org/

[7] *SciPy*. Paquete de Python para computación científica.
http://www.scipy.org/

[8] *Cython*. Extensiones en lenguaje C para Python.
http://cython.org/

[9] *TensorFlow*. Biblioteca opensource para Machine Learning
https://www.tensorflow.org/

[10] *RabbitMQ*. Open source message broker.
https://www.rabbitmq.com/

[11] *Docker*. Entorno de virtualizacion.
https://www.docker.com/

[12] *Python 3.6*. Lenguaje de programación.
https://www.python.org/downloads/release/python-360/

[13] *Boostrap*. Toolkit para diseño de interfaces web.
http://getbootstrap.com/

[14] *FusionCharts*. Biblioteca de javascript para creación de graficos.
`https://www.fusioncharts.com/`

[15] *Flask*. Microframework para crear aplicaciones web.
`http://flask.pocoo.org/`gráficos

[16] *Gunicorn*. Servidor WSGI HTTP para UNIX.
`http://gunicorn.org/`

[17] *PyCharm*. IDE para desarrollos basados en Python.
`https://www.jetbrains.com/pycharm`

[18] *Docker*. Plataforma de servicios en la nube para deploy de aplicaciones.
`https://www.heroku.com/`

[19] *Git*. Herramienta para el control de versiones.
`http://git-scm.com/`