Distribución de un Analizador de Contenido de Twitter utilizando el Framework Hadoop Map-Reduce

Maria Florencia Rodriguez[1,2]

[1] *PLADEMA, Universidad Nacional del Centro, Tandil, Buenos Aires, Argentina*
{mfrodri}@exa.unicen.edu.ar
[2] *Comisión de Investigaciones Científicas de la Provincia de Buenos Aires, La Plata, Buenos Aires, Argentina*

Resumen

En este trabajo presentamos la utilización de una plataforma distribuida para analizar diferentes estrategias de cómo crear perfiles de usuario en base a los intereses extraídos de los *tweets*. Estudiamos cómo esto beneficia el entendimiento de la semántica de las actividades de Twitter utilizándolos como motores para la recomendación de contenido y medimos los resultados en termino de tasa de error, exactitud y valor F. Luego proponemos crear perfiles de usuario combinando las estrategias planteadas con el fin de obtener un modelo más preciso. Para mejorar la eficiencia del cálculo dada la gran cantidad de datos , proponemos utilizar un entorno distribuido bajo un modelo de programación Map-Reduce con el objetivo de reducir los tiempos de análisis de información, al mismo tiempo que realizamos el procesamiento de texto en forma paralela.

Keywords: Hadoop Map-Reduce, minería de texto, procesamiento de lenguaje natural, sistemas de recomendación, Twitter

1. Introducción

Con más de 190 millones de usuarios y más de 65 millones de *posts* por día[1], Twitter[2] es el servicio de *micro-blogging* más popular entre las redes sociales. En Twitter los usuarios publican mensajes llamados *tweets* que están formados por texto plano de corta longitud y se muestran en la página principal del usuario.

En contraste con otras redes sociales como por ejemplo Last.fm[3], donde se deducen los gustos musicales, o Flickr[4] que ofrece información para inferir los intereses de los usuarios en lugares o eventos; los temas discutidos en Twitter no están restringidos a un determinado dominio. Los usuarios de Twitter pueden discutir acerca de cualquier tema que les interese o preocupe.

Dada la gran cantidad de contenido que circula en Twitter uno de los principales desafíos de esta tecnología, es poder brindar al usuario información y contenido que resulte de su interés.

La representación de la semántica de las actividades individuales de Twitter y el modelado de los intereses de los usuarios permite personalización y de esta forma, compensar la sobrecarga de información.

Como consecuencia de la diversidad y lo cambiante que son los temas que se discuten en Twitter, los perfiles generados a partir de los *tweets*, son beneficiosos para otras aplicaciones Web. Sin embargo, inferir automáticamente el significado semántico de mensajes de Twitter no es un problema trivial.

Si bien existen estudios acerca de las estructuras sociales en Twitter, se ha realizado escasa investigación en el área de comprensión de la semántica de las actividades individuales de Twitter y en deducir intereses de los usuarios de estas actividades. Dada la brevedad de la longitud de los *tweets*, dar sentido a los mensajes individuales y explotarlos para el modelado de usuario representan un desafío. Además se deben aplicar diversos mecanismos de procesamiento de datos ya que es muy común que los usuarios utilicen abreviaciones y no respondan a las reglas léxicas y ortográficas del lenguaje.

Al mismo tiempo la enorme cantidad de datos que circula a través de la red social hace necesario que se utilicen nuevos enfoques para el procesamiento masivo de datos. En algunas investigaciones, como Abdullah [1], utilizan el *framework* Hadoop Map Reduce para procesar datos y obtener un *ranking* de las urls más visitadas; sin embargo, se ha hecho poco hincpié en el procesamiento de grandes volúmenes de datos para obtener información acerca de los intereses de los usuarios en tiempo real.

En este trabajo realizamos un análisis acerca de cómo deducir los intereses de los usuarios de Twitter para recomendar mensajes de otros usuarios que tengan intereses similares. A la vez, proponemos el uso del *framework* Hadoop Map Reduce

[1] http://www.techcrunch.com/2010/06/08/twitter-190-million-users/
[2] http://www.twitter.com/
[3] http://www.last.fm/
[4] http://www.flickr.com/

para procesar los datos en un entorno distribuido.

2. Perfiles de usuario para la recomendación de contenido

La propuesta de solución al problema de la recomendación de contenido se basa en la exploración del texto de los mensajes de Twitter para deducir intereses, y en base a ellos modelar el perfil. Luego, ese perfil es utilizado como motor de clasificación para recomendar *tweets* que tengan contenido se aproxime a los intereses del usuario.

Perfil de usuario. Un perfil de usuario se define como una lista de palabras claves relevantes para el usuario [2]. Se pueden enmarcar dentro del proceso de recomendación como una estructura de almacenamiento de información sobre las preferencias del usuario.

Los perfiles de usuario son modelados mediante vectores (Eq (1)), en los que cada elemento es un concepto que representa el intereses de un usuario en determinado tema. Cada concepto es ponderado de acuerdo a una función que mide el nivel de importancia que tiene dicho elemento para el usuario (Eq (2)).

Definición 1 (Perfil de usuario). *Formalmente, el perfil de usuario $P(u)$ correspondiente al usuario u, del conjunto de usuarios U, es un vector de pares $(c, w(u,c))$ donde c es un concepto, del conjunto de conceptos C, extraído de los tweets $T_u \in T$ del usuario $u \in U$, y $w(u,c)$ es el resultado de aplicar una función de ponderación w (Eq (2)) que indica que tan significativo es el concepto c para el usuario $u \in U$.*

$$P(u) = \{c, w(u,c) : u \in U; c \in C\} \qquad (1)$$

La función $w(u,c)$ es calculada utilizando la frecuencia; de esta forma, el peso $w(u,c)$ de un concepto $c \in C$ queda determinado por el número de veces en los que el usuario $u \in U$ se refiere al concepto c en el conjunto de *tweets* $T_u \in T$. Por ejemplo, un valor de ponderación $w(u, technology) = 5$ indica que el usuario $u \in U$ ha escrito 5 veces la palabra "*technology*" en su conjunto de *tweets* $T_u \in T$. Este valor es dividido por el máximo valor de frecuencia.

$$w(u,c) = \frac{f_{u,c}}{max f_{u,c}} \qquad (2)$$

En este trabajo construimos perfiles tomando como entrada todo el historial de *tweets*. Algunos autores como Abel et al. [3] realizan un análisis donde estudian cómo los perfiles de usuario varían cuando son restringidos según diferentes patrones temporales.

Para deducir intereses de usuario, Abel et al. [4, 5] proponen vincular *tweets* con artículos de noticias con el objetivo de enriquecer el contenido del *tweet* y luego extraen entidades y tópicos para deducir los intereses.

Dado que los temas que se discuten en Twitter son variados y no están restringidos a un dominio en particular, no siempre se puede vincular un *tweet* con una noticia, entonces en este trabajo planteamos diferentes estrategias de procesamiento que se basan en analizar únicamente el contenido del *tweet*.

De esta forma, los perfiles creados en este trabajo varían según tres alternativas, de acuerdo a las estrategias de exploración del contenido de *tweets*.

2.1. Estrategias de procesamiento

En la Definición 1 se ha introducido la noción de *concepto*. Un concepto es una unidad cognitiva de significado, es un elemento que aporta información acerca de los intereses de los usuarios. Durante el desarrollo de este trabajo hemos planteado tres estrategias de extracción de conceptos.

Una primera estrategia, ingenua, consiste en procesar cada *tweet* como una tira de *tokens*, y luego calcular el peso de cada *token*. Este mecanismo considera todas las palabras que forman parte del texto del *tweet* como conceptos.

Definición 2 (Estrategia basada en bolsa de palabra). *Formalmente, un tweet $t_u \in T$ es representado mediante un vector $t_u = \{(c_1, f_1), (c_2, f_2), ...(c_n, f_n)\}$ donde cada elemento c_i corresponde a un término extraído de t_u y f_i es la frecuencia con la que c_i aparece en t_u.*

El perfil $P_B(u)$ para un usuario $u \in U$ es el conjunto de todos los tokens (excepto los stop-words) que aparecen en el conjunto de tweets $T_U \in T$ de u, junto con los valores de ponderación w_i para cada x_i calculado en base a la función $w(u,x)$.

El valor $w(u,x)$ es el peso asociado al token x para el usuario $u \in U$ correspondiente al conjunto U de usuarios.

Cada elemento del vector $P(u)$, es un par $(x, w(u,x))$ donde x corresponde a un *token* y $w(u,x)$ es la función de ponderación definida en Eq (2).

En esta estrategia, el proceso de extracción de conceptos se realizó utilizando el algoritmo de tokenización de la librería Lucene[5].

Lucene es un API para la recuperación de información. Es un proyecto *open source*, implementado en Java, es miembro del Apache Software Foundation y se distribuye bajo la Apache Software Licence.

[5] http://lucene.apache.org/core/

El algoritmo tokenizador de Lucene acepta una cadena de caracteres como entrada (un *tweet*), procesa la cadena para dividirla en palabras individuales, y emite una cadena de componentes léxicos como salida.

Las palabras vacías (*stop words*) son palabras que por su frecuencia y/o semántica no poseen valor discriminatorio alguno, es decir no permiten distinguir un texto de otro en una colección. Habitualmente se trata de artículos, pronombres, preposiciones, verbos muy frecuentes, adverbios, etc.

La eliminación de *stop words* se realiza chequeando el contenido del texto contra un listado disponible. Lucene provee un módulo procesador de texto, llamado *StandardAnalyzer*, que realiza una tokenización que elimina *stop words*. Durante el proceso de conversión a *token*, el *StandardAnalyzer* extrae el texto que se debe analizar mientras se le aplica lógica de transformación como ser el uso de *stop words*. Consigo mismo trae una lista de *stop words* comunes en Inglés.

Las estrategia planteada hasta el momento realiza un análisis léxico, y si bien obtiene el conjunto de conceptos a los que el usuario se refiere con mayor frecuencia, no construye perfiles con valor semántico.

Entonces, planteamos dos estrategias que realizan un análisis de la semántica del texto. Una de ellas está basada en reconocimiento de nombres de entidades y la otra en extracción de conceptos de DBpedia.

El reconocimiento de nombres de entidades (NER) etiqueta la secuencia de palabras de un *tweet* como nombre de cosas, por ejemplo nombres de personas, compañías, o lugares, y genera perfiles basados en entidad $P_E(u)$. Utilizando esta técnica, un modelo de perfil basado en entidades construye un perfil en función de los intereses del usuario en un determinado conjunto de entidades, como las personas, organizaciones o eventos. Los *tweets* son definidos como un vector donde cada elemento contiene las entidades usadas y la frecuencia.

Definición 3 (Estrategia basada en extracción de entidades). *El perfil basado en entidad $P_E(u)$ para un usuario $u \in U$ es el conjunto de nombres de entidades $e_i \in E$ que aparecen en el conjunto de tweets $T_u \in T$ de u, junto con los valores de ponderación w_i para cada e_i calculado en base a la función $w(u, e)$.*

El valor $w(u, e)$ es el peso asociado con la entidad e para el usuario u. E y U corresponden al conjunto de entidades y de usuarios respectivamente.

De esta forma, cada elemento del vector $P(u)$, es un par $(e, w(u, e))$ donde e corresponde a una entidad y $w(u, e)$ es la función de ponderación definida en Eq (2).

Las entidades son extraídas directamente a partir del contenido del *tweet* utilizando TwitIE[6]. El tokenizador de TweiIE es una adaptación del tokenizador de ANNIE [7].

A menudo, los usuarios suelen publicar *tweets* acerca de determinados conceptos que no corresponden a entidades; entonces, se ha planteado una estrategia que consiste en mapear el contenido de los *tweets* a recursos de DBpedia (que es la representación estructurada de Wikipedia[6]).

Definición 4 (Estrategia basada en extracción de recursos de DBpedia). *El perfil basado en conceptos de DBpedia $P_D(u)$ para un usuario $u \in U$ es el conjunto de conceptos $d_i \in D$ que aparecen en el conjunto de tweets $T_u \in T$ de u, junto con los valores de ponderación w_i para cada d_i calculado en base a la Eq (2) $w(u, d)$.*

El valor $w(u, d)$ es el peso asociado con la entidad e para el usuario u. E y D corresponden al conjunto de conceptos de DBpedia y de usuarios respectivamente.

De esta forma, cada elemento del vector $P(u)$, es un par $(d, w(u, d))$ donde d corresponde a un concepto de DBpedia y $w(u, d)$ es la función de ponderación definida en Eq (2).

Para construir perfiles utilizando esta estrategia, se toma como entrada el conjunto de *tweets* y se intenta buscar el recurso correspondiente en DBpedia.

2.2. Perfil de usuario combinado

El tipo de conceptos que forman el conjunto C determina el tipo de perfil, pudiendo ser una tira de *tokens* $P_B(u)$, un vector de entidades $P_E(u)$ o un conjunto de conceptos $P_D(u)$. Adicionalmente hemos planteado la posibilidad de armar perfiles que combinan las estrategias explicadas hasta el momento. Formalmente un perfil de usuario que combina estrategias se define como sigue:

Definición 5 (Perfil de usuario combinado). *Un perfil de usuario $P_M(u)$, para un usuario $u \in U$ que combina los perfiles $P_B(u)$, $P_E(u)$ y $P_D(u)$ generados a través de las estrategia basada en bolsa de palabras (sin palabras vacías), en entidades y en recursos de DBpedia, respectivamente, se define mediante la ecuación:*

$$P_M(u) = \{n1 * P_B(u) + n2 * P_E(u) + \\ +n3 * P_D(u) : u \in U, n1 + n2 + n3 = 1\} \quad (3)$$

En la definición anterior, $n1$, $n2$ y $n3$ indican el peso que se le da a cada perfil. De esta forma,

[6] http://www.wikipedia.org/

valores $n1 = n2 = n3 = 0{,}33$ indican que todos los perfiles tendrán el mismo peso; valores tales como $n1 = 0$ y $n2 = 0,5$ y $n3 = 0,5$ indican que los perfiles basados en bolsa de palabras no tendrán peso en la ponderación, y la influencia será dada por los perfiles basados en entidades y recursos de DBpedia de forma equitativa.

3. Modelo de recomendación

La creación de perfiles de usuario tiene como objetivo armar perfiles en base a los intereses extraídos a partir del contenido de los *tweets*. Luego esos perfiles son usados como modelos, para clasificar si determinado *tweet* contiene información que puede ser de interés para el usuario o no.

El *dataset* utilizado durante el desarrollo de este trabajo está compuesto por 2316204 *tweets* de 1619 usuarios[7]. Para el modelado de perfiles, se consideran los *post* de 500 usuarios que cuentan con una cantidad mínima de 100 *tweets*.

El conjunto de *tweets* utilizado para evaluación fue dividido en dos clases: una formada por *tweets* etiquetados como "*Interesa*" (*ejemplos positivos*), y otra clase formada por *tweets* marcados como "*No interesa*" (*ejemplos negativos*), para cada usuario del conjunto analizado.

El conjunto de los *tweets* etiquetados como "*Interesa*" fueron obtenidos de los *tweets* de cada usuario. Los *post* del conjunto de usuarios analizados, fue dividido en dos partes: un 70 % fue utilizado como *conjunto de entrenamiento* y el 30 % restante se reservó para *evaluar* las diferentes estrategias. Entonces, el 30 % de *tweets*, al ser publicados por el usuario en cuestión, fue marcado como "*Interesa*" para él.

Los *tweets* utilizados para formar parte de los *ejemplos negativos*, fueron seleccionados a partir del conjunto de *tweets* que no forman parte del conjunto de *tweets* ni de *re-tweets* del usuario a analizar.

Cada tupla se supone que pertenece a una clase predefinida, dada por uno de los atributos, llamada *etiqueta de clase*. El conjunto de todas las tuplas usadas para la construcción del modelo se llama *conjunto de entrenamiento*.

3.1. Algoritmo recomendador de *tweets*

Con el fin de evaluar la efectividad de las estrategias de modelado de perfiles de usuario como motores de recomendación, hemos aplicado un algoritmo basado en clasificar contenido de acuerdo a la similitud entre el *tweet* a recomendar y el perfil de usuario. De esta forma, el problema de la recomendación se convierte en una búsqueda y clasificación, donde el perfil es utilizado como motor de clasificación.

Definición 6 (Algoritmo de recomendación). *Dado un perfil de usuario $P(u)$ representado mediante un vector y un conjunto de tweets $T = \{P(t_1), \dots P(t_n)\}$ representadas mediante perfiles (con la misma representación vectorial), el algoritmo de recomendación clasifica los tweets según la similitud de coseno:*

$$sim_{coseno}(P(u), P(t_i)) = \frac{P(u).P(t_i)}{||P(u)||.||P(t_i)||} \quad (4)$$

Para evaluar el comportamiento de los diferentes tipos de perfiles, se considera que un *tweet* t_i es importante para un determinado usuario u, si forma parte de su conjunto de *tweets* T_u (incluyendo los *re-tweets*).

La evaluación de las diferentes estrategias de construcción de perfiles de usuario como motores recomendadores de contenido fue realizada mediante *Holdout*. Esta técnica de validación consiste en dividir el *dataset* en dos partes: una utilizada para realizar la construcción del perfil (llamada *conjunto de entrenamiento*), y otra para realizar evaluaciones (*conjunto de evaluación*). A partir del conjunto de entrenamiento, se crean diferentes modelos en base a la estrategia elegida (*entrenamiento del modelo*) que luego son evaluadas utilizando el conjunto de evaluación.

3.2. Función de similitud

El análisis de precisión consiste en calcular la similitud entre el perfil de cada usuario $P(u_i)$ y los perfiles $P(t_j)$ de cada uno de los *tweets* $t_j \in T$ que forman parte del conjunto de prueba.

La función de similitud (Eq (4)) se mide mediante el coseno entre dos perfiles de usuario representados mediante vectores.

La medida del coseno es una medida de similaridad de patrones de datos, que permite comparar usuarios o documentos, ya que el coseno mide el ángulo entre dos vectores en un espacio N-dimensional [8, 9].

Cuando los perfiles son idénticos, la función da como resultado 1, mientras que por el contrario, para perfiles totalmente diferentes, el coseno da cero. Si el resultado de dicha función entre un perfil $P(u_i), u_i \in U$ y un *tweet* $t_j \in T$ es mayor a determinado umbral, el *tweet* t_j es clasificado como "*Interesa*. Sino, como "*No interesa*".

4. Procesamiento masivo de datos bajo Hadoop Map-Reduce

Dada la gran cantidad de texto para analizar, los tiempos de procesamiento resultan elevados,

[7] http://wis.ewi.tudelft.nl/umap2011/#dataset/

entonces, para reducirlos se utilizó el modelo de programación MapReduce[10] implementado bajo el framework Hadoop[8].

Hadoop[11] es una infraestructura digital de desarrollo creada en código abierto bajo licencia Apache. Básicamente, permite desarrollar tareas intensivas de computación masiva, dividiéndolas en piezas menores y distribuyéndolas en un conjunto de máquinas.

MapReduce es el nombre dado a la combinación de dos procesos separados, necesarios para la extracción de valores de un gran número de orígenes de datos distintos. Permite procesar datos almacenados en un *clúster* de forma paralela haciendo uso de un sistema de archivos distribuido llamado HDFS. El *clúster* posee un conjunto de servidores. Cada servidor posee un conjunto de datos locales que serán procesados por él.

HDFS es el sistema de archivos utilizado por defecto en Hadoop. Es un sistema distribuido estructurado en bloques. Los archivos individuales se parten en bloques de un tamaño fijo. Estos bloques se almacenan a lo largo del *clúster* de la máquina donde están los datos. En este trabajo el conjunto de archivos guarda el texto contenido en los *tweets* analizados.

Como su nombre lo indica, el modelo MapReduce, presenta dos fases: *map* y *reduce*. La fase *map* funciona como extractor y asigna valores a determinadas claves para un único documento. Entonces, tomando como entrada los *tweets* del conjunto de usuarios, descompone cada *tweet* en un vector de *tokens*, entidades o conceptos, según la estrategia utilizada.

La fase *reduce* realiza la función de acumulación y combina las claves de múltiples documentos para crear un valor reducido (combinado) único para cada clave a partir de los múltiples valores generados. En este caso, para cada usuario, por cada concepto extraído de cada *tweet* en la etapa *map*, la fase *reduce* calcula el valor de la Eq (2).

Desde la perspectiva del flujo de datos, la ejecución de MapReduce consiste en M tareas *map* y R tareas reduce independientes, cada una de las cuales puede depender de M tareas *map*. Generalmente, los resultados intermedios se particionan en R trozos para R tareas reduce [10].

4.1. Características generales del dataset

Como se mencionó anteriormente, el *dataset* utilizado durante el desarrollo de este trabajo se compone de un gran volumen de *tweets*. Para obtenerlos, durante un período de más de dos meses Tao et al. [12] han seguido flujos de información Twitter de más de 20.000 usuarios. Finalmente, el conjunto de datos contiene 2.316.204 *tweets*.

Estos datos están disponibles para ser importados en una base de datos relacional. Con el objetivo de que sean procesados como un sistema de archivos en Hadoop, los transformamos en archivos de texto plano donde para cada usuario, existe un archivo en el cual almacenamos sus mensajes. Cada línea almacena un *tweet* de un usuario.

4.2. Mapper

En esta etapa, se procesan los archivos línea por línea y se analizan los *tweets* según la estrategia elegida, pudiendo realizar una tokenización para armar perfiles basados en bolsa de palabras, realizar un NER para generar perfiles basados en entidades o bien, vincular *tweets* a recursos de DBpedia para armar perfiles basados en conceptos.

Durante la fase *map* se recogen los datos de entrada produciendo como salida una lista de conceptos (c_i) que aparecen mencionados en los *tweets*. Aquí cada c_i corresponde a un concepto según Eq (1).

Estos elementos son agrupados por clave -es decir, por concepto- y pasados a la función *reduce* que se encarga de procesarlos y generar un resultado agrupado de los mismos.

Es necesario destacar que el modelo realiza esta tarea para varios *tweets* de forma paralela. De esta forma, se tienen varias tareas *map* ejecutando de forma paralela.

4.3. Reducer

El *reducer* es la parte del algoritmo que recoge, agrupados por concepto, todos los valores emitidos en la fase *map*. Para cada concepto c_i se calcula la frecuencia de aparición $f_{u,c}$ (ver Eq (2)) en el conjunto de *tweets* $T_u \in T$ de un usuario $u \in U$.

5. Resultados

5.1. Evaluación por estrategia

Si comparamos el comportamiento de las estrategias con respecto a la tasa de error (Fig. 1) vemos que la que menor error posee es la estrategia basada en bolsa de palabras. Luego siguen las estrategias basadas en extracción de conceptos de DBpedia y en NER. La estrategia que presenta mayor exactitud es la basada en bolsa de palabras; luego siguen las estrategias basadas en NER y extracción de recursos de DBpedia. Con respecto al valor F se puede ver que la construcción de perfiles con la estrategia basada en bolsa de palabras y finalmente la basadas en contenido semántico.

[8] http://hadoop.apache.org/

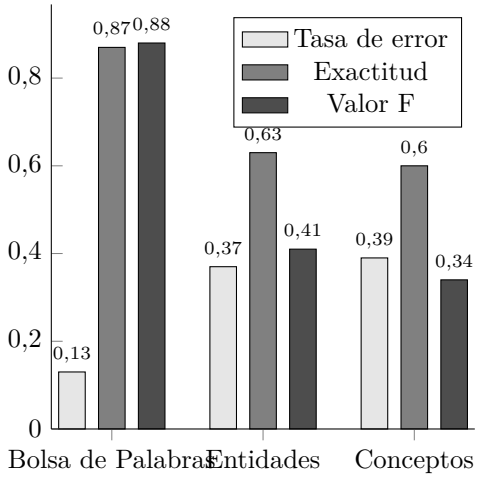

Figura 1: Tasa de error, exactitud y valor F para las estrategias.

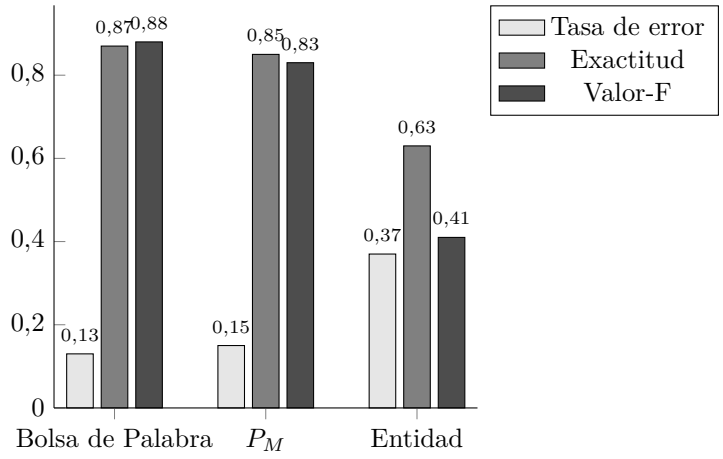

Figura 2: Tasa de error, exactitud y valor-F para las estrategias basada en bolsa de palabras, entidades y P_M.

Si bien la evaluación indica que las estrategias que presentan mayor tasa de error y menor exactitud son las basadas en NER y en extracción de recursos de DBpedia, se puede decir que la estrategia basada en bolsa de palabras construye perfiles carentes de contenido semántico y los conceptos que ocurren con mayor peso se refieren a términos usados frecuentemente por el usuario. Las estrategias basadas en NER y extracción de recursos de DBpedia filtran esos términos y en consecuencia, tienen una tendencia a tener mayor número de desaciertos durante la etapa de evaluación del modelo.

Además, se observó que este comportamiento es consecuencia del hecho de que las estrategias basadas en NER y en extracción de conceptos filtran contenido no interesante para el usuario con 100 % de precisión, pero no sucede lo mismo al clasificar contenido de interés para el usuario, ya que a menudo filtran contenido que es interesante para el usuario.

5.1.1. Perfil combinado

Para analizar cómo varía la recomendación de un perfil generado a partir de las combinaciones de las estrategias planteadas, hemos construido perfiles según Eq (3), asignando las ponderaciones mostradas a continuación (Eq (5)).

$$P_M(u) = \{0{,}7 * P_B(u) + 0{,}3 * P_E(u) + 0 * P_D(u) : u \in U\} \quad (5)$$

Al comparar las métricas arrojadas por el perfil combinado (Eq (5)) con los perfiles basado en bolsa de palabras y basado en entidad, se puede observar que el perfil combinado tiene menor tasa de error que el perfil basado en entidad (Fig. 2). A su vez, este valor es muy cercano al del perfil basado en bolsa de palabra. Lo mismo sucede con la exactitud y el valor F. Sin embargo, la principal ventaja que incorpora el perfil combinado con respecto al perfil basado en bolsa de palabra es un aumento en la precisión al momento de filtrar el contenido que no es interesante para el usuario. Por otro lado, posee la desventaja de generar un modelo de mayor tamaño al de los perfiles que lo componen.

5.2. Velocidad de procesamiento

Adicionalmente, analizamos los tiempos de ejecución al procesar los datos utilizando un único clúster, que cuenta con la versión Hadoop 2.6.4 previamente instalada en una máquina local con Windows 8. Al ejecutar los diferentes métodos de procesamiento sobre el *dataset* presentado en la Sección 4.1., pudimos observar que los perfiles fueron construidos con mayor velocidad que al realizar los mismos procesamientos sin utilizar un sistema de archivos distribuido.

A la vez, observamos que la ejecución en paralelo de varias tareas *map* implementadas bajo el modelo de programación Map-Reduce beneficia el aprovechamiento de CPU, al utilizar gran parte de la capacidad de procesamiento y al mismo tiempo reduce el tiempo de construcción de perfiles de usuario.

6. Conclusiones y trabajo futuro

En este trabajo se ha realizado un análisis de distintas técnicas de recuperación de información para recomendación de contenido en Twitter. A partir de los intereses extraídos de los *tweets* se han construido perfiles de usuario que son utili-

zados como motores de recomendación. La recomendación se ha realizado comparando el perfil del usuario con el ítem a recomendar. Luego el comportamiento fue validado mediante *Holdout*.

Por otro lado, al analizar los datos en un entorno distribuido se reducen notablemente los tiempos de procesamiento, a la vez que se aprovecha mejor la capacidad de procesamiento de la máquina como consecuencia de la ejecución paralela de varios procesos en simultaneo.

La información extraída de Twitter puede ser utilizada por diversas aplicaciones que se valen de los perfiles de usuario para recomendar contenido. El presente trabajo ha sido orientado a la recomendación de mensajes, sin embargo puede ser extendido para recomendar otro tipo de contenido, como por ejemplo noticias, otros usuarios de la red, páginas web.

Además, se pueden plantear otras alternativas para extraer información. Por ejemplo los *Hashtags* suelen ser buenos indicadores del contenido de los mensajes. Por otro lado, los *tweets* pueden ser contextualizados explotándolos *links* que incluyen o vinculándolos a conferencias o eventos.

Referencias

[1] I. B. Abdullah, *Incremental pagerank for twitter data using hadoop*. PhD thesis, Master's Thesis. University of Ediburgh. 2010. Accessed September 10. 2013. Reference Source, 2010.

[2] J. Tang, L. Yao, D. Zhang, and J. Zhang, "A combination approach to web user profiling," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 5, no. 1, p. 2, 2010.

[3] F. Abel, Q. Gao, G.-J. Houben, and K. Tao, "Analyzing temporal dynamics in twitter profiles for personalized recommendations in the social web," in *Proceedings of the 3rd International Web Science Conference*, p. 2, ACM, 2011.

[4] F. Abel, Q. Gao, G.-J. Houben, and K. Tao, "Analyzing user modeling on twitter for personalized news recommendations," in *User Modeling, Adaption and Personalization*, pp. 1–12, Springer, 2011.

[5] F. Abel, Q. Gao, G.-J. Houben, and K. Tao, "Semantic enrichment of twitter posts for user profile construction on the social web," in *The Semanic Web: Research and Applications*, pp. 375–389, Springer, 2011.

[6] K. Bontcheva, L. Derczynski, A. Funk, M. A. Greenwood, D. Maynard, and N. Aswani, "Twitie: An open-source information extraction pipeline for microblog text.," in *RANLP*, pp. 83–90, 2013.

[7] H. Cunningham, D. Maynard, K. Bontcheva, and V. Tablan, "Gate: an architecture for development of robust hlt applications," in *Proceedings of the 40th annual meeting on association for computational linguistics*, pp. 168–175, Association for Computational Linguistics, 2002.

[8] G. Salton, "Developments in automatic text retrieval," *Science*, vol. 253, no. 5023, p. 974, 1991.

[9] R. Krishnapuram, A. Joshi, O. Nasraoui, and L. Yi, "Low-complexity fuzzy relational clustering algorithms for web mining," *Fuzzy Systems, IEEE Transactions on*, vol. 9, no. 4, pp. 595–607, 2001.

[10] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[11] A. Holmes, *Hadoop in practice*. Manning Publications Co., 2012.

[12] K. Tao, F. Abel, Q. Gao, and G.-J. Houben, "Tums: twitter-based user modeling service," in *The Semantic Web: ESWC 2011 Workshops*, pp. 269–283, Springer, 2012.

[13] P. Zikopoulos, C. Eaton, *et al.*, *Understanding big data: Analytics for enterprise class hadoop and streaming data*. McGraw-Hill Osborne Media, 2011.

[14] D. F. ÁlvaroCuesta and M. D. R-Moreno, "A framework for massive twitter data extraction and analysis," *Malaysian Journal of Computer Science*, vol. 27, p. 1, 2014.