Buscando Flexibilidad, Escalabilidad y Mantenibilidad en la Normalización de Tuits

Yerai Doval, Jesús Vilares, and David Vilares

Grupo LYS, Departamento de Computación, Facultade de Informática,
Universidade da Coruña, Campus de A Coruña, 15071 – A Coruña
{yerai.doval, jvilares, david.vilares}@udc.es – `www.grupolys.org`

Resumen Este trabajo describe, desde el punto de vista de la arquitectura y el diseño, el sistema de normalización desarrollado por nuestro grupo para el preprocesamiento de tuits en tareas de Minería de Texto. Nuestras premisas básicas durante su desarrollo han sido flexibilidad, escalabilidad y mantenibilidad. Presentamos también su aplicación práctica en el caso de la normalización de tuits en español.

Keywords: normalización de tuits, arquitectura software, español

1. Introducción

Existen notables diferencias entre el lenguaje "estándar" y el llamado *texting* empleado en tuits y SMS. Gran parte de los fenómenos que caracterizan a este último tienen base fonética y son particulares al idioma [3], destacando, entre otros: ignorar intencionadamente las normas ortográficas y gramaticales; uso de acortamientos, contracciones, etc. para reducir al máximo la longitud del texto; o el empleo de *emoticonos* para manifestar emociones. Los términos resultantes (ej. `salu2` por `saludos`) se denominan *variantes léxicas*.

Por otra parte, la importancia de Twitter y servicios similares como fuente de información en ámbitos como el marketing, *business intelligence*, periodismo, etc. es patente. Por otra parte, tal cantidad de información sólo puede aprovecharse adecuadamente mediante el uso de técnicas de *Minería de Texto*. Sin embargo, los fenómenos antes comentados suponen un serio problema para ello, ya que las herramientas existentes no funcionan adecuadamente con este tipo de textos. Surge entonces la necesidad de *normalizarlos*, transformarlos a lenguaje estándar: `Akb l trbajo`, por ejemplo, se transformaría en `Acabé el trabajo`. Se trata éste de un campo de investigación bastante reciente.

2. Arquitectura Propuesta

El sistema de normalización de tuits aquí presentado ha sido desarrollado tomando como premisas básicas la *flexibilidad*, *escalabilidad* y *mantenibilidad* del mismo. Para ello tomaremos como punto de partida un prototipo anterior desarrollado por nuestro grupo para la normalización de tuits en español [2] y que, aunque funcional, no resultó todo lo flexible y mantenible que deseábamos.

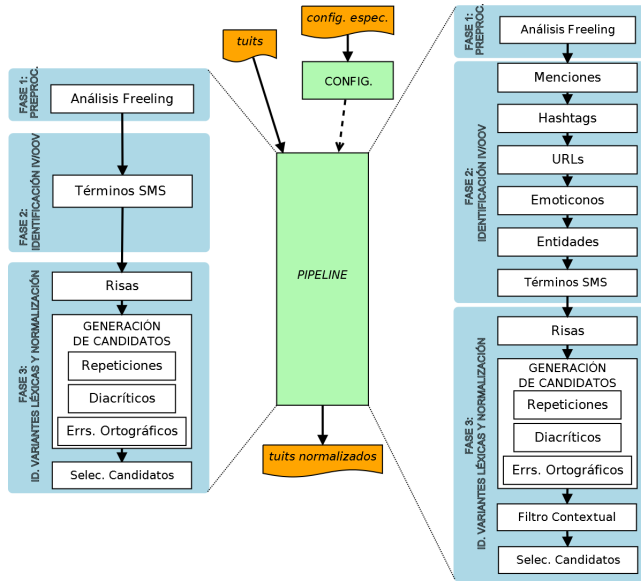

Figura 1. *Pipelines* original (izq.) y actual (der.) integrados en la nueva arquitectura.

El esquema general del sistema original, que hemos mantenido, imita al de [1] y consta de tres fases: *(1)* se *preprocesan* los tuits; *(2)* se identifican las *palabras dentro-de-vocabulario* (IV) del texto en base al léxico del sistema, obteniendo, por eliminación, el conjunto inicial de *palabras fuera-de-vocabulario* (OOV); *(3)* finalmente, procesamos dicho conjunto inicial identificando cuáles son OOV *propias* (palabras correctas pero desconocidas) y cuáles *variantes léxicas*, proponiendo para éstas últimas su forma estándar normalizada; este proceso consta, a su vez, de dos pasos, de forma que primero se aplican técnicas de normalización para generar *normalizaciones candidatas* y se elige luego la más adecuada. Cada una de estas etapas estaba implementada, en el sistema original, mediante *módulos* en un *pipeline*, estructura que nos permitía integrar, intercambiar o eliminar componentes de forma muy sencilla y con las mínimas interdependencias.

2.1. El *Pipeline*

Se decidió pasar a un enfoque *orientado a objetos* (en JAVA) en contraposición con el enfoque *imperativo* del sistema inicial (en PERL), empleando como guía la migración del sistema original al nuevo enfoque (véase parte izquierda de la Figura 1). El sistema se estructura en *procesadores*, antes denominados *módulos*, y cuyo cometido es el de realizar un determinado procesamiento sobre los tuits de entrada de modo que, a su salida, se obtengan candidatos para su normalización.

Siguiendo buenas prácticas de ingeniería, se ha hecho amplio uso de *patrones de diseño*. El primero es el patrón *decorador* que, en nuestro contexto, representa un *pipeline* sencillo, permitiéndonos apilar de forma dinámica un número

arbitrario de procesadores. Su combinación con el patrón *composición* permite agruparlos en *etapas* que siguen compartiendo la misma interfaz de los procesadores básicos, preservando así la flexibilidad del *decorador*. De este modo, la estructura resultante permite la construcción, de forma dinámica, de diferentes configuraciones del *pipeline*, de complejidad diversa y diferentes niveles de abstracción, sin tener que restringirse únicamente a la configuración original.

El patrón *plantilla* nos ha permitido factorizar procesos comunes a buena parte de los componentes, tales como la iteración secuencial por todos los tuits de entrada para un procesamiento particular, lo mismo para todos los términos de un tuit, etc. Esto supuso una gran homogeneización del código, simplificando notablemente su mantenimiento y permitiéndonos centrar nuestros esfuerzos en la implementación concreta de los métodos de procesamiento para cada caso.

Por otra parte, algunos procesadores como el de *normalización de abreviaturas* SMS o el de *corrección ortográfica* hacen uso de herramientas externas susceptibles de ser cambiadas incluso en tiempo de ejecución —de particular interés en entornos multilingües. Asimismo, dichas herramientas deberían poder emplearse dentro de componentes externos a los procesadores, de modo que su lógica pueda ser reutilizada por terceros. Todo esto conlleva desacoplar los procesadores de las implementaciones concretas de los componentes externos que utilicen, lo que hemos conseguido por medio del patrón *inversión de control*.

Asimismo, la comunicación entre los componentes del *pipeline* se realiza mediante ficheros de texto estructurados, lo que nos permite ganar en flexibilidad al poder integrar e intercambiar con suma facilidad nuevos módulos de procesamiento independientemente de su implementación particular [2]. En este caso hemos empleado XML junto con una implementación de *factoría abstracta* para su generación y *parseo*. El uso de este patrón facilitaría también futuras migraciones a otros lenguajes de representación de datos, tales como JSON.

Finalmente, hemos creado un *subsistema de configuración* dinámica que, mediante ficheros XML, permite definir e instanciar la estructura concreta del *pipeline* sobre el que deseamos procesar los tuits. Las ventajas son claras, tanto de cara al mantenimiento como a su empleo en experimentación: participa en el soporte del multilingüismo (definiendo configuraciones que hagan uso de los procesadores correspondientes al idioma en cuestión) y permite realizar experimentos de una forma sencilla, ágil y documentada (el propio fichero de configuración sirve también como documentación), además de no ser preciso modificar el código fuente del sistema.

3. Un Caso Práctico: Normalización de Tuits en Español

La arquitectura propuesta ha sido empleada en el marco de nuestros experimentos de normalización de tuits en español haciendo uso del *corpus* de *test* del TWEET-NORM 2013 [4]. Nuestros experimentos iniciales empleaban la configuración del *pipeline* del prototipo inicial y, a partir de ahí, en iteraciones sucesivas, hemos intentado ir mejorando dichos resultados iniciales, bien por medio de mejoras en algún procesador, bien a través de una redistribución de los mismos

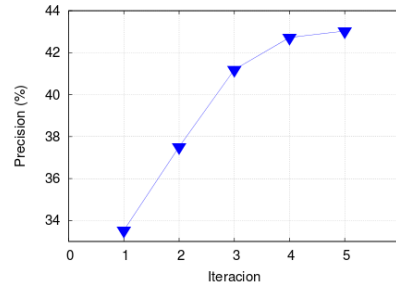

Figura 2. Evolución del sistema en términos de *precisión* (número de decisiones de normalización tomadas correctamente respecto al total de OOV a tratar).

en el *pipeline*, o bien a través de la inclusión de nuevos procesadores. La Figura 2 muestra la evolución de nuestro sistema, desde nuestra configuración inicial hasta la actual (partes izquierda y derecha de la Figura 1, respectivamente).

4. Desarrollo Futuro

Disponiendo ya de un diseño robusto, flexible y escalable, pretendemos seguir mejorando nuestro sistema de normalización de tuits en español. Para ello estamos analizando los resultados obtenidos hasta ahora y los tipos de error cometidos, para así ver en qué sentido deberíamos hacer evolucionar el sistema.

Por otra parte, pretendemos integrar nuestro sistema, a modo de preprocesador, tanto para español como para inglés, en el sistema de *Minería de Opiniones* que está siendo desarrollado en el seno de nuestro grupo [5].

Agradecimientos: Trabajo parcialmente financiado por el Ministerio de Economía y Competitividad y FEDER (TIN2010-18552-C03-02) y por la Xunta de Galicia (CN2012/008).

Referencias

1. B. Han y T. Baldwin. Lexical normalisation of short text messages: makn sens a #twitter. In *Proc. of HLT'11*, pp. 368–378, 2011. ACL.
2. M. A. Alonso, J. Vilares y D. Vilares. Prototipado rápido de un sistema de normalización de tuits: Una aproximación léxica. In I. Alegría et al. [4], pp. 39–43.
3. P. López Rúa. Teaching L2 vocabulary through SMS language: Some didactic guidelines. *Estudios de lingüística inglesa aplicada*, 7:165–188, 2007.
4. I. Alegría, N. Aranberri, V. Fresno, P. Gamallo, L. Padró, I. San Vicente, J. Turmo, y A. Zubiaga (eds.) *Tweet-Norm 2013*, vol. 1086 of *CEUR Workshop Proceedings*. 2013. Corpus disponible en `http://komunitatea.elhuyar.org/tweet-norm/`.
5. D. Vilares, M. A. Alonso, and C. Gómez-Rodríguez. On the usefulness of lexical and syntactic processing in polarity classification of Twitter messages. *Journal of the Association for Information Science and Technology (JASIST)*, 2014 (accepted).