Procesamiento del Lenguaje Natural, Revista nº 53, septiembre de 2014, pp 17-24 recibido 14-04-14 revisado 02-06-14 aceptado 10-06-14

Análisis morfosintáctico y clasificación de entidades nombradas en un entorno *Big Data* *

PoS tagging and Named Entitiy Recognition in a Big Data environment

Pablo Gamallo, Juan Carlos Pichel, Marcos Garcia
Centro de Investigación en Tecnoloxías da Información (CITIUS)
{*pablo.gamallo, juancarlos.pichel, marcos.garcia.gonzalez*}*@usc.es*

José Manuel Abuín y **Tomás Fernández-Pena**
Centro de Investigación en Tecnoloxías da Información (CITIUS)
{*tf.pena, josemanuel.abuin*}*@usc.es*

Resumen: Este artículo describe una suite de módulos lingüísticos para el castellano, basado en una arquitectura en *tuberías*, que incluye tareas de análisis morfosintáctico así como de reconocimiento y clasificación de entidades nombradas. Se han aplicado técnicas de paralelización en un entorno *Big Data* para conseguir que la suite de módulos sea más eficiente y escalable y, de este modo, reducir de forma signiticativa los tiempos de cómputo con los que poder abordar problemas a la escala de la Web. Los módulos han sido desarrollados con técnicas básicas para facilitar su integración en entornos distribuidos, con un rendimiento próximo al estado del arte.
Palabras clave: Análisis morfosintáctico, Reconocimiento y clasificación de entidades nombradas, Big Data, Computación Paralela

Abstract: This article describes a suite of linguistic modules for the Spanish language based on a pipeline architecture, which contains tasks for PoS tagging and Named Entity Recognition and Classification (NERC). We have applied run-time parallelization techniques in a Big Data environment in order to make the suite of modules more efficient and scalable, and thereby to reduce computation time in a significant way. Therefore, we can address problems at Web scale. The linguistic modules have been developed using basic NLP techniques in order to easily integrate them in distributed computing environments. The qualitative performance of the modules is close the the state of the art.
Keywords: PoS tagging, Named Entity Recognition, Big Data, Parallel Computing

1 Introducción

En la sociedad digital moderna, se estima que cada día creamos alrededor de 2,5 trillones de bytes de datos (1 exabyte), de tal forma que el 90 % de los datos en todo el mundo han sido creados en los últimos dos años[1]. Así por ejemplo, Twitter genera unos 8 terabytes de datos al día, mientras que Facebook captura unos 100 terabytes[2]. Una de las principales características de esta información es que, en muchos casos, se encuentra sin estructurar, ya que se trata en un porcentaje alto de información textual. Dado el ingente volumen de información textual generado a diario, se hace cada vez más necesario que el procesamiento y análisis lingüístico de esta información se efectúe de manera eficiente y escalable, lo que provoca que las tareas de PLN requieran de soluciones paralelas. Por consiguiente, el uso de la Computación de Altas Prestaciones (HPC) y de su derivación en el paradigma *Big Data*, se hace indispensable para reducir de forma notable los tiempos de cómputo y mejorar la escalabilidad de los sistemas de PLN. En este mismo sentido, cabe reseñar que la filosofía de los enfoques más recientes de la lingüística de corpus se basan en la *Web As Corpus*, línea de investigación donde se postula que con más datos y más

* Este trabajo ha sido subvencionado con cargo a los proyectos HPCPLN - Ref:EM13/041 (Programa Emergentes, Xunta de Galicia), Celtic - Ref:2012-CE138 y Plastic - Ref:2013-CE298 (Programa Feder-Innterconecta)

[1] IBM, Big Data at the Speed of Business: http://www-01.ibm.com/software/data/bigdata/

[2] http://hadoopilluminated.com/-hadoop_illuminated/Big_Data.html

ISSN 1135-5948

texto se obtienen mejores resultados (Kilgarriff, 2007). Y para procesar más corpus a la escala de la Web se requieren soluciones HPC.

En este artículo, nuestro principal objetivo es aplicar técnicas Big Data a un conjunto de tareas lingüísticas integradas en una suite de módulos PLN para el castellano y de código abierto, llamada *CitiusTool*, que incluye la etiquetación y desambiguación morfosintáctica (PoS tagging), así como el reconocimiento y clasificación de entidades nombradas (NERC). De esta manera, conseguimos que la suite de módulos de PLN sea más eficiente y escalable permitiendo reducir de manera signiticativa los tiempos de cómputo y abordar problemas de un tamaño aún mayor.

La arquitectura de la suite de módulos se basa en el paradigma de tuberías (o *pipeline*), y cada módulo lingüístico de la suite es una función escrita en Perl. La ventaja de este enfoque es que cada componente está directamente conectado con los otros a través de los tubos (o *pipes*), de tal modo que no es necesario esperar hasta que finalice un proceso antes de comenzar el siguiente de la *pipeline*. A diferencia de las arquitecturas PLN basadas en el flujo de trabajo (*workflow*), como GATE[3] (Tablan et al., 2013) o UIMA[4], en una tubería cuando un módulo comienza a producir algún tipo de salida esta se transmite como entrada del siguiente módulo sin producir ficheros de datos intermediarios. Una suite de módulos similar a la descrita en este artículo, para castellano e inglés, ha sido implementada en el proyecto opeNER (Agerri, Bermudez, y Rigau, 2014), y que ha dado lugar a IXA pipes (Agerri, Bermudez, y Rigau, 2014), cuyos módulos lingüísticos están basados en Java.

La simplicidad de los módulos PLN que se consideran en este trabajo, así como la clara independencia de las unidades lingüisticas de entrada de dicho módulos (frases, párrafos, textos, etc.), son factores que facilitan su integración en una arquitectura para *Big Data* que usa el modelo de programación *MapReduce*. En concreto, se utilizará la herramienta de código abierto *Hadoop*[5] que implementa dicho modelo. Por otro lado, debemos destacar que a pesar de la gran simplicidad de nuestros módulos lingüísticos, la calidad de sus resultados está muy próxima a la ofrecida por los sistemas considerados estado del arte.

Una vez integrados en una plataforma distribuida, nuestros módulos lingüísticos se podrán utilizar en aplicaciones más complejas y de alto nivel que verán así mejorar su eficiencia. Concretamente, las aplicaciones de ingeniería lingüística que pueden beneficiarse de estos módulos son: traducción automática, recuperación de información, búsqueda de respuestas, sistemas inteligentes de vigilancia tecnológica y, en general, sistemas relacionados con el paradigma de la analítica de textos (*text analytics*), tan en voga con el auge de las redes sociales.

El resto del artículo se organiza del siguiente modo. En la siguiente sección (2), se introduce la arquitectura de los módulos. A continuación, en las secciones 3 y 4, se describen los módulos de análisis morfosintáctico y clasificación de entididades, respectivamente. En ambas secciones se describen también experimentos y evaluaciones cualitativas de los módulos. Seguidamente, la sección 5 se centra en los experimentos realizados con la plataforma Hadoop y finalizamos con las conclusiones y trabajo futuro (sección 6).

2 Arquitectura

CitiusTool[6] es una herramienta lingüística de libre distribución (licencia GPL) concebida para ser fácil de usar, instalar y configurar. La figura 1 muestra la arquitectura en tubería de los diferentes módulos de la suite lingüística. La herramienta consta de varios módulos de análisis básico, un reconocedor de entidades (NER), un analizador morfológico (PoS tagger), que incluye un lematizador, y un clasificador de entidades (NEC). Hasta ahora, hemos adaptado los módulos al castellano, aunque estamos trabajando en su adaptación a otras lenguas peninsulares.

3 Herramientas de análisis

3.1 Análisis básico

Como se puede observar en la figura 1, el sistema consta de varios módulos de procesamiento básico del lenguaje:

- Separador de frases: toma en cuenta los puntos finales, líneas en blanco y un fichero externo de siglas y abreviaturas.

[3]https://gate.ac.uk/

[4]https://uima.apache.org/

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

[6]Disponible para descarga en: http://gramatica.usc.es/pln/tools/CitiusTools.html.

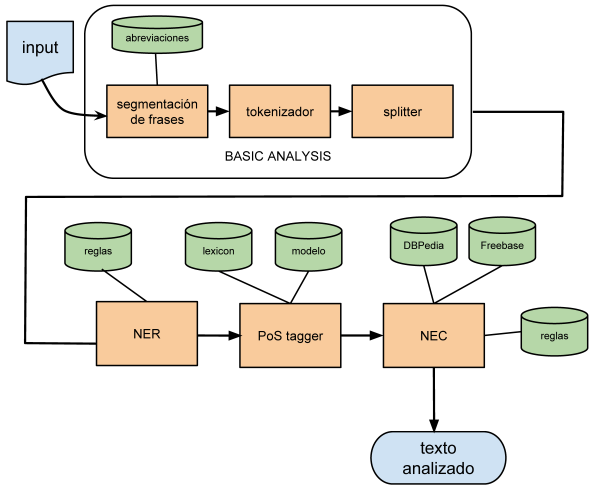


Figura 1: Arquitectura de la suite de módulos lingüísticos CitiusTool

- Tokenizador: separa las frases en tokens.
- Splitter: separa las contracciones en sus correpondientes tokens, por ejemplo, "del" = "de el", "comerlo = "comer lo".

El análisis básico sirve de entrada al Reconocedor de Entidades Nombradas (NER), que aplica un conjunto de reglas para la identificación de nombres propios formados por más de un token. Por ejemplo: *Universidad del País Vasco*, *Real Club Celta de Vigo*, etc. El formato de intercambio entre módulos es similar al de la suite lingüística FreeLing (Padró y Stanilovsky, 2012).

3.2 Análisis morfosintáctico

Hemos desarrollado un desambiguador morfosintáctico o *PoS tagger*, llamado CitiusTagger, utilizando el mismo léxico y etiquetario que FreeLing. Se fundamenta en un clasificador de bigramas bayesiano que asigna la etiqueta más probable a cada token ambiguo tomando en cuenta su contexto a la izquierda y a la derecha. Para desambiguar un token ambiguo, calculamos la probabilidad de cada etiqueta e_i asociada a ese token, dado un conjunto de atributos contextuales $A_1, ...A_n$:

$$P(e_i \mid A_1, ...A_n) = P(e_i) \prod_{i=0}^{N} P(A_i \mid e_i) \qquad (1)$$

El conjunto de atributos que, en experimentos preliminares, nos dieron mejores resultados es el formado por:

- e_{i-1} : la etiqueta que aparece inmediatamente a la izquierda de e_i
- e_{i+1} : la etiqueta que aparece inmediatamente a la derecha de e_i
- (t_i, e_{i-1}) : la copresencia del token ambiguo t_i junto con la etiqueta que aparece inmediatamente a la izquierda de e_i
- (t_i, e_{i+1}) : la copresencia del token ambiguo t_i junto con la etiqueta que aparece inmediatamente a la derecha de e_i

Tomando en cuenta estos cuatro atributos específicos, similares a los utilizados en (Banko y Moore, 2004), el producto de atributos genéricos de la ecuación 1 se puede especificar de esta manera:

$$P(e_{i-1} \mid e_i)P(e_{i+1} \mid e_i)P(t_i, e_{i-1} \mid e_i)P(t_i, e_{i+1} \mid e_i) \qquad (2)$$

Finalmente, se selecciona la etiqueta con el valor máximo de todas las posibles asignadas a un token ambiguo. Optamos por un algoritmo que desambigua los tokens de izquierda a derecha, lo que significa que el contexto a la izquierda de una palabra ambigua es un token ya desambiguado. Solo los contextos a la derecha pueden contenener tokens ambiguos susceptibles de asociarse a varias etiquetas, el conjunto de las cuales forma parte de los atributos contextuales de la etiqueta a desambiguar.

La estrategia bayesiana descrita aquí se encuentra conceptualmente próxima al formalismo de los modelos de Markov ocultos (HMM) (Brants, 2000), formalismo en el que se basa el desambiguador morfosintáctico de la suite lingüística FreeLing (Carreras et al., 2004; Padró y Stanilovsky, 2012), con el que comparamos nuestro sistema. La principal diferencia es que, en nuestro modelo, la desambiguación se hace token a token (como un problema de clasificación), en vez de buscar la mejor secuencia de etiquetas asociada a la frase de entrada mediante programación dinámica (algoritmo Viterbi). El motivo que nos ha llevado a utilizar un simple clasificador bayesiano como desambiguador es su eficiencia computacional.

3.3 Experimentos y evaluación

Para conocer la calidad de los resultados de nuestro sistema de análisis morfosintáctico, CitiusTagger, lo comparamos con otras

dos herramientas, FreeLing y Tree-Tagger (Schimd, 1995), entrenados con el mismo corpus de entrenamiento, el mismo léxico y el mismo conjunto de etiquetas (*tagset*). El corpus de entrenamiento utilizado fue generado con los primeros 300 mil tokens del corpus anotado Ancora (Taulé, Martí, y Recasens, 2008). El léxico es el utilizado por FreeLing, cuyo tagset coincide con el del corpus Ancora. El corpus de test consiste en otros 90 mil tokens extraídos del mismo corpus. El módulo de desambiguación morfosintáctica de FreeLing se basa en Modelos Ocultos de Markov (HMM), a partir de trigramas. El método de análisis de Tree-Tagger, por su parte, se basa en árboles de decisión.

La precisión se calcula dividiendo el número de tokens etiquetados correctamente por cada sistema entre el número total de tokens del corpus de test. La evaluación se llevó a cabo teniendo en cuenta las dos primeras letras de las etiquetas (o tres para algunas categorías), que fueron las únicas utilizadas en el proceso de entrenamiento. La tabla 1 muestra los resultados obtenidos por los tres sistemas. FreeLing alcanza la precisión más alta superando en 0,4 puntos a nuestro sistema, que a su vez, supera en 0,93 a Tree-Tagger. Estos resultados están muy próximos a otros experimentos descritos en (Gamallo y Garcia, 2013) para el portugués, donde FreeLing supera en 1,6 puntos a Tree-Tagger, usando también ambos sistemas el mismo corpus de entrenamiento y el mismo léxico.

Cabe destacar que hemos realizado otra prueba con la versión de Tree-Tagger original para español[7], adaptando las etiquetas de salida a nuestro corpus de test. Los resultados de esta versión son significativamente más bajos, alcanzando apenas el 92,13 % de precisión.

Sistemas	**Precisión (%)**
CitiusTagger	96,45
FreeLing	96,85
Tree-Tagger	95,52

Tabla 1: Precisión de tres sistemas de análisis morfosintáctico con el mismo corpus de entrenamiento y léxico

[7]http://www.cis.uni-muenchen.de/-~schmid/tools/TreeTagger/

4 Clasificación de Entidades (NEC)

Otra de las herramientas de nuestra suite lingüística es un módulo de clasificación de entidades nombradas (CitiusNEC). En la versión actual, este módulo está configurado para clasificar cuatro tipo de entidades: personas, lugares, organizaciones y otras (o misceláneo). El sistema se basa principalmente en consultas a ficheros de recursos extraídos de fuentes enciclopédicas y en un conjunto de reglas de desambiguación semántica. Los ficheros de recursos utilizados por el módulo fueron construidos utilizando como principales fuentes las grandes bases de información enciclopédica FreeBase[8] y DBpedia[9]. Las reglas de desambiguación se aplican a las entidades previamente identificadas por el reconocedor de entidades (NER), con el fin de resolver problemas de entidades ambiguas (a las que se asocian varias clases semánticas) o desconocidas (que no existen en los recursos). Para el caso de entidades conocidas no ambiguas, la clasificación es trivial, pues la clase asignada es la que se encuentra en el recurso enciclopédico correspondiente.

4.1 Reglas de desambiguación

La entrada del sistema NEC es texto etiquetado con entidades previamente identificadas por el sistema NER. Además, se requieren dos tipos de recursos: *gazetteers* y *triggers*. Los primeros son entidades clasificadas que fueron extraídas de FreeBase y DBpedia. Se construyeron tres ficheros de gazetteers (para personas, lugares y organizaciones), dejando fuera la categoría más heterogénea: "misceláneo". Los triggers son sustantivos que pueden subclasificar cada una de las tres categorías utilizadas. Por ejemplo, para la categoría "organización", seleccionamos como triggers, en base al árbol de categorías de la Wikipedia, lemas tales como: *institución, partido, asociación, federación, sindicato, club, entidad, empresa, cooperativa*, etc.

Dada una Entidad Nombrada (EN), el algoritmo de desambiguación procede de esta manera:

Consulta a los gazetteers: Si la EN se encuentra solo en una clase de gazetteers,

[8]http://www.freebase.com/

[9]http://dbpedia.org

entonces se considera que no es ambigua y se le asigna la clase encontrada.

Búsqueda de triggers: Si la EN aparece en varios gazetteers (ambigua) o si es desconocida (no se encuentra en los gazetteers), entonces se busca en el contexto lingüístico la aparición de triggers relevantes. El contexto se define como una ventana de N lemas a la izquierda y a la derecha de la EN a desambiguar, siendo la instanciación $N = 3$ la que mejores resultados devuelve en expermientos preliminares.

Ordenamiento de clases: Si la EN es ambigua y no puede desambiguarse por medio de la búsqueda contextual (etapa anterior), se selecciona la clase más probable (*prior probability*). Calculamos la probabilidad calculando la distribución de los gazetteers en la Wikipedia.

Verificación interna: Si la EN es desconocida y no puede desambiguarse por medio de la búsqueda contextual, entonces se verifica si la primera expresión constituyente de la EN coincide con la primera expresión de una EN en los gazetteers, o si es un nombre común que se encuentra en alguna de las listas de triggers. En caso de que se den varias opciones, se da preferencia a los gazetteers sobre los triggers y, cuando hay ambiguedad, utilizamos el ordenamiento de clases, tal y como se ha descrito arriba.

Else: Si ninguna regla se aplica, la EN se clasifica como "misceláneo".

Cabe destacar que las reglas en sí mismas son independientes de la lengua. Lo que es dependiente de una lengua concreta son los recursos utilizados. En (Gamallo y Garcia, 2011; Garcia, González, y del Río, 2012) se describe un sistema similar para el portugués y gallego, respectivamente, basado en reglas y dependiente de recursos externos.

4.2 Experimentos y evaluación

A continuación, comparamos el módulo CitiusNEC descrito arriba con dos sistemas NEC de aprendizaje supervisado:

- El módulo NEC de FreeLing, el cual obtuvo los mejores resultados en la competición *CoNLL-2002 shared task* (Tjong y Erik, 2002). Este sistema se basa en el algoritmo AdaBoost que consiste en combinar varios clasificadores básicos (Carreras et al., 2002). También utiliza recursos externos (gazetteers y triggers) para definir atributos específicos. El modelo utilizado en los experimentos descritos en esta sección es el que se encuentra en la última versión estable del paquete FreeLing (versión 3.1).
- Apache OpenNLP (Apache Software Foundation, 2014), cuyo módulo NEC permite entrenar modelos con dos algoritmos de aprendizaje: uno basado en redes neuronales (perceptrón) y otro basado en el principio de máxima entropía. Hemos entrenado un modelo para NEC con la siguiente configuración: algoritmo de máxima entropía y $Cutoff = 1$, que fue la que nos proporcionó mejores resultados.

Es importante reseñar que los modelos estadísticos de estos dos sistemas fueron entrenados con el corpus de entrenamiento de *CoNLL-2002 shared task*. En el caso de OpenNLP hemos añadido el corpus de desarrollo de dicha competición. Nuestro sistema, en cambio, no depende de ningún corpus de entrenamiento anotado, ya que se basa en recursos externos (supervisión distante).

Para llevar a cabo la evaluación cualitativa, nos servimos de dos corpus de test: el utilizado en *CoNLL-2002 shared task* para evaluar los sistemas en competición, y otro que llamamos *Hetero*, construido por nosotros a partir de diferentes fuentes: la Wikipedia y noticias de periódicos online (Gamallo y Garcia, 2011). Las tablas 2 y 3 muestran la precisión, *cobertura* y *f-score* obtenidos por los tres sistemas en ambos corpus de test.

Los resultados muestran que los dos sistemas entrenados con el corpus de *CoNLL-2002 shared task*, FreeLing y OpenNLP, consiguen mejores resultados que CitiusNEC cuando se evalúan con el corpus de test de *CoNLL-2002 shared task*, y por lo tanto, de características semejantes al corpus de entrenamiento. La precisión de estos dos sistemas baja significativamente cuando se evalúan con un corpus de naturaleza distinta a la del corpus de entrenamiento, como es el corpus de test *Hetero*. Nuestro módulo, CitiusNEC, mantiene resultados estables independientemente del tipo de corpus utilizado en la evaluación e, in-

Sistemas	Precisión (%)	Cobertura (%)	F-score (%)
CitiusNEC	67,47	66,33	66,89
FreeLing	75,08	76,90	75,98
OpenNLP	**78,96**	**79,09**	**79,02**

Tabla 2: Resultados de tres sistemas NEC utilizando el corpus de test *CoNLL-2002 shared task*

Sistemas	Precisión (%)	Cobertura (%)	F-score (%)
CitiusNEC	**67,47**	65,37	**66,40**
FreeLing	65,67	65,44	65,56
OpenNLP	64,50	**66,84**	65,65

Tabla 3: Resultados de tres sistemas NEC utilizando el corpus de test *Hetero*

cluso, supera ligeramente en f-score a estos dos sistemas con el corpus *Hetero*. Cabe destacar que en el trabajo descrito en (Gamallo y Garcia, 2011), se observó la misma tendencia al evaluar FreeLing y una versión anterior a CitiusNEC para el portugués.

5 *Paralelización en la ejecución de los módulos*

Una tubería de módulos como la descrita en este artículo puede adaptarse para su procesamiento en paralelo mediante el modelo de programación MapReduce, y usando Apache Hadoop. Desde la publicación por parte de Google del modelo de programación MapReduce (Dean y Ghemawat, 2004), surgieron un conjunto de herramientas, muchas de ellas de código abierto, que utilizan este algoritmo distribuido. MapReduce divide el procesamiento de un algoritmo en etapas paralelizables que se ejecutan en muchos nodos (*mappers*), así como en etapas de agregación donde los datos obtenidos en la fase previa son procesados en un único nodo (*reducers*). De esta manera se facilita el almacenamiento y procesamiento del llamado *Big Data*. La herramienta más extendida basada en el modelo MapReduce es Hadoop, desarrollada inicialmente por Yahoo y después publicada como proyecto Apache. Hadoop permite almacenar y procesar de manera distribuida enormes cantidades de datos no estructurados haciendo uso de clusters de computadores de bajo coste (*commodity hardware*), proporcionando al mismo tiempo facilidad de programación, escalabilidad y tolerancia a fallos. De hecho, su modelo de programación es simple y oculta al desarrollador detalles de bajo nivel. Según estimaciones de Shaun Connolly lanzadas en el Hadoop Summit 2012, en el año 2015 alrededor del 50 % de todos los datos mundiales serán almacenados y procesados en clusters basados en Apache Hadoop[10].

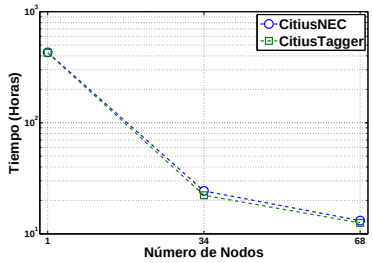
La integración de nuestra suite lingüística en Hadoop permite que los textos de entrada se particionen en subconjuntos, y que cada uno de ellos se procese en un nodo del cluster en paralelo con el resto de nodos. Con el objetivo de evitar cualquier tipo de modificación sobre los módulos PLN originales aprovechamos también un servicio que ofrece Hadoop, llamado "*Hadoop streaming*", que permite el uso de cualquier ejecutable como *mapper* o *reducer*, independientemente del lenguaje de programación en el que esté escrito.

5.1 Datos sobre eficiencia

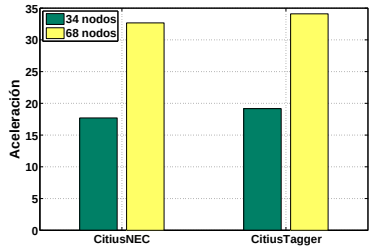
En la figura 2 mostramos los resultados de rendimiento de los módulos CitiusNEC y CitiusTagger una vez integrados en la infraestructura de Hadoop. Los experimentos se han llevado a cabo en un clúster del Centro de Supercomputación de Galicia (CESGA) compuesto por 68 nodos, cada uno de los cuales a su vez consiste en un procesador Intel Xeon E5520. Como texto de entrada de ambos módulos para los tests se ha utilizado la Wikipedia en español, cuyo tamaño en texto plano es de 2,1 GB.

En la figura 2(a) mostramos una comparación entre los tiempos de ejecución de los módulos originales cuya ejecución es secuencial (es decir, usando un único procesador), y los módulos una vez integrados con Hadoop usando en la ejecución paralela 34 y 68 nodos. Los módulos en su ejecución secuencial tardan en torno a unas 425 horas para procesar la Wikipedia en español, lo que supone algo más de 2 semanas. Estos tiempos de ejecución hacen inviable la aplicación de estos

[10] http://www.biganalytics2012.com/resources/Shaun_Connolly-HadoopNowNextBeyond-v10.pdf



(a)



(b)

Figura 2: Rendimiento de los módulos PLN después de su integración con Hadoop: tiempos de ejecución (a) y aceleración (b).

módulos PLN a problemas de gran tamaño como el que abordamos aquí. Sin embargo, cuando usamos la versión paralela con Hadoop los tiempos se reducen de forma notable. Así, por ejemplo, usando 68 nodos, los módulos CitiusNEC y CitiusTagger necesitarían 13 y 12,1 horas respectivamente. Aunque este tiempo sigue siendo elevado debemos tener en cuenta que si usásemos un mayor número de nodos en nuestro clúster Hadoop, el tiempo seguiría escalando y reduciéndose.

Para comprobar la escalabilidad de nuestros módulos PLN mostramos en la figura 2(b) los resultados de aceleración con respecto al caso secuencial. La aceleración se calcula como el cociente entre el tiempo de ejecución secuencial y el tiempo de ejecución paralelo. En el caso ideal, usando un clúster de N procesadores, obtendríamos una aceleración de N. Sin embargo, esta situación es difícilmente alcanzable, puesto que siempre hay tiempos necesarios para la inicialización, además de partes no paralelizables del código (Ley de Amdhal). En nuestro caso obtenemos aceleraciones superiores a 30 usando 68 nodos. Esto indica que los módulos integrados en la infraestructura Hadoop obtienen los resultados 30 veces más rápido que los módulos en su versión secuencial.

5.2 Trabajo relacionado

A diferencia de los algoritmos de minería de datos donde existen herramientas específicas que explotan las capacidades analíticas de Hadoop (p.ej. Apache Mahout para clasificadores, recomendadores y algoritmos de clustering y Apache Giraph para el procesamiento de grafos), no conocemos a día de hoy ninguna herramienta que emplee de forma integrada soluciones de PLN en *Big Data*.

Recientemente, el paradigma MapReduce se ha comenzado a aplicar a algunas tareas de PLN, como por ejemplo la tradución estatística (Ahmad et al., 2011; Dyer, Cordora, y Lin, 2008), la construción de matrices de coocurrencias (Lin, 2008), la minería de textos (Balkir, Foster, y Rzhetsky, 2011), la computación de similaridades semánticas (Pantel et al., 2009), o la adquisición de paráfrasis (Metzel y Hovy, 2011).

6 Conclusiones

Hemos presentado una suite de módulos lingüísticos escritos en Perl y organizados en una simple arquitectura en *tuberías*, cuyo desempeño está próximo al estado del arte. Los datos de entrada pueden particionarse en varios tubos ejecutados en paralelo en un clúster con la ayuda de *Hadoop Streaming*.

Cabe destacar que el módulo de clasificación de entidades nombradas puede extenderse con facilidad a un conjunto mayor de clases semánticas, puesto que no depende de la construcción de un corpus anotado. Al basarse en la técnica de la supervisión distante, solo depende de las fuentes externas enciclopédicas. En estos momentos, estamos extendiendo los recursos para dar cuenta de nuevas categorías semánticas específicas ligadas al dominio de la informática.

Actualmente, hemos desarrollado una versión de los módulos para portugués y estamos también desarrollando nuevas versiones para gallego e inglés, con el propósito de crear una suite multilingüe adaptada para su uso con tecnologías *Big Data*.

Bibliografía

Agerri, R., J. Bermudez, y G. Rigau. 2014. Efficient and easy nlp processing with ixa pipeline. En *Demo Sessions of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2014)*, Gothenburg, Sweden.

Ahmad, R., P. Kumar, B. Rambabu, P. Sajja anda M.K. Sinha, y P. Sangal. 2011. Enhancing throughout of a machine translation system using mapreduce framework: An engineering approach,. En *9th International Conference on Natural Language Processing ICON-2011*, Hyderabad, India.

The Apache Software Foundation, 2014. *Apache OpenNLP*.

Balkir, A.S., I. Foster, y A. Rzhetsky. 2011. A distributed look-up architecture for text mining applications using mapreduce. En *International Conference for High Performance Computing, Networking, Storage and Analysis*.

Banko, Michele y Robert Moore. 2004. Part of speech tagging in context. En *COLING'04, 20th international conference on Computational Linguistics*.

Brants, Throrsten. 2000. Tnt: A statistical part-of-speech tagger. En *6th Conference on Applied Natural Language Processing. ANLP, ACL-2000*.

Carreras, X., I. Chao, L. Padró, y M. Padró. 2004. An Open-Source Suite of Language Analyzers. En *4th International Conference on Language Resources and Evaluation (LREC'04)*, Lisbon, Portugal.

Carreras, X., L. Marquez, L. Padró, y M. Padró. 2002. Named entity extraction using adaboost. En *COLING-02 proceedings of the 6th Conference on Natural Language Learning*.

Dean, J. y S. Ghemawat. 2004. Mapreduce: Simplified data processing on large clusters OSDI-04. En *Sixth Symposium on Operating System Design and Implementation*, San Francisco, CA, EE.UU.

Dyer, C., A. Cordora, y J. Lin. 2008. Fast, easy and cheap: Construction of statistical machine translation model with mapreduce. En *3rd Workshop on Statistical Machine Translation*, Columns, Ohio.

Gamallo, Pablo y Marcos Garcia. 2011. A resource-based method for named entity extraction and classification. *LNCS*, 7026:610–623.

Gamallo, Pablo y Marcos Garcia. 2013. Freeling e treetagger: um estudo comparativo no âmbito do português. En *ProLNat Technical Report, vol. 01, URL: http://gramatica.usc.es/∼gamallo/artigos-web/PROLNAT_Report_01.pdf.*

Garcia, M., I. González, y I. del Río. 2012. Identificação e classificação de entidades mencionadas em galego. *Estudos de Linguística Galega*, 4:13–25.

Kilgarriff, Adam. 2007. Googleology is bad science. *Computational Linguistics*, 31(1):147–151.

Lin, J. 2008. Scalable language processing algorithms for the masses: A case study in computing word co-occurrence matrices with mapreduce. En *2008 Conference on Empirical Methods in Natural Language Processing*, Honolulu, USA.

Metzel, D. y E. Hovy. 2011. Mavuno: a scalable and effective hadoop-based paraphrase acquisition system. En *LDMTA-11, Third Workshop on Large Scale Data Mining: Theory and Applications*.

Padró, Lluís. y Evgeny Stanilovsky. 2012. Freeling 3.0: Towards wider multilinguality. En *Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey.

Pantel, P., E. Crestan, A. Borkovsky, A.M. Popescu, y V. Vyas. 2009. Web-scale distributional similarity and entity set expansion. En *Conference on Empirical Methods in Natural Language Processing*, Singapur.

Schimd, Helmut. 1995. Improvements in part-of-speech tagging with an application to german. En *ACL SIGDAT Workshop*, Dublin, Ireland.

Tablan, V., I. Roberts, H. Cunningham, y K. Bontcheva. 2013. Gatecloud. net: a platform for large-scale, open-source text processing on the cloud. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 371.

Taulé, M., M.A. Martí, y M. Recasens. 2008. Ancora: Multilevel annotated corpora for catalan and spanish. En *The 6th International Conference on Language Resources and Evaluation (LREC).*, Marrakesh, Morocco.

Tjong, Kim Sang y F. Erik. 2002. Introduction ot the CoNLL-2002 shared task: Language independent named entity recognition. En *Conference on Natural Language Learning*.