Un Framework de Ingeniería del Lenguaje para el Pre-procesado Semántico de Textos*

M. V. Rosas[1], M. L. Errecalde[1] y P. Rosso[2]

[1] LIDIC (Research Group). Universidad Nacional de San Luis. San Luis, Argentina.
{mvrosas,merreca}@unsl.edu.ar
[2] Natural Language Engineering Lab. - ELiRF, DSIC, Universidad Politécnica de Valencia. Valencia, España. prosso@dsic.upv.es

Resumen Es común, hoy en día, recibir mucha más información de la que se desea o se puede procesar. Gran parte de dicha información es de tipo textual; razón por la cual, las aplicaciones vinculadas al Procesamiento del Lenguaje Natural (PLN) adquieren, día a día, una mayor relevancia. Distintos trabajos en Ingeniería del Lenguaje, una disciplina que es la intersección entre la Ingeniería de Software y el PLN, han realizado aportes interesantes al tratar de solucionar el problema de la baja tasa de reuso e integración de componentes que en general se puede observar en los desarrollos de PLN. Sin embargo, estos trabajos suelen presentar la falencia de que el costo de entenderlos y usar sus abstracciones es más alto que el costo estimado por el programador en desarrollarlo nuevamente desde cero. En estos casos, un problema adicional es el hecho de que muchas de las arquitecturas introducidas se han concentrado en modelos muy generales y poco flexibles para ciertos problemas particulares de PLN. En este trabajo, se presenta un prototipo de marco de trabajo que sea igualmente flexible como sencillo de aplicar, brindando interfaces y clases abstractas básicas para las tareas involucradas en el pre-procesamiento semántico de documentos. Uno de los beneficios principales del *framework* desarrollado es la reusabilidad de código, generando un ahorro de tiempo importante en la implementación de experimentos, y que incentiva a continuar en esta línea de investigación.

Keywords: Marco de trabajo, Procesamiento del Lenguaje Natural, Ingeniería del Lenguaje

1. Introducción

El exceso de información a la que una persona está expuesta hoy en día, generalmente ocasiona la imposibilidad de identificar, seleccionar y procesar lo que realmente se necesita. En la sociedad de la información y la comunicación en que estamos inmersos, el problema tiende a profundizarse debido a que el conjunto de textos en lenguaje natural aumenta considerablemente. En este

* El trabajo del segundo y tercer autor ha sido soportado por el proyecto MICINN (Plan I+D+i) TEXT-ENTERPRISE 2.0 (TIN2009-13391-C04-03).

contexto, las aplicaciones y técnicas vinculadas al procesamiento y organización automática de documentos (recuperación, categorización, agrupamiento, etc.) juegan un papel importante en nuestros días. En este sentido, el desarrollo de métodos efectivos para lograr mejoras en estas tareas, continúa siendo un tema abierto de investigación. A pesar de que se han obtenido importantes aportes en trabajos pertenecientes al ámbito de PLN [1,3,13] en el aspecto ingenieril se pueden encontrar un número limitado de estudios realizados. En relación a lo anterior, algunas de las falencias en los sistemas de PLN, tales como robustez, productividad (a través del reuso de componentes) y flexibilidad, son los temas de mayor relevancia en la Ingeniería del Lenguaje (IL). Si consideramos, por ejemplo, la baja reutilización de componentes en aplicaciones de PLN se observa que es uno de los problemas más complejos a solucionar. Esto se debe a que la reutilización en este caso presenta una doble limitación, necesitando no sólo intercambiar datos usando el mismo formato e interpretación, sino que también se necesita realizar la comunicación entre componentes que pueden estar escritos en distintos lenguaje e incluso ejecutar en diferentes plataformas. Por todo esto, generalmente cuando se hace mención al reuso de componentes en PLN en realidad se está trabajando en un típico problema de integración. Claramente, éste es un obstáculo que puede ser superado aplicando conceptos básicos de la Ingeniería de Software (IS) como son las Interfaces de Programación de Aplicaciones (*APIs*) o los marcos de trabajo (en inglés, *Frameworks*).

La literatura vinculada a la IL, nos brinda ejemplos interesantes de sistemas, prototipos y propuestas de entornos de integración y herramientas vinculadas a PLN. Entre las arquitecturas existentes orientadas al desarrollo de componentes y su integración para la creación de sistemas PLN, se pueden citar como ejemplos a GATE [4] y TEXTRACT [9]. Éstas y otras propuestas disponibles tienen la particularidad de ser muy completas pero demasiado complejas y poco flexibles en determinadas situaciones.

La hipótesis de trabajo en que se basa el presente trabajo de investigación, es que el diseño y desarrollo de frameworks para tareas específicas y relevantes del PLN, como lo es el pre-procesado para la incorporación de información semántica en la representación de documentos, puede resultar beneficiosa respecto al uso de sistemas de PLN de carácter general. En este contexto, el principal objetivo es realizar un aporte en el área de la IL brindando un prototipo de marco de trabajo flexible y lo suficientemente simple para ser usado en diferentes proyectos en el pre-procesado semántico de documentos para aplicaciones vinculadas a PLN. Dicho proceso de pre-procesamiento fue dividido en un número de tareas más pequeñas implementadas como módulos separados. El análisis sintáctico, el proceso de lematizado y los algoritmos de desambiguación (que realizan el análisis semántico) son algunas de tales tareas. Al ser sencilla su extensibilidad, es posible acoplar nuevos módulos variando las tareas nombradas anteriormente las cuáles conforman el *framework* desarrollado. La efectividad del enfoque propuesto, es analizada a partir de las experiencias logradas en su aplicación a dos problemas de Minería de Textos recientemente abordados.

El resto del trabajo se organiza de la siguiente manera: las Secciones 2 y 3 presentan conceptos básicos relacionados al PLN y trabajos relacionados en IL; la descripción del *framework* y algunas experiencias de su uso se describen en las Secciones 4 y 5. Finalmente, la Sección 6 presenta las conclusiones y los posibles trabajos futuros.

2. Conceptos Introductorios

Considerada generalmente como una rama de la Inteligencia Artificial, el PLN se encarga de facilitar la comunicación hombre-computadora por medio del lenguaje natural. Las aplicaciones y técnicas vinculadas al PLN, tales como recuperación de la información y categorización de textos, entre otras, deben no sólo estructurar y almacenar la gran cantidad de información que se recibe cada día sino también deben ser capaces de procesarla en forma eficiente.

Las aplicaciones de PLN requieren para un adecuado procesamiento del lenguaje natural de un estudio profundo del mismo y para lo cuál se aplican distintos tipos de análisis. El estudio del lenguaje natural supone abarcar básicamente 4 niveles lingüísticos [2], estos son:

1. *Análisis morfológico*: análisis de las palabras para extraer raíces, rasgos flexivos, unidades léxicas compuestas y otros fenómenos.
2. *Análisis sintáctico*: análisis de la estructura sintáctica de la frase mediante una gramática de la lengua en cuestión.
3. *Análisis semántico*: extracción del significado de la frase, y la resolución de ambigüedades léxicas y estructurales.
4. *Análisis pragmático*: análisis del texto más allá de los límites de la frase, por ejemplo, para determinar los antecedentes referenciales de los pronombres.

Además de los nombrados, se pueden incluir otros niveles de conocimiento como por ejemplo, la información fonológica, pero van más allá del esquema básico mínimo necesario. Estas etapas de análisis forman una cadena o un flujo de aplicación. Por ejemplo, para realizar el análisis semántico se pueden necesitar los resultados del análisis sintáctico y a su vez el análisis sintáctico podría utilizar los resultados del análisis morfológico. Cada uno de estos análisis se vale del resultado de los análisis anteriores. Además, es importante destacar que es posible que para realizar correctamente una tarea se necesiten recursos externos como diccionarios, ontologías o gramáticas.

Para cada una de las tareas de análisis existen distintos enfoques de implementación. Debido a que el lenguaje natural es intrínsecamente ambiguo, una de las tareas más importantes del esquema planteado, y en la que nos enfocaremos en este trabajo, es el *análisis semántico* de un texto. El uso de información semántica implica, en este contexto, la incorporación del *significado* de los términos a la representación de los documentos. La determinación de cuál es el significado que corresponde a los distintos términos no es una tarea directa debido a los problemas de *polisemia* y *sinonimia*. Por este motivo, se requieren de métodos de "desambiguación del sentido de las palabras" (*WSD*, por sus siglas

en inglés) para decidir el significado correcto de cada palabra en el documento dependiendo de su contexto.

Muchas investigaciones en el campo del PLN han estudiado diferentes enfoques para resolver la ambigüedad semántica de las palabras. Una manera general y simple de clasificar los distintos métodos utilizados para WSD es diferenciando al grupo de las estrategias que necesitan colecciones de entrenamiento etiquetadas semánticamente, denominado el grupo de los sistemas *supervisados* de aquellos que no necesitan esa anotación para poder funcionar correctamente, los sistemas *no supervisados*. Una forma alternativa de clasificar los sistemas de WSD es basándose en la principal fuente de conocimiento utilizada para establecer los diferentes sentidos [16]. En este sentido, los experimentos de este trabajo fueron realizados a partir del enfoque de WSD denominado *basado en conocimiento* que, como se explica en [10], utiliza diccionarios, tesauros o bases de conocimiento léxicas para la desambiguación y constituye en muchos casos la única alternativa posible cuando no se cuenta con una colección etiquetada previamente. La resolución de la ambigüedad semántica es considerada como una "tarea intermedia" necesaria y esencial para diversas aplicaciones del PLN.

Es importante destacar que, el preprocesado de un texto, que requiere generalmente pasar por algunas o todas las etapas de análisis planteadas anteriormente se realiza sobre una *representación* del documento. Esto es, se aplica un "modelo de indexación" al texto a procesar para transformarlo en una representación formal. Distintos modelos han sido definidos y ampliamente usados, como por ejemplo el "Modelo Bayesiano", el "Modelo Booleano extendido" y el "Modelo de espacio vectorial" (*VSM*, por sus siglas en inglés) [12] entre otros. En este trabajo se aplicó una variante al VSM en la que se incluye información semántica brindada por la etapa de WSD. Se hará referencia a estas representaciones como "conceptos" y "términos+conceptos" [10]. Limitaciones de espacio impiden una descripción detallada de estos enfoques, pero la idea intuitiva es que en el primer caso, sólo se incluye en la representación los significados de los términos del documento, mientras que en el segundo, estos significados son agregados al conjunto de términos originales .

Por otro lado, desde el punto de vista de la IS, se puede ver cada etapa como un módulo diferente. Los módulos forman la cadena o flujo, y la información va pasando por cada uno de ellos, hasta que llega al final de la cadena donde se obtiene el resultado. De esta manera, la propuesta es simple: desarrollar un marco de trabajo para el conjunto de tareas, que fuera flexible y lo suficientemente simple para ser usado en diferentes proyectos de PLN. En este sentido, un marco de trabajo (*framework*) se considera como un diseño reusable para todo o parte del sistema de análisis. El *framework* estará compuesto por "un conjunto de bloques de construcción de software ya desarrollados, que los programadores pueden usar, extender, o adaptar a soluciones específicas" [6].

En un lenguaje más técnico, un *framework* expone una colección de clases individuales que proveen un conjunto de servicios para un dominio particular y que los clientes pueden usar o adaptar [8]. Es equivalente a un librería de clases que han sido diseñadas para un cierto dominio dotadas de una buena docu-

mentación que incluye detalles precisos de la API exportada junto con ejemplos de cómo ser usada en contextos de problemas particulares. Dicha librería puede ser adaptada y/o extendida para solucionar problemas en un dominio específico. El concepto de *framework* se basa en el desarrollo basado en componentes [15].

Mediante la provisión de una infraestructura estándar, a través del *framework*, se decrementa la cantidad de código que el desarrollador tiene que escribir, testear y depurar. Esto también puede bajar los costos de mantenimiento y acelerar la puesta en producción de una aplicación o la puesta en ejecución de experimentos por parte de un investigador en el dominio.

3. Trabajos Relacionados en Ingeniería del Lenguaje

La mayoría de los sistemas de PLN desarrollados hasta el momento tienen arquitecturas similares. Esto es, cada plataforma permite encadenar y combinar los diferentes componentes de análisis de acuerdo a una jerarquía de ejecución. En general, los sistemas permiten crear flujos lineales de módulos, donde las salidas de uno son las entradas del que le sigue. Ejemplos de estos sistemas:

- *GATE*: *General Architecture for Text Engineering*. Arquitectura que permite construir sistemas PLN a partir de componentes previamente desarrollados. Proporciona un núcleo con los módulos básicos de análisis del lenguaje (morfológico y sintáctico) permitiendo el acoplamiento de nuevos componentes. Está implementado en Java por lo que la integración de componentes se realiza considerando a todo recurso como una clase Java [4].
- *ALEP*: *The Advanced Language Engineering Platform*. Proyecto soportado por la *Commission of the European Community* que tiene como objetivo proporcionar a la comunidad de investigadores en PLN de Europa, un entorno de desarrollo abierto, versátil y de uso general. El sistema ofrece un formalismo basado en la unificación y una máquina lingüística con análisis, generación y transferencia de componentes. Implementado mayoritariamente en Prolog [14].
- *NLTK*: *Natural Language Programming Toolkit*. Sistema que provee herramientas básicas para la manipulación de datos y para la realización de tareas PLN. Debido a sus características, es ampliamente usado con un enfoque pedagógico en el ámbito académico, permitiendo construir fácilmente sistemas PLN usando Python [7].
- *TEXTRACT*: *The TALENT System*. Similar a GATE, pero pensado para el procesamiento a gran escala de textos y con un enfoque industrial. Desarrollado en C++, ofrece una serie de componentes (plugins) con las funciones básicas de PLN (tokenizer, PoSTagger, etc). Estos plugins comparten un repositorio, asociado a cada documento, a través del cual se comunican [9].

En la actualidad, los nuevos sistemas de PLN continúan implementando la misma arquitectura pero con mayor flexibilidad y simplicidad para la integración de nuevos módulos de análisis. Éste es uno de los objetivos del "framework" desarrollado: facilitar el acoplamientos de nuevos algoritmos que implementen las diferentes tareas.

4. Descripción del *Framework* Desarrollado

En este trabajo se propone un framework sencillo, orientado al dominio PLN que brinda interfaces y clases abstractas básicas para las tareas involucradas en el pre-procesamiento semántico de los documentos. Los componentes abstractos ofrecidos deben ser implementados por las clases que desean hacer uso de los servicios que brinda el framework. Siguiendo esta convención, dichas clases pueden luego ser manipuladas de forma predecible por los entornos de desarrollo y las aplicaciones.

Un aspecto a tener en cuenta sobre el diseño y creación de un marco de trabajo, es que inclusive el framework más "elegantemente" diseñado nunca va a ser usado a menos que el costo de entenderlo y usar sus abstracciones sea más bajo que el costo estimado por el programador en desarrollarlo nuevamente. La ganancia real se obtiene cuando las clases y los mecanismos son reutilizados a menudo, indicando que otros obtienen beneficios del desarrollo, permitiendo que el desarrollador/investigador se enfoque en la parte particular del problema sin tener que lidiar con cuestiones adyacentes al mismo.

El framework desarrollado tiene un arquitectura simple y modular. La tarea de pre-procesamiento de documentos fue dividida en un número de tareas más pequeñas implementadas como módulos separados que representan las distintas etapas del análisis lingüístico.

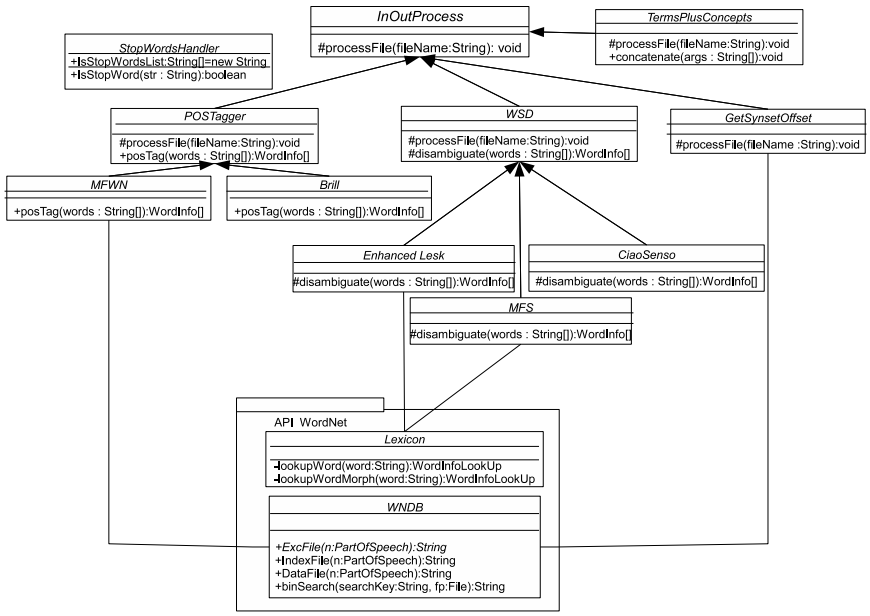

Figura 1. Diagrama de clases

La Figura 1 muestra la arquitectura general simplificada del sistema desarrollado mediante un diagrama de clases. Se muestra un conjunto de clases básicas que realizan distintas tareas de pre-procesado de documentos. Se tienen clases abstractas tales como WSD que luego son implementadas de acuerdo al algoritmo específico utilizado en cada caso. Por ejemplo, para implementar el algoritmo de desambiguación llamado "EnhancedLesk" [1] sólo se debe heredar de la clase abstracta WSD e implementar el método *disambiguate(String[] words)* según la lógica propia de dicho algoritmo sin tener que preocuparse por cuestiones de entrada y salida con la etapa anterior y posterior de este módulo.

Para la API de la ontología Wordnet[3] sólo se detallan las clases principales *Lexicon* y *WNDB* con algunos de sus métodos más importantes. A continuación se precisan los componentes del diagrama de clases obviando los detalles de sus atributos y métodos:

- *InOutProcess*, provee funcionalidad de entrada y salida de archivos para el procesamiento de los diferentes documentos de una colección.
- *StopWordsHandler*, provee métodos para determinar si una palabra pertenece a la lista de palabras de paro (stopword).
- *POSTagger*, representa la clase abstracta que brinda el etiquetado (análisis) sintáctico de cada palabra.
- *WSD*, representa la clase abstracta que realiza la tarea de desambiguación del sentido de las palabras (análisis semántico).
- *GetSynsetOffset*, obtiene el identificador único del synset (offset) determinado por el algoritmo de desambiguación utilizando la ontología WordNet.
- *TermsPlusConcepts*, se obtiene la representación formal del documento, una variante del VSM constituída por los términos más los conceptos obtenidos con el proceso de WSD.

La Figura 2 muestra las diferentes tareas que conforman el proceso. Cada subtarea acepta datos de la etapa previa, aplica una transformación a los datos y los pasa procesados a la próxima etapa. A la figura de esta sección, se le ha anexado al lado de cada tarea la clase abstracta de la que pueden heredar los algoritmos que la llevarían a cabo.

El framework comprende un grupo de clases programadas en lenguaje Java. Los objetos de estas clases pueden ser instanciados en programas del usuario, los métodos pueden ser invocados sobre estos objetos e inclusive se pueden heredar las clases para brindar una mayor funcionalidad a las ya existentes. Un usuario podría crear sus propios módulos (algoritmos) para ejecutar cualquiera de las tareas mencionadas anteriormente, siempre y cuando los módulos adhieran al protocolo especificado por el esquema. Por ejemplo, para implementar el algoritmo de análisis sintáctico (POS-tagging) llamado "Brill tagger" sólo se debe heredar de la clase abstracta *PosTagger* e implementar únicamente el método *posTag(String[] words)* donde debe desarrollarse la lógica propia de dicho algoritmo sin tener que preocuparse por cuestiones de entrada y salida con la etapa anterior y posterior de este módulo.

[3] Base de datos léxica ampliamente utilizada en PLN que define los diferentes sentidos de las palabras y las relaciones entre ellas `http://wordnet.princeton.edu/`.

De esta manera, se podrían incluir diversos algoritmos para los diferentes módulos pudiendo llegar a crear un conjunto de posibilidades que se pueden combinar de formas variadas para la creación y ejecución de experimentos de forma más sencilla sin tener que desarrollar de cero muchas cuestiones que estarían ya solucionadas en el framework.

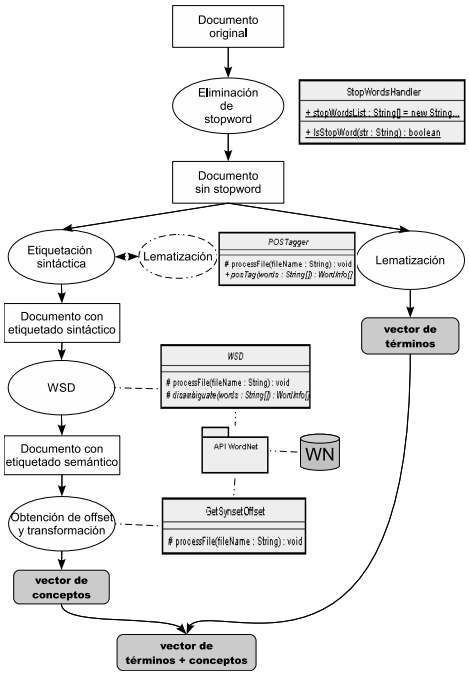

Figura 2. Diagrama de actividades e interacción de clases

5. Experiencias en el uso del *Framework*

El framework propuesto en este trabajo, fue inicialmente utilizado en tareas de *categorización de textos cortos* [10], en estudios que incluyeron 7 colecciones de textos cortos, y el uso de 5 de los métodos más difundidos de categorización, provistos en este caso por el sistema Weka. En este estudio, quedó demostrada la importancia de contar con un framework como el propuesto en este trabajo, facilitando de manera significativa el desarrollo del trabajo experimental. Las razones de estas afirmaciones están fundamentadas en las características particulares que surgen al incorporar información semántica en tareas de categorización de textos. En estos casos, al igual que en otros dominios de PLN que utilizan información semántica, es sabido que la efectividad de incorporar los conceptos en la representación de documentos, depende fuertemente del tipo de codificación

usada en los documentos, del algoritmo de clasificación, del método de WSD y, fundamentalmente, de las características particulares de los documentos de las distintas colecciones. En este sentido, el uso del framework permitió un rápido desarrollo de las distintas instancias experimentales, simplificando la generación de las distintas combinaciones de parámetros y métodos para detectar aquellas que resultaran más efectivas en cada caso.

La experiencia lograda en este primer trabajo con el framework permitió, en un plazo muy breve, analizar si el excelente desempeño en tareas de *clustering de textos cortos* logrado por el algoritmo iterativo ITSA* [5], también se verificaba en aquellos casos en que la representación de los documentos incorporan información semántica. Como resultado de este análisis, en [11] se concluye que los resultados obtenidos son altamente prometedores respecto a la efectividad de este enfoque, obteniéndose resultados muy competitivos en la mayoría de las instancias experimentales consideradas y, en algunos casos, alcanzándose los mejores resultados reportados en la literatura específica en el área.

Es importante notar que, si bien los estudios citados previamente se encuadran claramente en la investigación científica, varias de las colecciones utilizadas en estos casos, corresponden a documentos originales del mundo real obtenidos de repositorios públicos de empresas. Este es el caso, por ejemplo, de las subcolecciones del corpus con cables de noticias *Reuters-21578*, en el que el enfoque semántico utilizado mostró en algunos casos, los mejores valores reportados al día de la fecha. Estos resultados, sumados a la simplicidad en el uso del framework, sugiere que la utilización efectiva del mismo en problemas empresariales concretos del mundo real (como la categorización automática de noticias), es altamente factible y es un aspecto a ser abordado en el futuro.

6. Conclusiones y Trabajos Futuros

El presente trabajo, nos permite concluir que el uso de frameworks para tareas específicas y relevantes del PLN, como lo es el pre-procesado semántico de documentos, puede resultar beneficiosa no sólo para el desarrollo de trabajos de investigación, sino también para aplicaciones futuras en problemas empresariales/industriales que involucran técnicas de PLN. El framework desarrollado brinda un acceso más sencillo a los archivos que se utilizan en los experimentos, un acceso unificado a la base de datos léxica WordNet, algoritmo de lematizado (o *stemming*), de análisis sintáctico (o *pos-tagging*) y de análisis semántico (o *WSD*). En base a esta experiencia se puede concluir que durante el desarrollo de los algoritmos particulares de las distintas tareas del proceso, la parte básica de acceso a recursos externos (como los archivos de WordNet) ya estaría solucionada mediante las clases superiores lo que permite concentrarse en la lógica de solución del problema sin tener que preocuparse por los demás aspectos.

Si bien la combinación de los diferentes algoritmos en las distintas etapas aún se hace manualmente, ésta es más sencilla por lo mencionado previamente. El framework puede ser mejorado creando un "controlador" que, basándose en patrones de diseño (p.ej. *factory* o *abstract factory*) y con configuraciones apro-

piadas, pueda construir anticipadamente un flujo y no tener que ejecutar cada módulo por separado, sino de forma integral. Otra posible extensión, es exportar y acoplar los módulos de desambiguación al framework *GATE*.

Los trabajos futuros incluyen la aplicación del framework a problemas de detección de plagio, e incorporar en el framework el manejo de hiperónimos. Es importante notar que, debido a restricciones de espacio, no es posible profundizar en la descripción del framework en detalles particulares de su implementación y uso. No obstante esto, el lector interesado puede solicitar al primer autor de este trabajo, documentación adicional de sus distintas componentes como asi también del software desarrollado.

Referencias

1. Banerjee S. y Pedersen T.: An Adapted Lesk Algorithm for Word Sense Disambiguation Using Wordnet. En: CICLing, pp 136–145 (2002).
2. Bolshakov I. y Gelbukh A.: Computational Linguistics: Models, Resources, Applications. (Instituto Politécnico Nacional) Mexico (2004).
3. Buscaldi D., Rosso P., y Masulli F.: The upv-unige-ciaosenso wsd System. En: SENSEVAL-3, pp 77–82, Barcelona, España (2004).
4. Cunningham H., Wilks Y., y Gaizauskas R.: Gate-a General Architecture for Text Engineering. En: COLING, pp 1057–1060 (1996).
5. M. Errecalde and D. Ingaramo and P. Rosso: ITSA*: an effective iterative method for short-text clustering tasks. Proc. of IEA-AIE 2010, Springer-Verlag, LNAI 6096, pp. 550-559 (2010).
6. Johnson R.: Components, Frameworks, Patterns. Com. ACM, vol. 40: pp 10–17 (1997).
7. Loper E. y Bird S.: Nltk: The Natural Language Toolkit (2002).
8. Monarchi D., Henderson-Sellers B., Booch G., Jacobson I., Mellor S., Rumbaugh J., y Wirfs-Brock R.: Methodology Standards: Help or Hindrance? OOPS Messenger, vol. 5(cap. 4):pp 54–58 (1994).
9. Neff M., Thomas I., Byrd R., y Boguraev B.: B.k.: The Talent System: Textract Architecture and Data Model. En: Proc. of JNLP-BA. Schmid, H.: Probabilistic, pp 307–326 (2004).
10. Rosas M., Errecalde M. y Rosso P.: Un Análisis Comparativo de Estrategias para la Categorización Semántica de Textos Cortos. Sociedad Española para el Procesamiento del Lenguaje Natural, Revista SEPLN nº 44, pp 11–18 (2010).
11. Rosas M., Ingaramo D., Errecalde M. y Rosso P.: Clustering Iterativo de Textos Cortos con Representaciones basadas en Conceptos. Workshop on Natural Language Processing and Web-based Technologies, Iberamia 2010. En evaluación (2010).
12. Salton G.: The SMART Retrieval System Experiments in Automatic Document Processing. Prentice-Hall, Inc. (1971).
13. Sebastiani F.: Text Categorization. En: L. Rivero J. Doorn, y V. Ferraggine, eds., Encyclopedia of Database Technologies and Applications, pp 683–687 (2005).
14. Simkins N. K.: An Open Architecture for Language Engineering. En: First Language Engineering Convention, Paris, Francia (1994).
15. Sommerville I.: Construction by Configuration: Challenges for Software Engineering Research and Practice. Australian Software Eng. Conf., pp 3–12 (2008).
16. Vázquez S.: Resolución de la Ambigüedad Semántica mediante Métodos Basados en Conocimiento y su Aportación a Tareas de PLN. Ph.D. tesis, España (2009).