Analizadores Multilingües en FreeLing

Lluís Padró
Dept. Lenguajes y Sistemas Informáticos
Centro de Investigación TALP
Universitat Politècnica de Catalunya
padro@lsi.upc.edu

Resumen

FreeLing es una librería de código abierto para el procesamiento multilíngüe automático, que proporciona una amplia gama de servicios de análisis lingüístico para diversos idiomas. FreeLing ofrece a los desarrolladores de aplicaciones de Procesamiento del Lenguaje Natural funciones de análisis y anotación lingüística de textos, con la consiguiente reducción del coste de construcción de dichas aplicaciones. FreeLing es personalizable y ampliable, y está fuertemente orientado a aplicaciones del mundo real en términos de velocidad y robustez. Los desarrolladores pueden utilizar los recursos lingüísticos por defecto (diccionarios, lexicones, gramáticas, etc), ampliarlos, adaptarlos a dominios particulares, o –dado que la librería es de código abierto– desarrollar otros nuevos para idiomas específicos o necesidades especiales de las aplicaciones. Este artículo presenta los principales cambios y mejoras incluidos en la versión 3.0 de FreeLing, y resume algunos proyectos industriales relevantes en los que se ha utilizado.

1. Introducción

FreeLing[1] es una librería de código abierto para el procesamiento multilingüe, que proporciona una amplia gama de funcionalidades de análisis para varios idiomas.

El proyecto FreeLing se inició desde el centro TALP[2] de la UPC para avanzar hacia la disponibilidad general de recursos y herramientas básicos de Procesamiento del Lenguaje Natural (PLN). Esta disponibilidad debería posibilitar avances más rápidos en proyectos de investigación y costes más reducidos en el desarrollo de aplicaciones industriales de PLN.

El proyecto se estructura como una librería que puede ser llamada desde cualquier aplicación de usuario que requiera servicios de análisis del lenguaje. El software se distribuye como código abierto bajo una licencia *GNU General Public License*[3] y bajo licencia dual a empresas que deseen incluirlo en sus productos comerciales.

El planteamiento como un proyecto de código abierto ha sido muy fructífero durante los ocho años de vida de FreeLing (la primera versión fue lanzada en 2003). La versión 2.2 ha sido descargada más de 64.000 veces desde su lanzamiento en septiembre de 2010 por una amplia comunidad de usuarios, la cual ha ampliado el número inicial de tres idiomas (inglés, español y catalán) a nueve, además de la inclusión de la variante diacrónica del español de los siglos XII al XVI (Sánchez-Marco, Boleda, y Padró, 2011). La naturaleza de código abierto del proyecto ha hecho también posible –junto con su arquitectura modular– incorporar el código de otros proyectos similares, como el módulo de desambiguación del sentido de las palabras basado en UKB (Agirre y Soroa, 2009).

La versión actual soporta (a diferentes niveles de completitud) las siguientes lenguas: asturiano, catalán, castellano, galés, gallego, inglés, italiano, portugués, y ruso. Las funcionalidades existentes para cada idioma se resumen en la tabla 1.

La sección 2 describe los principales módulos y servicios de FreeLing. A continuación se describen las principales novedades de la versión 3.0, y la sección 4 resume algunos de los proyectos industriales en los que se ha utilizado la librería. Por último, se esbozan algunas conclusiones y líneas de trabajo futuro.

2. Estructuras de datos y servicios de análisis lingüístico

FreeLing está concebido como una librería sobre la cual se puedan desarrollar potentes aplicaciones de PLN, y orientado a facilitar la integración con las aplicaciones de niveles superiores de los servicios lingüísticos que ofrece.

[1] http://nlp.lsi.upc.edu/freeling
[2] http://www.talp.cat
[3] http://www.gnu.org/copyleft/gpl.html

	as	ca	cy	en	es	gl	it	pt	ru
Tokenization	X	X	X	X	X	X	X	X	X
Sentence splitting	X	X	X	X	X	X	X	X	X
Number detection		X		X	X	X	X	X	X
Date detection		X		X	X	X		X	X
Morphological dictionary	X	X	X	X	X	X	X	X	X
Affix rules	X	X	X	X	X	X	X	X	
Multiword detection	X	X	X	X	X	X	X	X	
Basic named entity detection	X	X	X	X	X	X	X	X	X
B-I-O named entity detection				X	X	X			
Named Entity Classification				X	X				
Quantity detection		X		X	X	X		X	X
PoS tagging	X	X	X	X	X	X	X	X	X
WN sense annotation		X		X	X				
UKB sense disambiguation		X		X	X				
Shallow parsing	X	X		X	X	X		X	
Full/dependency parsing	X	X		X	X	X			
Coreference resolution					X				

Cuadro 1: Servicios de análisis disponibles para cada lengua.

La arquitectura de la librería se basa en un enfoque de dos capas cliente-servidor: una capa básica de servicios de análisis lingüístico (morfológico, morfosintáctico, sintáctico, ...) y una capa de aplicación que, actuando como cliente, realiza las peticiones deseadas a los analizadores y usa su respuesta según la finalidad de la aplicación.

La arquitectura interna de la librería se estructura en dos tipos de objetos: los que almacenan datos lingüísticos con los análisis obtenidos y los que realizan el procesamiento en sí.

2.1. Clases de almacenamiento de datos lingüísticos

Las clases básicas de la librería tienen la finalidad de contener los datos lingüísticos (palabras, etiquetas morfológicas, frases, árboles sintácticos, párrafos, ...) resultado de los análisis realizados. Cualquier aplicación cliente debe usar estas clases para poder proporcionar a los módulos de análisis los datos en el formato oportuno, y para poder recuperar el resultado de los analizadores.

Las clases de datos lingüísticos en la versión actual son las siguientes:

- `analysis`: Una tupla <lema, etiqueta, probabilidad, lista de sentidos>.
- `word`: Forma de una palabra, con una lista de posibles objetos `analysis`.
- `sentence`: Una lista de objetos `word` marcada como una frase completa. Puede contener también un árbol de constituyentes o de dependencias.
- `paragraph`: Una lista de objetos `sentence` marcada como un párrafo independiente.
- `document`: Una lista de objetos `paragraph` que forman un documento completo. Puede contener también información sobre la coreferencia entre las menciones a entidades del documento.

La figura 1 presenta un diagrama UML con las clases de datos lingüísticos.

2.2. Clases de procesamiento

Aparte de las clases para contener datos lingüísticos descritas anteriormente, la librería proporciona también clases para transformarlos, usualmente enriqueciendolos con información adicional. La figura 2 muestra un diagrama UML con las clases de procesamiento que se describen a continuación:

- `lang_ident`: Identificador de idioma. Recibe texto plano y devuelve una lista de pares <idioma,probabilidad>.
- `tokenizer`: Recibe texto plano y devuelve una lista de objetos `word`.
- `splitter`: Recibe una lista de objetos `word` y devuelve una lista de objetos `sentence`.
- `morfo`: Recibe una lista de objetos `sentence` y analiza morfológicamente cada `word` de cada `sentence` de la lista. Esta clase es un meta-analizador que simplemente aplica una cascada de analizadores especializados (detección de números, fechas, locuciones y multipalabras, búsqueda en formario, etc.) cada uno de los

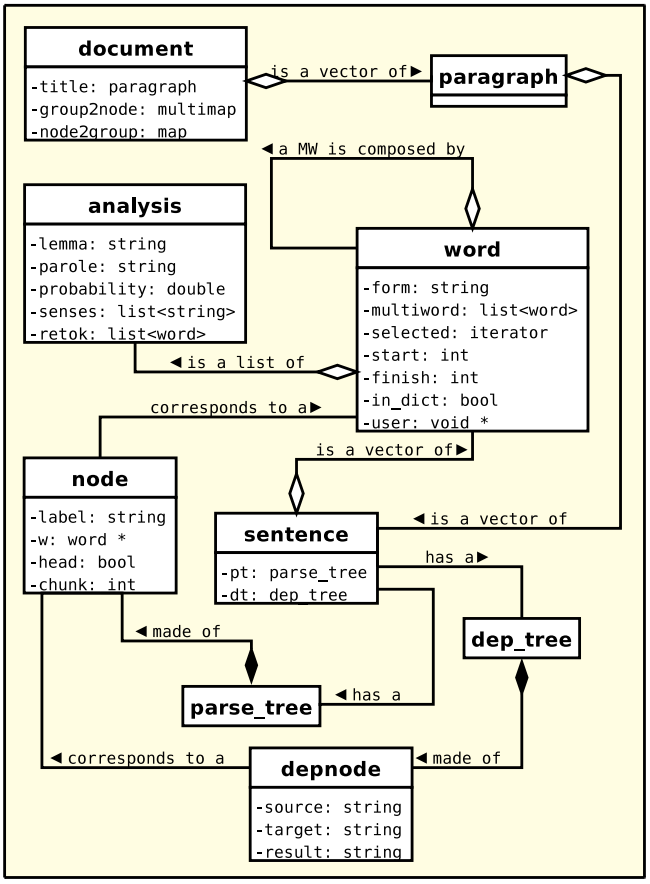

Figura 1: Clases de datos lingüísticos en FreeLing-3.0.

cuales es a su vez una clase de procesamiento que puede ser llamada independientemente si es necesario. Estas clases son:

- `user_map`: Reconocedor de expressiones regulares definidas por el usuario, que permite la asignación directa de pares lema/categoría a palabras que cumplan ciertos patrones.
- `locutions`: Reconocedor de multipalabras.
- `dictionary`: Búsqueda en formario y gestión de afijos.
- `numbers`: Reconocedor de expresiones numéricas.
- `dates`: Reconocedor de expresiones temporales (fechas/horas).
- `quantities`: Reconocedor de expresiones de proporciones, porcentajes, magnitudes físicas y monetarias.
- `punts`: Anotador de signos de puntuación.
- `probabilities`: Anotador de probabilidades léxicas y gestión de palabras desconocidas.
- `ner`: Reconocedor de nombres propios. Se proporcionan dos módulos para esta tarea: Un analizador rápido y simple basado en patrones de mayúsculas (con una precisión alrededor del 90 %), y un reconocedor basado en el sistema ganador de la *shared task* del CoNLL-2002 (Carreras, Màrquez, y Padró, 2002), sensiblemente más lento, pero con una precisión superior al 94 %.

- `tagger`: Recibe una lista de objetos `sentence` y desambigua la categoría morfosintáctica de cada palabra en las frases de la lista. Si el análisis seleccionado incorpora información de retokenización (p.e. *del* → *de+el*, *dárselo* → *dar+se+lo*) la palabra puede separarse en varias. FreeLing ofrece dos *taggers* con una precisión del estado del arte (97 %-98 %): Uno basado en modelos ocultos de markov, según se describe en (Brants, 2000) y otro basado en *relaxation labelling* (Padró, 1998) que permite la combinación de información estadística con reglas manuales.

- NE `classifier`: Recibe una lista de objetos `sentence` y clasifica cada `word` etiquetada como nombre propio que aparezca en las frases dadas. Este módulo está basado en el sistema ganador de la *shared task* del CoNLL-2002 (Carreras, Màrquez, y Padró, 2002).

- `sense annotator`: Recibe una lista de `sentence` y añade information sobre los sentidos posibles (segun WordNet) a los objetos `analysis` de cada palabra.

- `word sense disambiguator`: Recibe una lista de objetos `sentence` y ordena por relevancia en el contexto los posibles sentidos de cada palabra. El código de este módulo se incluye directamente del del proyecto del desambiguador UKB (Agirre y Soroa, 2009).

- `chunk parser`: Recibe una lista de `sentence` y enriquece cada una con un árbol de análisis. Este módulo consiste en un *chart parser*, y es una reimplementación y extensión de (Atserias y Rodríguez, 1998).

- `dependency parser`: Recibe una lista de `sentence` analizadas sintácticamente y las enriquece con un árbol de dependencias. Este módulo usa un conjunto de reglas escritas manualmente que operan en tres etapas: primero completan el árbol sintáctico superficial construido por el *chart parser*, a continuación transforman el árbol de constituyentes a dependencias, y finalmente etiquetan la función de cada dependencia. Este módulo es una extensión del que se describe en (Atserias, Comelles, y Mayor, 2005).

- `coreference solver`: Recibe un documento formado por objetos `sentence` analizados sintácticamente y lo enriquece con información de coreferencia. Este módulo se basa en el sistema propuesto por (Soon, Ng, y Lim, 2001).

3. Novedades en FreeLing 3.0

La versión 3.0 presenta algunos cambios importantes que tienen como objetivo hacer que la herramienta más flexible, usable, y fácil de instalar. Estos cambios pueden agruparse en tres grandes clases: los cambios relacionados con la ampliación del soporte al multilingüismo, los cambios en los componentes de la librería basados en aprendizaje automático, y cambios relacionados con aspectos de ingeniería.

3.1. Ampliación del soporte al multilingüismo

La primera contribución relevante que amplia la cobertura de FreeLing con respecto a la cantidad y variedad de idiomas que puede procesar es el desarrollo de datos lingüísticos para el analizador y el desambiguador morfológico del español de los siglos XII al XVI (Sánchez-Marco, Boleda, y Padró, 2011). Este trabajo utiliza los datos por defecto para el español moderno, más las pertinentes adaptaciones y extensiones para procesar las variaciones ortográficas propias del español antiguo. Además, se ha desarrollado un corpus de entrenamiento del etiquetador, y se ha usado la herramienta resultante en un estudio lingüístico sobre la evolución del uso del verbo *haber* (Sánchez-Marco y Evert, 2011; Sánchez-Marco, 2012).

Otra modificación en la versión 3.0 de FreeLing es el soporte a la codificación de caracteres en Unicode (UTF-8). Este es un cambio importante, y una de las principales razones para el cambio de número de versión.

Las versiones anteriores de FreeLing soportaban varios idiomas, pero el desarrollador de los datos lingüísticos de cada lengua debía decidir que codificación usar, y cuidar de la consistencia de la codificación entre los datos de diversos módulos (diccionario, gramáticas, lexicones semánticos, etc.) o sus reglas o ficheros de configuración (p.e. la escritura de expresiones regulares para el tokenizador). Adicionalmente, dado que cada idioma podía utilizar una codificación diferente, no era fácil integrar un identificador de idioma capaz de manejar textos en diferentes alfabetos.

Con el soporte de las codificaciones UTF-8, la misma aplicación puede manipular textos en diferentes idiomas y alfabetos. Esta extensión ha permitido diversas mejoras funcionales:

- Un nuevo módulo para la identificación de lenguaje basado en (Padró y Padró, 2004) se ha integrado en la librería.
- Posibilidad de cambiar la configuración regional (*locale*) de la aplicación para que coincida con la del idioma del texto procesado, incluso si es distinta de la predeterminada en el sistema local.
- Las expresiones regulares utilizadas, ya sea en archivos de configuración o cableadas en el código son más expresivas y fáciles, ya que se permiten extensiones POSIX. Por ejemplo, la expresión regular usada por el tokenizador para reconocer una palabra formada por caracteres alfabéticos en español solía ser `[A-Za-záéíïóúüñÁÉÍÏÓÚÜÑ]+`, mientras que en la nueva versión se puede escribir como `[[:alpha:]]+`, que no sólo es más simple y simplifica su mantenimiento, sino que es independiente del idioma, ya que esta misma expresión puede utilizarse para reconocer palabras de caracteres alfabeticos en cualquier idioma y/o alfabeto simplemente cambiando la localización activa de la aplicación.

El uso de la codificación UTF8 deja via libre a los desarrolladores interesados en la adición de soporte para idiomas con alfabetos no latinos. Es el caso de los desarrolladores del ruso, que han conseguido un analizador morfológico muy completo y un etiquetador competitivo que no habría sido posible en las versiones anteriores. Además, no sólo se han desarrollado los datos lingüísticos para el ruso, sino también código fuente para los módulos dependientes del idioma, como los reconocedores de números o fechas.

3.2. Mejora de los módulos basados en aprendizaje automático

Otro cambio importante en la arquitectura FreeLing 3.0 es la organización y contenido de los módulos de aprendizaje automático: el motor de extracción de características por un lado, y los algoritmos de aprendizaje/clasificación por el otro.

En versiones anteriores, las funciones de aprendizaje automático eran proporcionadas por dos librerías externas a FreeLing: *Omlet&Fries*[4]. En la versión 3.0, el código de estas librerías está incluido en el paquete FreeLing, lo que ofrece una organización de código más clara y una reducción en el número de dependencias que simplifica el proceso de instalación.

[4] `http://nlp.lsi.upc.edu/omlet+fries`

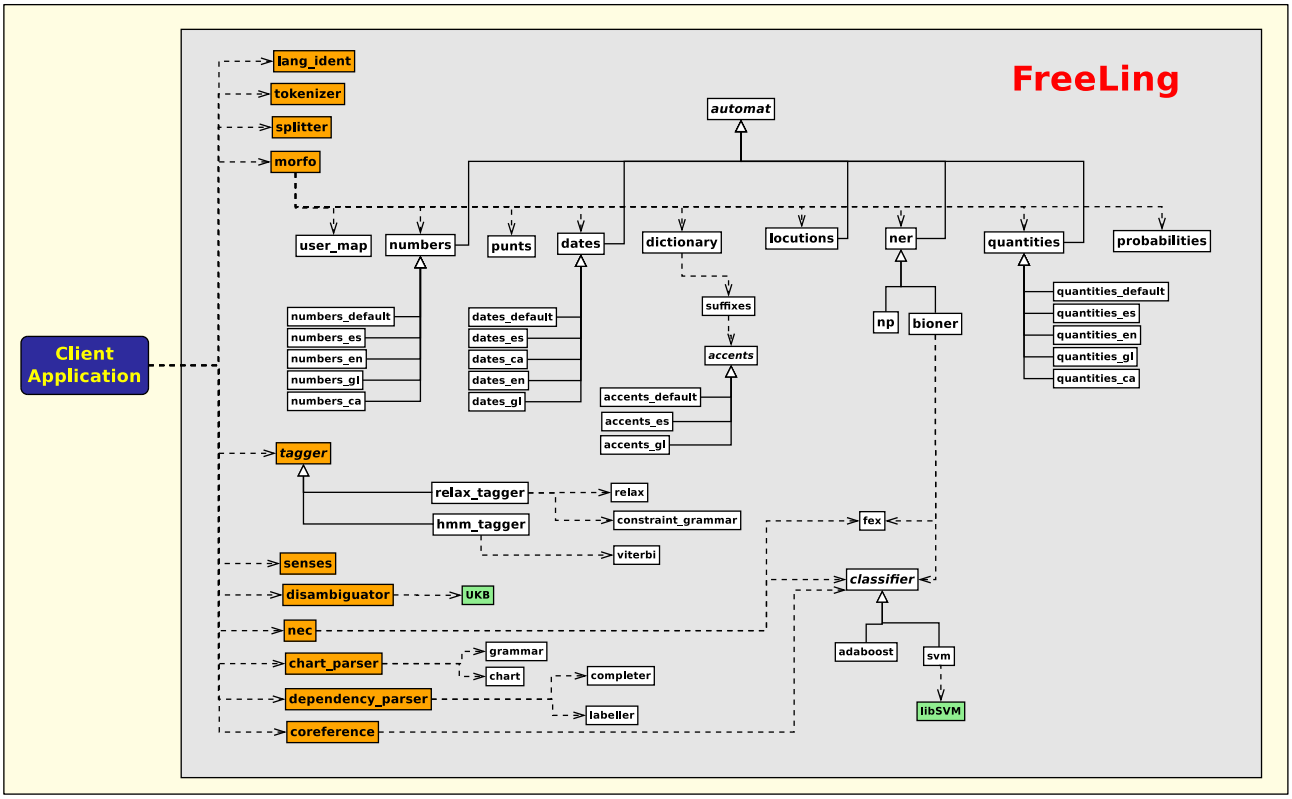

Figura 2: Clases de procesamiento en FreeLing-3.0.

El módulo de extracción de características ha sido completamente reescrito, proporcionando un formalismo de reglas de extracción más claro, y una API más flexible para aquellos desarrolladores que deseen incluir sus propias funciones de extracción de rasgos. Además, el repertorio de algoritmos de aprendizaje disponibles se ha ampliado con *Support Vector Machines* (SVM) gracias al proyecto de código abierto libSVM (Chang y Lin, 2011). El código de libSVM se ha integrado en FreeLing bajo un *wrapper* común con los clasificadores existentes. Esta estrecha integración tiene la ventaja adicional de evitar añadir un nuevo elemento a la lista de dependencias necesarias para construir e instalar FreeLing.

Por último, los modelos para reconocimiento y clasificación de entidades nombradas han sido entrenados con la nueva arquitectura. Módulos basados en aprendizaje automático para el reconocimiento y clasificación de NE se encuentran disponibles para el español, inglés, gallego y portugués. Los dos primeros ofrecen modelos AdaBoost y SVM, mientras que sólo modelos AdaBoost están disponibles para los últimos.

3.3. Modificaciones técnicas

El tercer tipo de cambios en FreeLing son cuestiones técnicas relacionadas con la organización de las dependencias externas, la migración a plataformas distintas de Linux, y el uso de FreeLing en modo servidor.

3.3.1. Dependencias externas

Un aspecto importante de estas modificaciones de ingeniería es la gestión de las dependencias de librerías externas requeridas por FreeLing. Esta es una razón principal –junto con el cambio a codificación Unicode descrito anteriormente– para la actualización del número de versión.

Se ha realizado un esfuerzo importante para reducir y simplificar la lista de dependencias, a fin de facilitar la construcción y instalación de la librería, así como simplificar su uso en productos comerciales bajo licencia dual.

Las dependencias de las versiones anteriores eran:

- BerkeleyDB - Acceso rápido a los archivos de diccionario en disco.
- PCRE - Gestión de expresiones regulares.
- libcfg+ - Gestión opciones de configuración para el programa principal `analyzer`.
- Omlet&Fries - Módulos de Aprendizaje Automático (véase más arriba).

FreeLing 3.0 ya no requiere BerkeleyDB: Los diccionarios se cargan en RAM ya sea en *prefix-trees* o en estructuras *map* de la STL. El rendimiento temporal es aproximadamente el

mismo, y el aumento en el consumo de memoria no supone ningún problema para una máquina moderna. Por otra parte, la mayor simplicidad en la instalación (menos dependencias, no necesidad de indexar los diccionarios durante la instalación), y en la gestión de los diccionarios (no es necesario reindexar después de modificar un diccionario) compensan ampliamente el ligero aumento en el tiempo de inicialización que este cambio supone.

La nueva versión FreeLing tampoco usa ya PCRE ni `libcfg+`: Tanto las funcionalidades relacionadas con las expresiones regulares como las relativas a la gestión de opciones de configuración se han transferido a las librerías *boost*[5]. Esto supone dos ventajas: las dependencias se unifican bajo un solo proveedor, y la instalación es mucho más simple ya que `liboost` es parte de todas las distribuciones Linux.

3.3.2. Compilación nativa en MS-Windows

Utilizar FreeLing bajo MS-Windows solía ser una empresa difícil. Era necesario usar emuladores o compiladores cruzados como MinGW[6] o Cygwin[7], y los resultados obtenidos no siempre eran fácilmente integrables en una aplicación de MS-Windows.

Todo el código C++ en la versión 3.0 se ha adaptado para ser compilado por MS-Visual C++, y se proporcionan los archivos de proyecto para compilar la librería bajo dicho entorno, lo que simplifica enormemente la construcción y el uso de FreeLing en MS-Windows, ya que los binarios obtenidos son nativos en dicho sistema.

3.3.3. Mejoras en el modo servidor

Por último, una mejora técnica de menor importancia es la capacidad multicliente que ofrece el programa de demostración `analyzer` en la nueva versión:

El modo de servidor de las versiones anteriores estaba concebida solamente como un medio para evitar la repetición de la inicialización de los módulos en el caso que se quisiera procesar un gran número de archivos pequeños. Por ello, todas las peticiones eran atendidas de forma secuencial por el mismo servidor, resultando poco adecuado o lento en el caso que se quisiera atender a muchos clientes simultáneamente.

En la nueva versión, el código sigue una arquitectura estándar de un servidor Linux: Un proceso *dispatcher* espera solicitudes de los clientes a través de un *socket*. Cuando se establece una nueva conexión con un cliente, el *dispatcher* crea un nuevo proceso *worker* que se hará cargo del cliente, mientras el *dispatcher* regresa de nuevo a esperar peticiones entrantes.

Esto hace posible el uso de FreeLing en procesamiento paralelo (por ejemplo, para procesar grandes cantidades de texto en una máquina multiprocesador, o para usarlo como un servidor en una aplicación web multiusuario) que no era posible en las versiones anteriores.

Sin embargo, el nuevo servidor no limita el número máximo de clientes conectados, ni tiene una cola de espera para las peticiones. Así pues, las aplicaciones con un cantidad potencialmente masiva de los clientes deben adaptar el código de servidor para manejar con seguridad una cola de solicitudes pendientes.

3.4. Otras mejoras

Otras novedades destacables incluidas en FreeLing 3.0 son las sigüientes:

- *UserMap*: Se trata de un nuevo módulo que permite al usuario definir una serie de expresiones regulares y asignar a cada una de ellas un conjunto de pares <lema,etiqueta> que se asignarán a las palabras que cumplan dicho patrón. El objetivo de este módulo es facilitar al desarrollador de aplicaciones el tratamiento específico de casos no cubiertos por otros módulos en FreeLing. Por ejemplo, una aplicación de procesamiento de *Twitter* podría requerir la anotación como nombres propios de las palabras con el patrón `@nombre`. En lugar del costoso trabajo de modificar o reentrenar el detector de nombres propios, ahora se puede simplemente añadir una regla:
  ```
  @[a-z][a-z0-9]* $$ NP00000
  ```
  que reconoce dichas palabras, asignándoles su propia forma como lema y `NP00000` como etiqueta.
- Trigramas `<Forbidden>`: El tagger basado en HMM efectúa suavizado de las probabilidades de transición no observadas, reservando cierta masa de probabilidad para casos lingüísticamente imposibles (como p.e. un determinante seguido de un verbo finito, o un *haber* auxiliar seguido de algo que no sea un participio). Estos casos se pueden explicitar en el fichero de configuración del tagger, evitando su suavizado y forzando a que tengan probabilidad cero, reduciendo así la tasa de error de la desambiguación.
- *Phonetics*: Otro servicio nuevo en FreeLing es la codificación fonética del sonido de una palabra. Este módulo usa un fichero de reglas de

[5] `http://www.boost.org`
[6] `http://www.mingw.org`
[7] `http://www.cygwin.com`

transcripción que traducen el texto a su codificación en el estándar SAMPA[8]. También es capaz de usar un diccionario fonético de transcripciones de palabras completas para las excepciones a las reglas (o para idiomas, como el inglés, con una fonética poco regular).

4. FreeLing en proyectos industriales

La versión de desarrollo de FreeLing 3.0 está disponible en el SVN del proyecto, y ya ha sido utilizada en varios proyectos industriales, de los cuales resumimos brevemente los más relevantes:

- Ruby Reader: Aplicación para el iPhone que ayuda a los hablantes de japonés a comprender textos en inglés. Desarrollado por CA-Mobile (`http://www.camobile.com`).
- Vi-Clone: Impresionantes asistentes virtuales para páginas web corporativas. Algunos componentes de FreeLing se están integrando en el sistema de diálogo. Vi-Clone está financiando el desarrollo del módulo de corrección ortográfica que permitirá a FreeLing procesar frases del usuario escritas en variantes no estándar. `http://www.vi-clone.com`.
- TextToSign: Traductor de texto en español al lenguaje de señas, que utiliza FreeLing para el procesamiento de texto. `http://www.textosign.es`.
- Dixio: diccionario inteligente capaz de ayudar al lector de un texto, ofreciendo definiciones contextualizadas. Desarrollado por Semantix (`http://www.semantix.com`).
- Aport News: Portal de noticias que usa FreeLing como un preprocesador para enriquecer texto en ruso. El resultado de la anotación se utiliza en la clasificación y agrupación de las noticias. `http://news.aport.ru`.

5. Conclusiones y trabajo futuro

Hemos presentado las principales mejoras y cambios realizados en la versión 3.0 de FreeLing, y algunos proyectos industriales en los que se ha utilizado.

Gracias a estos cambios, y a la activa comunidad en torno a este proyecto, esperamos seguir ampliando el número de idiomas soportados, ampliando las funcionalidades proporcionadas, y mejorando la usabilidad de estos analizadores en aplicaciones industriales del PLN.

[8] `http://www.phon.ucl.ac.uk/home/sampa`

Una de las líneas de trabajo que más interés despierta es la inclusión de un módulo de corrección ortográfica que –como parte de una cadena de análisis robusto– constituya una piedra angular para el desarrollo de aplicaciones orientadas a textos no estándar como chats de internet, foros, microblogs, etc.

Agradecimientos

Este trabajo ha sido parcialmente financiado por el Gobierno Español a través de los proyectos KNOW-2 (TIN2009-14715-C04-03/04) y OpenMT-2 (TIN2009-14675-C03-01), así como por la Unión Europea a través del proyecto FAUST (FP7-ICT-2009-4). Agradecemos también a ViClone (`www.viclone.com`) por financiar parte del desarrollo de FreeLing.

Bibliografía

Agirre, Eneko y Aitor Soroa. 2009. Personalizing pagerank for word sense disambiguation. En *Proceedings of the 12th conference of the European chapter of the Association for Computational Linguistics (EACL-2009)*, Athens, Greece.

Atserias, Jordi, Elisabet Comelles, y Aingeru Mayor. 2005. Txala un analizador libre de dependencias para el castellano. *Procesamiento del Lenguaje Natural*, (35):455–456, September.

Atserias, Jordi y Horacio Rodríguez. 1998. Tacat: Tagged corpus analizer tool. Technical report lsi-98-2-t, Departament de LSI. Universitat Politècnica de Catalunya.

Brants, Thorsten. 2000. Tnt - a statistical part-of-speech tagger. En *Proceedings of the 6th Conference on Applied Natural Language Processing, ANLP*. ACL.

Carreras, Xavier, Lluís Màrquez, y Lluís Padró. 2002. Named entity extraction using adaboost. En *Proceedings of CoNLL Shared Task*, páginas 167–170, Taipei, Taiwan.

Chang, C.C. y C.J. Lin. 2011. Libsvm : a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):1–27, April.

Padró, Lluís. 1998. *A Hybrid Environment for Syntax–Semantic Tagging*. Ph.D. tesis, Dep. Llenguatges i Sistemes Informàtics. Universitat Politècnica de Catalunya, February. http://www.lsi.upc.es/˜padro.

Padró, Muntsa y Lluís Padró. 2004. Comparing methods for language identification. *Proce-*

samiento del Lenguaje Natural, (33):155–162, September.

Soon, W.M., H. T. Ng, y D.C.Y. Lim. 2001. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 27(4):521–544.

Sánchez-Marco, Cristina. 2012. *Tracing the development of Spanish participial constructions: An empirical study of language change*. Ph.D. tesis, Universitat Pompeu Fabra, Barcelona, Spain. (forthcoming).

Sánchez-Marco, Cristina, Gemma Boleda, y Lluís Padró. 2011. Extending the tool, or how to annotate historical language varieties. En *Proceedings of the 5th ACL-HLT Workshop on Language Technology for Cultural Heritage, Social Sciences, and Humanities*, páginas 1–9, Portland, OR, USA, June. Association for Computational Linguistics.

Sánchez-Marco, Cristina y Stefan Evert. 2011. Measuring semantic change: The case of spanish participial constructions. En *Proceedings of 4th Conference on Quantitative Investigations in Theoretical Linguistics (QITL-4)*, Berlin, Germany, March.