Etiquetación Robusta del Lenguaje Natural: Preprocesamiento y Segmentación*

Jorge Graña Gil, Fco. Mario Barcala Rodríguez y **Jesús Vilares Ferro**
Departamento de Computación
Universidad de A Coruña
España
{grana,barcala,jvilares}@dc.fi.udc.es

Resumen Una de las tareas previas más importantes para la etiquetación robusta del lenguaje natural es la correcta segmentación o preprocesamiento de los textos. Esta fase, que puede involucrar a procesos mucho más complejos que la simple identificación de las diferentes frases del texto y de cada uno de sus componentes individuales, es a menudo obviada en muchos de los desarrollos actuales.

A pesar de esto, se trata de una tarea de enorme importancia práctica y abordarla con pleno rigor científico, sin caer repetidamente en el análisis de la casuística particular de cada fenómeno detectado, es una labor que resulta especialmente compleja.

En este trabajo hemos desarollado un esquema de preprocesamiento orientado a la desambiguación y etiquetación robusta del gallego. No obstante, se trata de una propuesta de arquitectura general que puede ser aplicada a otros idiomas, como por ejemplo el español, con modificaciones muy ligeras.

1 *Introducción*

Los diferentes tipos de etiquetadores que existen actualmente asumen que el texto de entrada aparece ya correctamente segmentado, es decir, dividido de manera adecuada en *tokens* o unidades de información de alto nivel de significado, que identifican perfectamente cada uno de los componentes de dicho texto. Esta hipótesis de trabajo no es en absoluto realista debido a la naturaleza heterogénea tanto de los textos de aplicación como de las fuentes donde se originan.

Así pues, algunas lenguas, como el gallego [1] o el español, presentan fenómenos que es necesario tratar antes de realizar la etiquetación. Entre otras tareas, el proceso de segmentación se encarga de identificar unidades de información tales como las frases o las propias palabras. Esta operación puede ser más compleja de lo que parece *a priori*. Por ejemplo, la identificación de las frases se suele realizar considerando ciertas marcas de puntuación. Sin embargo, un simple punto puede ser indicativo de fin de frase, pero podría corresponder también al caracter final de una abreviatura.

En el caso de las palabras, la problemática se centra en que el concepto ortográfico de palabra no siempre coincide con el concepto lingüístico. Se presentan entonces dos opciones:

1. Las aproximaciones más sencillas consideran igualmente las palabras ortográficas y amplían las etiquetas para representar aquellos fenómenos que sean relevantes. Por ejemplo, la palabra `reconocerse` podría etiquetarse como `V000f0PE1`[1] aun cuando está formada por un verbo y un pronombre enclítico, y las palabras de la locución `a pesar de` se etiquetarían respectivamente como `C31`, `C32` y `C33` aun cuando constituyen un único término. Sin embargo, en idiomas como el gallego, este planteamiento no es viable, ya que su gran complejidad morfológica produciría un creci-

* Este trabajo ha sido parcialmente financiado por la Unión Europea (bajo el proyecto FEDER 1FD97-0047-C04-02), por el Ministerio de Educación y Ciencia (bajo el proyecto TIC2000-0370-C02-01), y por la *Xunta de Galicia* (bajo el proyecto PGIDT99XI10502B).

[1] Las etiquetas que aparecen en este trabajo pertenecen a los *tag sets* utilizados en los proyectos Galena (Generador de Analizadores para LEnguajes NAturales) y Corga (*COrpus de Referencia do Galego Actual*). El apéndice A incluye la descripción de cada una de las etiquetas utilizadas. Más información sobre estos proyectos está disponible en `http://coleweb.dc.fi.udc.es`.

miento excesivo del juego de etiquetas.

2. La solución pasa entonces por no ampliar el juego de etiquetas básico. Como ventajas, la complejidad del proceso de etiquetación no se verá afectada por un número elevado de etiquetas, y la información relativa a cada término lingüístico se puede expresar de manera más precisa. Por ejemplo, a lo que antes era un simple pronombre enclítico se le pueden atribuir ahora valores de persona, número, caso, etc. Como desventaja, se complican las labores del preprocesador, que no sólo se verá obligado a identificar las palabras ortográficas, sino que unas veces tendrá que partir una palabra en varias, y otras veces tendrá que juntar varias palabras en una sola.

   Las mayores dificultades surgen cuando esta segmentación es ambigua. Por ejemplo, la expresión `sin embargo` se etiquetará normalmente de manera conjunta como una conjunción, pero en algún otro contexto podría ser una secuencia formada por una preposición y un sustantivo. De igual forma, la palabra `ténselo` puede ser una forma del verbo `tener` con dos pronombres enclíticos, o bien una forma del verbo `tensar` con un solo pronombre. Este fenómeno es muy común en gallego, no sólo con los pronombres enclíticos, sino también con algunas contracciones. Por ejemplo, la palabra `polo` puede ser un sustantivo (en español, pollo), o bien la contracción de la preposición `por` (por) y del artículo `o` (el), o incluso la forma verbal `pos` (pones) con el pronombre enclítico `o` (lo).

En este trabajo hemos adoptado la segunda opción, es decir, la de separar y unir (separar, por ejemplo, el verbo de sus pronombres, y unir, por ejemplo, los diferentes constituyentes de una locución). En cualquier caso, la primera opción, la de trabajar al nivel de la palabra ortográfica, necesitaría, después de la fase de etiquetación, una fase de postprocesamiento, cuando se desee identificar los diferentes componentes sintácticos del texto. Dicha fase de postprocesamiento realizaría las labores análogas a las que involucra nuestro preprocesador.

En nuestro caso, se ha desarrollado un preprocesador que será útil como paso previo a la etiquetación, y teniendo en cuenta que nos centraremos en el preprocesado de textos planos, es decir, se tratarán directamente los fenómenos lingüísticos anteriormente mencionados (contracciones, pronombres enclíticos, locuciones, etc.). Si bien es cierto que existen otros formatos de documentos que presentan otras problemáticas a resolver (eliminación de códigos HTML, XML, etc.), una vez resueltas, dan lugar a los mismos fenómenos que se afrontan aquí, ya que éstos son inherentes a la escritura del idioma.

El objetivo final del presente artículo es desarrollar un preprocesador modular, con algoritmos generales, de manera que pueda ser utilizado para diferentes idiomas, pero obteniendo un comportamiento mucho más fino mediante la introducción en el sistema de información lingüística relativa a una lengua en particular. Por lo tanto, resulta también de especial importancia la definición del tipo de información linguística que va a resultar útil, permitiendo su integración en el sistema en los casos en los que ésta esté disponible.

Además de esto, y como segundo objetivo, no hay que olvidar que nuestro preprocesador está especialmente diseñado como fase previa a la etiquetación de textos, por lo que también va a realizar tareas de pre-etiquetación. La idea subyacente consiste en que en un proceso de etiquetación/desambiguación completo, sea el módulo que más información tiene sobre algún fenómeno el que desambigüe dicho fenómeno.

2 Estructura

Esta sección describe los diferentes módulos que incorpora el preprocesador presentado. Dichos módulos son los que se muestran en la figura 1.

2.1 Filtro

El módulo de filtro es el encargado de compactar los separadores de *tokens*, es decir, de eliminar múltiples espacios, espacios a inicio de frase, etc. Además puede incluir cualquier otro tipo de función que facilite el desarrollo de los módulos posteriores. En este punto se podrían desarrollar los procesos mencionados en la introducción para convertir un texto en un determinado formato en un texto plano, aunque también se podrían aplicar antes de este módulo.

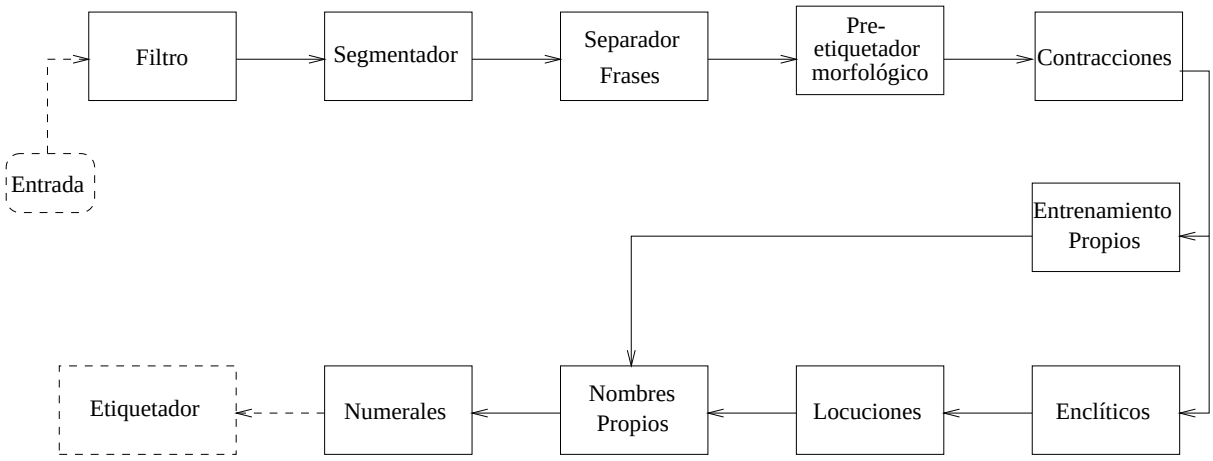

Figura 1: Estructura general del preprocesador.

2.2 Segmentador

La función principal del módulo segmentador es la de identificar y separar los *tokens* presentes en el texto, de manera que cada palabra individual y también cada signo de puntuación constituyan un *token* diferente. El módulo considera las abreviaturas, las siglas, los números con decimales, o las fechas en formato numérico, para no separar el punto, la coma o la barra (respectivamente) de los elementos anteriores y/o posteriores. Para ello utiliza un diccionario de abreviaturas y otro de siglas, así como un conjunto de reglas para la detección de números con decimales y de fechas en formato numérico (`dd/mm/aaaa`).

2.3 Separador de frases

La función de este módulo es la de delimitar las frases [4, 6, 7, 8], tarea que aunque es aparentemente sencilla, presenta numerosas dificultades. La regla general consiste en separar una frase cuando hay un punto seguido de mayúscula, pero se deben tener en cuenta las abreviaturas para no segmentar después de un punto de alguna abreviatura especial como `D.` (Don) que suele ir acompañada de mayúscula, y también las siglas, para no segmentar cada una de las mayúsculas que la componen.

2.4 Preetiquetador morfológico

La función de este módulo es la de etiquetar elementos cuya etiqueta se puede deducir a partir de la morfología de la palabra y no existe una manera más fiable de hacerlo. Así, los números se etiquetan como `Cifra`, al igual que un número seguido del símbolo `%`. De igual manera, se asigna la etiqueta `Data` a las fechas en formatos tales como `7/4/82`, `7 de enero de 2001` o `8 de abril`. En estos últimos casos, se utiliza el símbolo `&` para unir los diferentes elementos del *token*. Por ejemplo, `7 de abril de 1982` produciría la siguiente salida:

```
7&de&abril&de&1982
        [Data 7&de&abril&de&1982]
```

donde los elementos entre corchetes hacen referencia a la etiqueta y al lema del *token* considerado.

2.5 Contracciones

El módulo de contracciones se encarga de desdoblar una contracción en sus diferentes *tokens*, etiquetando además cada uno de ellos. Para ello utiliza información externa que especifica cómo se descomponen las contracciones, de manera que sólo es necesario cambiar esta información para poner en funcionamiento este módulo sobre otro idioma.

Por ejemplo, la salida correspondiente a la contracción `do` (del) es:

```
de [P de]
+o [Ddms o]
```

Es decir, `do` se ha descompuesto en la preposición `de` y en el artículo `+o`. Nótese que el símbolo `+` indica que ha habido una escisión.

2.6 Pronombres enclíticos

Este módulo se encarga de analizar los pronombres enclíticos que aparecen en las formas verbales. Este es un problema importante en algunas lenguas, y en particular para el gallego, donde pueden aparecer hasta cuatro o cinco pronombres enclíticos unidos al verbo.

El objetivo es separar el verbo de sus pronombres, etiquetando correctamente cada una de las partes.

Para realizar su función, en el caso del gallego, este módulo utiliza los siguientes elementos:

- Un diccionario que contiene el máximo número de formas verbales.
- Un diccionario con el máximo número de raíces verbales que pueden llevar pronombres enclíticos.
- Una lista con todas las combinaciones válidas de pronombre enclíticos.
- Una lista con todos los posibles pronombres enclíticos, junto con sus etiquetas y sus lemas.

El proceso consiste en ver si una palabra puede ser un verbo con pronombres enclíticos. Para ello, se analiza la palabra de izquierda a derecha carácter a carácter y se comprueba si es una posible raíz verbal que pueda tener enclíticos. Si es así, se verifica si los caracteres restantes constituyen una combinación de enclíticos válida para esa raíz. En este caso, se hace la segmentación y etiquetación de los correspondientes componentes.

Por ejemplo, la descomposición de `comelo` (comerlo) es:

```
come [V0f000 comer]
     [V0f1s0 comer]
     [V0f3s0 comer]
     [Vfs1s0 comer]
     [Vfs3s0 comer]
+o   [Raa3ms o]
```

donde se observa que los componentes son `comer`, que puede ser infinitivo, infinitivo conjugado o futuro de subjuntivo, y `+o` que es pronombre masculino singular.

2.7 Locuciones

El módulo de locuciones se encarga de unir los *tokens* que componen una locución y de etiquetarlos como una unidad conjunta [3]. En este caso existen dos diccionarios de locuciones: uno con las locuciones que se sabe con seguridad que siempre son locuciones (morfológicamente), y otro donde se encuentran las que pueden serlo o no. De este modo, `a pesar de` (a pesar de) es una locución segura, mientras que `sen embargo` (sin embargo) sería insegura, ya que no se sabe si es locución o si es por un lado `sen` (preprosición) y por otro `embargo` (sustantivo o verbo).

Un dilema muy importante a tener en cuenta consiste en que ante una locución insegura, la información que tiene el módulo no es suficiente para desambiguar. Surge así la necesidad de un elemento de representación de este tipo de fenómenos con el objeto de que sea el módulo adecuado (en este caso, el etiquetador) el que desambigüe entre las dos alternativas. Nuestro preprocesador representaría el ejemplo mencionado de la siguiente manera:

```
<alternativa>
<alternativa1>
sen
embargo
</alternativa1>
<alternativa2>
sen&embargo
</alternativa2>
</alternativa>
```

Como se puede apreciar, se utiliza el símbolo `&` nuevamente para unir los diferentes elementos que forman una mismo *token*.

Este tipo de problema no se presenta únicamente en las locuciones, y profundizaremos algo más en la sección 2.11. No obstante, ya se puede adelantar que el etiquetador debe poder considerar este tipo de representación para proceder a la desambiguación.

2.8 Entrenamiento propios

Este módulo constituye una primera fase para el tratamiento de los nombres propios. Su salida será utilizada por el siguiente módulo (Nombres Propios, según el gráfico de la figura 1) para identificar y etiquetar correctamente estos elementos del texto.

Siguiendo los trabajos de Andrei Mikev [5, 6, 7, 8], se ha implementado un sistema de entrenamiento de nombres propios que reconoce palabras en mayúsculas, siempre y cuando se encuentren en posiciones no ambiguas, es decir, posiciones en las que si una palabra comienza por mayúscula indica efectivamente que es un nombre propio. Por tanto, no se considerarían, por ejemplo, las palabras en mayúscula que aparecen después de un punto. Con estas palabras se genera un nuevo diccionario que será utilizado por el módulo siguiente.

Se identificarán además todas aquellas secuencias de palabras en mayúsculas que aparezcan interconectadas con algún posible nexo válido, formando así un único *token*. Por ejemplo, si en un texto aparece `Consello Superior de Cámaras de Comercio`, se generarían los siguientes nombres propios válidos:

`Consello&Superior&de&Cámaras&de&Comercio`

`Consello&Superior&de&Cámaras`

`Consello&Superior`

`Superior&de&Cámaras&de&Comercio`

`Superior&de&Cámaras`

`Cámaras&de&Comercio`

Todas estas secuencias son añadidas también al diccionario de nombres propios.

2.9 Nombres propios

A partir del diccionario que se ha generado en el módulo anterior, y a partir de los nombres propios que podamos tener en un diccionario externo, este nuevo módulo etiqueta los nombres propios, tanto simples como compuestos, y tanto en posiciones no ambiguas como ambiguas.

Para ello, en el caso de posiciones no ambiguas, se detecta en primer lugar el posible alcance del nombre propio (secuencias válidas que empiezan y terminan con una palabra en mayúscula). Si el alcance total o una subsecuencia de él se encuentran en el diccionario externo, se etiqueta con la etiqueta correspondiente del diccionario. Si por el contrario no existe un subsecuencia en el diccionario externo, se etiqueta como nombre propio pero sin especificar el género.

En el caso de una posición ambigua, se sigue un proceso similar. Se detecta el posible alcance del nombre propio. Si este alcance o una subsecuencia de él se encuentran en el diccionario externo, se le asigna su correspondiente etiqueta. Si por el contrario no existe ninguna subsecuencia en el diccionario externo, pero sí en el diccionario de entrenamiento de nombres propios, se etiqueta como nombre propio sin especificar el género. Y si finalmente no existe ninguna subsecuencia en ninguno de los diccionarios, este módulo no asigna ninguna etiqueta.

Así, por ejemplo, si aparece el nombre propio `Javier Pérez del Río`, y durante el entrenamiento sólo ha aparecido `Pérez del Río` en una posición no ambigua, pero además tenemos que Javier está en el diccionario externo como nombre propio masculino singular, todo el nombre se etiquetaría como nombre propio masculino singular.

2.10 Numerales

Este módulo se encarga de unir numerales para generar un numeral compuesto. Así, si aparece el numeral `mil douscentos vintecinco` (mil doscientos veinticinco), se une cada uno de sus componentes de la misma manera que una locución, produciendo un único *token*.

En este caso, sí es el preprocesador el que asigna al numeral compuesto su etiqueta definitiva. Sin embargo, en el caso de las locuciones, como se ha visto anteriormente, el preprocesador no etiqueta, sino que simplemente genera todas las segmentaciones posibles, siendo el etiquetador el que selecciona posteriormente una de esas alternativas.

2.11 Problemas combinados

Para dar una idea de la complejidad de los problemas que se afrontan vamos a poner algunos casos típicos que se han resuelto.

Ejemplo 1

Supongamos que tenemos la expresión `polo tanto` (por lo tanto o por el tanto). En este caso, estamos ante una locución insegura, es decir, `polo tanto` puede ser una locución, `polo` a su vez puede ser un sustantivo, una contracción o un verbo con pronombres enclíticos, y `tanto`, por su parte, puede ser sustantivo o adverbio, si no forma parte de la locución. La respresentación sería la siguiente:

```
<alternativa>
<alternativa1>
polo [Scms polo]
tanto
</alternativa1>
<alternativa2>
por  [P por]
+o   [Ddms o]
tanto
</alternativa2>
<alternativa3>
po   [Vpi2s0 pór] [Vpi2s0 poñer]
+o   [Raa3ms o]
tanto
</alternativa3>
```

```
<alternativa4>
por&+o&tanto
</alternativa4>
</alternativa>
```

Un ejemplo de aplicación de las 4 diferentes acepciones sería:

- **Sustantivo+Adverbio**: `Coméche-lo polo tanto, que non quedaron nin os osos` (comiste el pollo tanto, que no quedaron ni los huesos).
- **Preprosición+Artículo+Sustantivo**: `Gañaron o partido polo tanto da estrela` (ganaron el partido por el tanto de la estrella).
- **Verbo+Pronombre+Adverbio**: `Pois agora polo tanto ti coma el` (pues ahora lo pones tanto tú como él).
- **Locución**: `Estou enfermo, polo tanto quédome na casa` (estoy enfermo, por lo tanto me quedo en casa).

Ejemplo 2

Un ejemplo de conflicto entre dos posibles descomposiciones de enclíticos aplicable al español sería `ténselo`, que puede ser `tense` (de tensar) más `lo` ó `ten` (de tener) más `se` más `lo`, por lo que la representación correspondiente es:

```
<alternativa>
<alternativa1>
ténse [V2spm0 tensar]
+lo    [Re3sam el]
</alternativa1>
<alternativa2>
tén   [V2spm0 tener]
+se    [Re3yyy se]
+lo    [Re3sam el]
</alternativa2>
</alternativa>
```

3 Etiquetador

No es el objetivo de este trabajo la presentación de nuestro etiquetador/desambiguador, pero al menos una breve descripción de su funcionamiento interno tiene cierta relevancia. Debido a las segmentaciones ambiguas que hemos descrito anteriormente, este etiquetador debe ser capaz de enfrentarse a flujos de *tokens* de distinta longitud. Es decir, no sólo debe ser capaz de decidir qué etiqueta asignar a cada *token*, sino que además debe saber decidir si algunos de ellos constituyen o no una misma entidad y asignar, en cada caso, la cantidad de etiquetas adecuada, en función de las alternativas de segmentación que le proporciona el preprocesador.

Para llevar a cabo este proceso, se podría considerar la construcción de etiquetadores especializados para todas las posibles alternativas, la posterior comparación de sus correspondientes salidas, y la selección de la más plausible. No obstante, esta posibilidad plantea varios inconvenientes. En primer lugar, sería necesario definir algún criterio objetivo de comparación. Si el paradigma de etiquetación que se está utilizando es, por ejemplo, el de los modelos de Markov [2] ocultos, como es nuestro caso, dicho criterio podría ser la comparación de las probabilidades acumuladas normalizadas. En otros paradigmas, los criterios podrían no ser tan sencillos de identificar, y, en cualquier caso, la evaluación individual de cada posible combinación de las diferentes alternativas conllevaría un coste computacional bastante elevado.

Por ello, en nuestro caso, hemos preferido abordar el diseño de una extensión del algoritmo de Viterbi [9] que, sin pérdidad de generalidad y sin necesidad de información extra, es capaz de etiquetar flujos de *tokens* de distinta longitud, y además con una complejidad temporal comparable al del algoritmo de Viterbi clásico. Este es, por tanto, el paso final del proceso, y su salida es precisamente el texto segmentado y desambiguado.

4 Conclusiones y trabajo futuro

Este trabajo presenta realizaciones concretas de mecanismos específicos de preprocesamiento y segmentación. Como se ha podido comprobar, la complejidad del tratamiento de muchos de los fenómenos que aparecen en este nivel es elevada, y por esta razón muchos de ellos suelen ser obviados en las aplicaciones finales.

La presentación ha sido orientada hacia la obtención de mejoras en la etiquetación robusta de los textos. No obstante, la utilización y necesidad de los mecanismos de preprocesamiento y segmentación no se limita únicamente a la desambiguación automática. El lugar de etiquetador podría ser ocupado por cualquier otra herramienta de análisis (sintáctico, semántico, etc.), o simplemente por un *scanner* que proporcione todas las alternativas de segmentación y sus correspon-

dientes etiquetaciones, permitiendo así realizar de manera más cómoda las labores de desambiguación manual.

Actualmente, este último uso está siendo explotado intensamente. Debemos tener en cuenta que nuestro trabajo se centra en torno al procesamiento automático del gallego, lengua para la cual la carencia de recursos lingüísticos es casi absoluta.

Entre nuestros objetivos futuros más inmediatos se encuentra la mejora y generalización de los algoritmos propuestos para permitir una mejor adaptación a diferentes idiomas, simplificando en la medida de lo posible este proceso.

Por otra parte, es necesario el desarrollo de nuevas técnicas que permitan la realización tanto de las tareas de preprocesamiento que ya hemos contemplado aquí, como de otras todavía en estudio, pero en cualquier caso sin la utilización de una gran cantidad de recursos linguísticos (que pueden no existir y pueden ser muy costosos de construir). En caso de estar disponibles, estos recursos se utilizarían como un refinamiento del comportamiento global, pero no como un requisito.

El objetivo final de la arquitectura general de preprocesamiento y segmentación que hemos presentado es su integración en un sistema global de recuperación de información, y se espera que el trabajo específico de los módulos descritos contribuya a mejorar el rendimiento de dicho sistema.

A Descripción de las etiquetas

Este apéndice describe las etiquetas utilizadas en los ejemplos incluidos en el presente trabajo. Como ya se ha mencionado anteriormente, dichas etiquetas provienen de los *tag sets* utilizados en los proyectos GALENA y CORGA. Las correspondientes descripciones son:

Etiquetas GALENA

`C31`	Primer elemento de una conjunción de tres elementos
`C32`	Segundo elemento de una conjunción de tres elementos
`C33`	Tercer elemento de una conjunción de tres elementos
`Re3sam`	Pronombre enclítico acusativo masculino tercera persona del singular
`Re3yyy`	Pronombre enclítico acusativo o dativo masculino o femenino tercera persona singular o plural
`V000f0PE1`	Verbo infinitivo con un pronombre enclítico
`V2spm0`	Verbo presente de imperativo segunda persona del singular

Etiquetas CORGA

`Cifra`	Cifra
`Data`	Fecha
`Ddms`	Artículo determinante masculino singular
`P`	Preposición
`Raa3ms`	Pronombre átono acusativo masculino tercera persona del singular
`Scms`	Sustantivo común masculino singular
`V0f000`	Verbo infinitivo
`V0f1s0`	Verbo infinitivo conjungado primera persona del singular
`V0f3s0`	Verbo infinitivo conjungado tercera persona del singular
`Vfs1s0`	Verbo futuro subjuntivo primera persona del singular
`Vfs3s0`	Verbo futuro subjuntivo tercera persona del singular
`Vpi2s0`	Verbo presente indicativo segunda persona del singular

Referencias

[1] Álvarez, Rosario; Regueira, X.L; Monteguado, H. (1986). Gramática Galega. *Ed. Galaxia*

[2] Brants, T. (2000). TNT - A statistical part-of-speech tagger. In *Proceedings of the Sixth Applied Natural Language Processing Conference (ANLP-2000)*, Seattle, WA.

[3] Chanod, Jean-Pierre; Tapanainen, Pasi. (1996). A Non-deterministic Tokeniser for Finite-State Parsing. *ECAI'96 workshop on Extended finite state models of language*, Budapest.

[4] Grefenstette, Gregory; Tapanainen, Pasi. (1994). What is a word, What is a sentence? Problems of Tokenizacion *3rd Conference on Computational Lexicongraphy and Text Research*, COMPLEX'94, July 7-10, 1994.

[5] Mikev, Andrei. (1999). A knowledge-free Method for Capitalizad Word Disambiguation. *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, 20-26 June 1999, Maryland, USA.

[6] Mikev, Andrei. (1999). Periods, Capitalized Words, etc. Disponible en `http://www.ltg.ed.ac.uk/~mikheev/papers.html`

[7] Mikev, Andrei. (2000). Document Centered Approach to Text Normalization. *Proceedings of SIGIR'2000*, Athens, pp.136-143.

[8] Mikev, Andrei. (2000). Tagging Sentence Boundaries. *Proceedings of NAACL'2000*, Seatle, pp.264-271

[9] Viterbi, A.J. (1967). Error bounds for convolutional codes and an asymptotically optimal decoding algorithm. *IEEE Trans. Information Theory*, vol. IT-13 (April), pp. 260-269.