Adaptación del Método de Etiquetado No Supervisado TBL

Jesús González Martí
Intelligent Dialogue Systems
Sevilla
j.gonzalez@indisys.es

David González Maline
G. Investigación Julietta
Universidad de Sevilla
dgmaline@us.es

José Antonio Troyano
Departamento LSI
Universidad de Sevilla
troyano@lsi.us.es

Resumen: Se presenta una mejora al método de etiquetado de Brill basado en reglas y transformaciones. Se disminuye considerablemente los tiempos de entrenamiento sin sacrificar de ninguna manera la eficicacia del método.
Palabras clave: Brill, TBL, regla, transformación, etiquetado morfosintáctico, categoria

Abstract: This paper proposes an improvement of the Brill's "Transformation-Rule Based" POS-Tagger Algorithm. Our improvement decreases training times considerably without affecting the accuracy of the algorithm.
Keywords: Brill, TBL, rule, transformation, Part-of-OS tagging, category

1. Introducción

De sobra es conocida la necesidad de realizar el etiquetado morfosintáctico dentro de los procesos de tratamiento del lenguaje natural. Para llevar a cabo esta tarea se pueden utilizar diversos tipos de algoritmos, entre los cuales podemos encontrar el propuesto por Brill(Brill, 1992)(Brill, 1994) y en concreto la versión no supervisada de su etiquetador basado en transformaciones(Brill, 1995).

La gran ventaja de este método es que permite una mayor implicación por parte de los lingüistas a la hora de mejorarlo y adaptarlo a los diferentes ámbitos de aplicación del mismo. Sin embargo, y a pesar de la efectividad del método, los tiempos de entrenamiento que requiere un algoritmo de este tipo son considerablemente superiores a los tiempos que emplean los etiquetadores basados en los modelos de Markov.

Existen gran cantidad de trabajos que intentan mejorar de alguna forma este problema: (Roche y Schabes, 1995), (Ramshaw y Marcus, 1996), (Aone y Hausman, 1996), (Ngai y Florian, 2001), etc... Pero la mayoría de los mismos sacrifican la eficacia del etiquetado por velocidad de ejecución; por consiguiente el método cae en desventaja con respecto a otros que proporcionan un mejor resultado en tiempos semejantes. Todos los trabajos mencionados se centran en estudiar las probabilidades de las reglas para evitar generar aquellas que a posteriori sean innecesarias. Sin embargo, esto no se cumple siempre y a veces se pierden reglas que podrían ser muy provechosas para aumentar el número de palabras correctamente etiquetadas.

En este artículo se presenta una adaptación del método de Brill que no desecha ninguna regla, pero que realiza una mejora temporal muy interesante ya que permite competir con los ya mencionados algoritmos basados en modelos de Markov. En definitiva, aumenta en proporción la calidad de los resultados respecto a los tiempos de entrenamiento, sin sacrificar el grado de precisión y cobertura de las raglas obtenidas.

2. Algoritmo de Brill No Supervisado

El algoritmo de Brill parte de un texto sin etiquetar y de un diccionario que contiene las diferentes categorías gramaticales a las que puede optar una palabra. A continuación realiza los siguientes pasos:

1. En primer lugar, etiqueta cada palabra con su conjunto de etiquetas correspon-

diente.

2. Genera todas las reglas posibles basándose en un conjunto determinado de plantillas (*PREVWORD, NEXTWORD, PREVTAG, NEXTTAG, ...*).[1]
3. Puntúa cada regla en función de una serie de parámetros, como por ejemplo la frecuencia de aparición de las mismas.
4. Si no existe ninguna regla que haya sido puntuada por encima de un cierto umbral termina el entrenamiento del etiquetador. Si existe, la aplica a todas las palabras del texto que concuerden con el contexto de la regla y la añade al conjunto de reglas aprendidas por el etiquetador.
5. Volver al paso 2.

Veamos un breve ejemplo. Supongamos el siguiente estado en el proceso de entrenamiento del etiquetador:

... elevaba(VERB) a(PREP-NOMB) la(DET-NOMB) orilla(NOMB-VERB) misma(ADJ) del(DET) río(NOMB-VERB) y(CONJ) ...

...

... habitaba(VERB) un(DET) pueblo(NOMB) de(PREP) hombres(NOMB) vigorosos(ADJ) ...

Para la palabra ambigua "orilla" de la primera frase y tomando como referencia la palabra no ambigua "pueblo" de la seguda frase, tenemos la siguiente coincidencia de contexto:

- PREVTAG(DET): NOMB-VERB → NOMB

Como se ve, para las dos palabras "orilla" y "pueblo", existe una posible etiqueta común que les precede: "DET". Dado que tenemos un caso no ambiguo ("pueblo") en el cual una palabra posee la categoría gramatical de "NOMB", en el caso de "orilla", generamos la regla anteriormente citada. Si esta regla obtiene la mayor puntuación se aplicará y en el siguiente paso del algoritmo el texto se habrá transformado en:

... elevaba(VERB) a(PREP-NOMB) la(DET-NOMB) orilla(NOMB) misma(ADJ) del(DET) río(NOMB-VERB) y(CONJ) ...

[1] La versión implementada por los autores sólo contempla los citados tipos de plantillas.

3. Variación de Estados

La mejora del algoritmo se basa en un estudio de los estados por los que pasa el proceso de entrenamiento. En cada iteración, se genera un conjunto de reglas posibles entre las que elegir aquella que obtenga la mayor puntuación, que normalmente coincide con aquella que más palabras desambigua.

Podemos definir un estado del algoritmo mediante los siguientes elementos: el texto de entrada, el conjunto de reglas candidatas, la regla de mayor puntuación y el conjunto de reglas aprendido durante los anteriores estados del algoritmo. Para conseguir una mejora que no comprometa la eficacia debemos conseguir un algoritmo diferente, más eficiente pero que mantenga el mismo número de estados así como la secuencia de los mismos. Por tanto, la idea clave de esta propuesta consiste en encontrar cuáles son los factores que implican un cambio de un estado a otro, trabajar con dichos elementos y generar un nuevo algoritmo que no necesite el esfuerzo computacional del original.

Para proponer una nueva solución se estudió previamente el algoritmo original para encontrar su parte crítica, esto es, la que mayor carga computacional soporta. Ésta recae sobre la tarea de generar el conjunto de reglas candidatas y puede llegar a consumir el 85 % del tiempo del algoritmo. Esto es debido a que el algoritmo original no tiene más remedio que volver a generarlas a cada paso o estado. La puntuación de reglas también consume una parte importante del tiempo pero no es significativa en comparación al tiempo necesario para generar las miles (o incluso millones) de reglas que pueden llegar a generarse.

El problema de la generación de dichas reglas consume mucho tiempo porque requiere recorrer el texto de entrada un número muy elevado de veces en cada iteración:

$$\frac{n^{o}\ de\ recorridos}{n^{o}\ iteraciones} = (palabras\ ambiguas)^2$$

4. Generación Rápida de Reglas

Una vez se estudian los diferentes estados por los que pasa el algoritmo durante sus ejecuciones, se puede observar que el estado de reglas candidatas varía en función de una serie de patrones que presentamos a

continuación:

- Aparición en nuevos contextos

  La aparición de nuevas reglas en el conjunto de reglas candidatas cuando se cambia de un estado a otro en una iteración, se debe exlusivamente a la aparición de nuevos contextos que cuadren con las palabras que aún siguen siendo ambiguas en el texto. Esto quiere decir que, cuando desambiguamos una palabra en una iteración, en la siguiente pueden aparecer nuevas reglas originadas por las palabras ambiguas que ahora tienen en común su contexto con el contexto no ambiguo.

  Por ejemplo, dado el siguiente fragmento de un texto sobre el que está trabajando el sistema de entrenamiento:

  ... elevaba(VERB) a(PREP-NOMB) la(DET-NOMB) orilla(NOMB-VERB) misma(ADJ) del(DET) río(NOMB-VERB) y(CONJ) ...

  ... la(NOMB)[2] *mayor(NOMB-ADJ) parte(NOMB-VERB) de(PREP) la(DET-NOMB) gente(NOMB) ...*

  ... habitaba(VERB) un(DET) pueblo(NOMB) de(PREP) hombres(NOMB) vigorosos(ADJ) ...

  Si tenemos en cuenta el reducido conjunto de reglas que se generarían a partir de la palabra ambigua "orilla" de la primera frase, vemos que sólo existe una palabra no ambigua que concuerda con su contexto para la plantilla *PREVTAG*: "pueblo", pues tanto la palabra que la precede, como la que precede a "orilla" pertenecen a la categoría gramatical de *determinante*. Por lo tanto la única regla que se genera sería:

  - PREVTAG(DET): NOMB-VERB → NOMB

  Esta regla significa que si estamos ante una palabra que puede ser o bien un nombre o bien un verbo y la palabra que le precede contiene en sus posibles etiquetas *DET*, entonces debemos elegir para ella la categoría de *NOMB*.

  En este mismo estado, la regla que obtiene mayor puntuación sirve para desambiguar la palabra "mayor" a *NOMB*. Si ahora tuvieramos que volver a generar todas las reglas del conjunto para la palabra "orilla", ya no es "pueblo" quien concuerda en contexto, sino también la palabra "mayor" generando estas dos nuevas reglas en el conjunto de reglas posibles:

  - PREVTAG(NOMB): NOMB-VERB → NOMB
  - PREVWORD("la"): NOMB-VERB → NOMB

- Desaparición por la palabra desambiguada

  En cuanto a la desaparición de reglas, se pueden dar dos casos. El más evidente es el caso en el que una palabra ambigua que generaba reglas, pasa a ser no ambigua. En este caso, desaparecen todas las reglas que se generaban utilizándose dicha palabra como referencia.

  En el ejemplo anterior, teníamos un total de tres reglas generadas a partir de la palabra "orilla" como palabra ambigua de referencia. Si en una de las iteraciones se eligiera una regla que la desambiguase, las tres reglas que se generaron en el caso anterior desaparecerían.

- Desaparición por contextos propios y ajenos

  Existe un segundo caso de desaparición de reglas que se produce al desambiguar palabras que cuadran con los contextos de alguna palabra potencialmente utilizable para la generación. Por ejemplo, presentando el siguiente estado análogo al anterior pero con la palabra "mayor" desambiguada:

  ... elevaba(VERB) a(PREP-NOMB) la(DET-NOMB) orilla(NOMB-VERB) misma(ADJ) del(DET) río(NOMB-VERB) y(CONJ) ...

[2] La categoría *NOMB* viene del nombre de la nota musical "LA". Considerese el ejemplo como un caso en el que se ha desambiguado incorrectamente.

... la(NOMB) mayor(NOMB) parte(NOMB-VERB) de(PREP) la(DET-NOMB) gente(NOMB) ...

... habitaba(VERB) un(DET) pueblo(NOMB) de(PREP) hombres(NOMB) vigorosos(ADJ) ...

Las reglas que se generan son las siguientes:

- PREVTAG(DET): NOMB-VERB → NOMB
- PREVTAG(NOMB): NOMB-VERB → NOMB
- PREVWORD("la"): NOMB-VERB → NOMB

Sin embargo, se puede observar que el "la" que precede a la palabra "orilla" es también una palabra ambigua, lo cual da lugar a dos contextos diferentes a tener en cuenta a la hora de generar reglas. Si dicha palabra fuera desambiguada en una de las iteraciones del algoritmo, tendríamos que tener en cuenta cuáles son las reglas que desaparecen por esta causa. En este caso concreto, si desambiguamos "la" a *determinante*, la regla que desaparecería sería:

- PREVTAG(NOMB): NOMB-VERB → NOMB

5. Resultados

Para la realización de las pruebas de medición de tiempos y comparación entre algoritmos, se utilizó una máquina con procesador Intel Pentium III a 800 Mhz, 256 MBytes de memoria RAM y ejecutando un sistema operativo Windows 2000.

La implementación del algoritmo original utilizada fue creada a partir del artículo publicado por Brill sobre el entrenamiento no supervisado. Primero se realizó una implementación "al pie de la letra" y a continuación se implementó una versión que usaba estructuras de datos indexadas. Finalmente se realizó una tercera implementación que sacaba provecho de los principios mencionados en la sección anterior. Cada versión se fue construyendo en base a la anterior por lo que todo el código común es compartido haciendo de este modo más fiable los resultados y comparativas.

El siguiente cuadro muestra los diferentes tiempos de ejecución (en segundos) del algoritmo en sus tres variantes utilizadas en las pruebas:

Tamaño del Texto	**1 Kb**	**5 Kb**	**10 Kb**	**20 Kb**
Original	1.14	160.97	640.87	5329.37
Indexada	0.53	103.95	408.24	2834.84
Rápida	0.34	48.31	138.13	807.09

Cuadro 1: Tiempos de ejecución

El mismo resultado puede apreciarse gráficamente en la figura 1. Se observa el crecimiento exponencial del tiempo de ejecución en las tres versiones. El número de reglas que se puede llegar a generar en un texto depende exponencialmente del tamaño del mismo. Se observa que la diferencia de tiempos a medida que el tamaño crece es muy considerable.

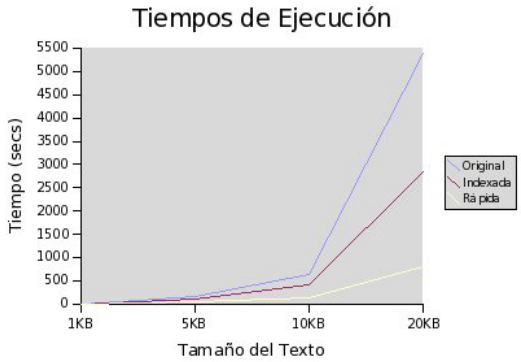

Figura 1: Tiempos de ejecución

Como se citó anteriormente, la mejora se ha realizado únicamente en la generación de reglas y no en otra parte del algoritmo. Esta característica puede apreciarse mejor si medimos la diferencia de tiempos que existe entre cada iteración. Ya que las tres implementaciones se comportan exactamente de la misma forma, incluso a la hora de conservar el estado entre las iteraciones, realizamos la medición entre la versión indexada y la versión mejorada, obteniendo una gráfica como la de la figura 2.

Figura 2: Tiempos de iteración

Se observa que la ejecución del algoritmo mejorado parece mucho más constante en su tiempo de iteración que el algoritmo original. Esto se debe a que el algoritmo original genera la totalidad del conjunto de reglas posibles en cada iteración, mientras que el algoritmo propuesto tan sólo realiza dicha operación al comienzo de la ejecución misma. De este modo, realiza en cada iteración únicamente los cálculos necesarios para averiguar las modificaciones del conjunto de reglas de la iteración anterior.

La figura 3 muestra la diferencia de tiempos en cada iteración de los dos algoritmos comparados. Ya que el tiempo de puntuación de las reglas es idénticamente igual en ambos algoritmos, se observa que la disminución en la diferencia se debe a cómo progresivamente decrece el número de reglas candidatas que se pueden calcular, consecuencia lógica de cómo se van desambiguando las palabras en el texto.

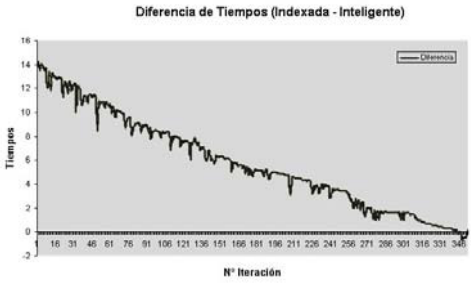

Figura 3: Diferencia de Tiempos

6. *Conclusión*

En este artículo se ha buscado una solución a la lentitud de ejecución del algoritmo de entrenamiento para el etiquetado no supervisado propuesto por Brill. Para ello se han estudiado los diferentes estados por los que pasa el algoritmo, así como los factores que hacen variar de un estado a otro, de una iteración a la siguiente. A partir de esa información, se diseña un nuevo algoritmo capaz de evitar tener que regenerar la información obtenida del texto sino que se genera basándose en el estado anterior.

Es importante hacer hincapié en el hecho de que los conjuntos de reglas solución de la versión de Brill y la nueva versión propuesta son exactamente idénticos. Esta solución por tanto puede definirse como una adaptación que mejora la eficiencia temporal del algoritmo original.

El resultado es un nuevo etiquetador no supervisado que se ejecuta en unos márgenes de tiempo considerablemente inferiores a los de la versión original de Brill, con la ventaja añadida de que el comportamiento y los resultados del algoritmo con respecto al original se mantienen intactos.

7. *Trabajo Futuro*

Al igual que se ha conseguido reducir el tiempo de generación de las reglas candidatas en cada iteración del algoritmo, de un modo semejante se podría mejorar la eficiencia en el tiempo necesario para la puntuación de las mismas. Este enfoque saca partido de que la variación en las puntuaciones de las reglas siguen también ciertos patrones de modificación semejantes a los del conjunto de reglas candidatas.

Dado que en la versión mejorada aquí propuesta la generación de reglas ha sido reducida a un segundo plano en cuanto a consumo de recursos, el nuevo cuello de botella es la puntuación. La posible reducción del tiempo de puntuación daría lugar a un algoritmo mucho más rápido que el actual, aunque en proporción, no se conseguirá una mejora comparable a la descrita en el presente trabajo.

Puesto que las mejoras presentadas se han realizado sobre el etiquetador original de Brill, sería interesante relizar comparativas con otros etiquetadores (por ejemplo: *fnTBL*), incluso etiquetadores basados en modelos de Markov (como *TNT*).

Por otro lado, sería muy interesante comparar el grado de eficacia del algoritmo cuando se aumenta el número de plantillas posibles. En nuestro estudio sólo se han implementado PREVWORD, NEXTWORD, NEXTTAG, y PREVTAG. Algunas plantillas posibles serían: PREVPREVWORD (palabra anterior a la anterior), PREVPREVTAG (etiqueta de la palabra anterior a la anterior), ..., 2-PREVWORD (las dos palabras anteriores), 2-NEXTTAG (las etiquetas de las dos palabras posteriores), etc...

Bibliografía

Aone, Chinatsu y Kevin Hausman, 1996. *Unsupervised learning of a rule-based Spanish Part of Speech tagger.*

Brill, Eric. 1992. A simple rule-based part os speech tagger. En *3rd Conference on Applied Natural Language Processing. Trento, Italia.*

Brill, Eric. 1994. Some advances in rule-based part of speech tagging. En *12th Conference on Artificial Intelligence (AAAI-94). Seattle, Wa.*

Brill, Eric. 1995. Unsupervised learning of disambiguation rules for part of speech tagging. En *3rd Workshop on Very Large Corpora.*

Ngai, Grace y Radu Florian, 2001. *Transformation-based learning in the fast lane.*

Ramshaw, Lance A. y Mitchell P. Marcus, 1996. *Exploring the statistical derivation of transformation rule sequences for part-of-speech tagging.*

Roche, Emmanuel y Yves Schabes, 1995. *Deterministic Part-of-Speech tagging with finite state transducers.*