Morfología de estados finitos en software libre: aplicación al euskera[1]

Finite-state morphology using free software: application to Basque

Iñaki Alegria, Izaskun Etxeberria, Nerea Ezeiza, Montse Maritxalar
Grupo IXA. Euskal Herriko Unibertsitatea. 649 Postakutxa. 20080 Donostia
i.alegria@ehu.es

Resumen: Se presenta como se ha adaptado la descripción morfológica del euskera desde la descripción para Xerox a *foma*, nuevo software libre para morfología de gran eficiencia.
Palabras clave: morfología, euskera, software libre

Abstract: In this paper we describe the process of conversion and testing of the description for the Basque morphology from the Xerox toolkit to *foma*, a new open-source tool.
Keywords: morphology, Basque, open source software

1 Morfología de estados finitos

Un procesador morfológico es una herramienta básica para el PLN. Si el idioma es de flexión rica una lista de palabras con su análisis correspondiente no es una buena solución.

La propuesta más adecuada para crear un procesador morfológico basado en la morfología de dos niveles para un idioma aglutinante como el euskera era utilizar la morfología de dos niveles propuesta por Koskenniemi, (Alegria, 1996). El exitoso corrector ortográfico Xuxen está basado en esta tecnología. La morfología se describe por medio de dos ficheros, el léxico, donde se describen los morfemas y los conjuntos de morfemas que les pueden seguir (morfotáctica); y las reglas fonológicas, que describen los cambios producidos al encadenar los morfemas.

La idea de la morfología de dos niveles fue tomada y mejorada por Xerox dando lugar a la morfología de estados finitos (Beesley y Karttunen, 2003). La diferencia fundamental entre la propuesta de Koskenniemi y la posterior evolución en Xerox es que permiten tanto reglas paralelas como reglas secuenciales para expresar las reglas fonológicas.

Las reglas paralelas tienen la ventaja de que el orden no es significativo y que no hace falta definir lenguajes intermedios entre las palabras del texto (nivel superficial) y la representación léxica (nivel léxico). Sin embargo las reglas paralelas deben tener en cuenta en sus contextos los efectos de las reglas relacionadas, y esto muchas veces es fuente de errores. Ambos tipos de reglas pueden ser compiladas y convertidas a transductores muy eficientes. Veamos un ejemplo en euskera. El prefijo *ber* antecede a las raíces verbales. Delante de vocal la erre se dobla (es *r* fuerte) y delante de consonante se transforma en *bir*. Además delante de una *h*, hace que la *h* desaparezca. Si representamos el prefijo como *beR* (marcando la r fuerte), tenemos que de *beR*+*egin* (nivel léxico) se genera *berregin* (nivel superficial), de *beR*+*gai*+*tu birgaitu* y de *beR*+*hasi berrasi*.

Usando reglas paralelas (simplificadas) tenemos la descripción del ejemplo (1).

```
R->r ; +->0 ;
h->0  || # b e R:r 0:r +:0 _ Vocal;
0->r  || R:r _ +:0 (h:0) Vocal;
e->i  || # b _ R +:0 Cons;
```

Usando reglas secuenciales (simplificadas) tenemos la descripción del ejemplo (2).

```
h->0  || # b e R + _ Vocal;
0->r  || R _ + Vocal;
e->i  || # b _ R + Cons;
R->r ; +->0 ;
```

En el segundo grupo las reglas son más sencillas, pero su orden es significativo.

El toolkit de Xerox proporciona herramientas para reglas paralelas (*twolc*) y reglas secuenciales (*xfst*), además de para el léxico (*lexc*). Su experiencia indica que el uso de reglas secuenciales resulta a la larga más cómodo, tendencia que se ha visto confirmada en posteriores implementaciones. Las reglas

[1]El trabajo aquí presentado se ha realizado gracias a la financiación por parte del MEC del proyecto *OpenMT: Traducción automática en código libre mediante métodos híbridos*. TIN2006-15307-C03-01)

paralelas para el euskera se describen en Alegria (1996).

2 Morfología en software libre

Las herramientas de Xerox son de gran calidad y consiguen transductores muy compactos, pero tienen un gran inconveniente; la licencia es muy restrictiva y salvo excepciones no puede ser usada para aplicaciones comerciales. Ante ello se han desarrollado en software libre herramientas que pueden sustituir las de Xerox. Queremos destacar dos: *hunspell* y *foma*.

hunspell no trabaja con transductores, pero es una solución mejor que sus predecesores *ispell*, *aspell* y *myspell*, ya que permite mayor número de paradigmas y doble encadenamiento de sufijos. No acepta reglas paralelas ni secuenciales independientes del léxico, ya que los cambios deben ser explicitados en la descripción del léxico. Su mayor interés es que está aceptado por las últimas versiones de OOffice y Mozilla, por lo que una descripción para un idioma usando *hunspell* deviene automáticamente en un corrector ortográfico para estas herramientas.

foma (Hulden, 2009) es un toolkit que quiere ser equivalente a las herramientas *xfst* y *lexc* de Xerox pero con una implementación independiente y licenciado bajo GPL (*http://foma.sourceforge.net/*). Por lo tanto es adecuado para descripciones de morfología de estados finitos usando reglas secuenciales. Según su autor las descripciones para Xerox funcionan directamente, y la compilación es más eficiente. Tiene una ventaja adicional, y es que las descripciones preparadas para *lexc* y *xfst* de Xerox se compilan directamente en *foma* (salvo alguna excepción y siempre que los datos estén en Unicode)*;* ya que esta nueva herramienta integra los mismos comandos y la misma sintaxis que las herramientas citadas. Esto hace que el libro citado (Beesley y Karttunen, 2003) pueda ser usado en su mayor parte como un manual de *foma*.

3 Migración a Foma

La demo que se propone es el resultado de migrar a *foma* nuestra descripción para Xerox. El mayor trabajo ha sido convertir las reglas paralelas a secuenciales, o mejor dicho, pasar de la descripción para twolc (que solo permite reglas paralelas) a la descripción correspondiente para *xfst* o *foma*. Para los idiomas que tienen sus propias descripciones en *xfst* esto no haría falta.

La descripción del léxico fue aceptada directamente por *foma* sin ningún tipo de transformación salvo la conversión a Unicode de ciertos caracteres. Esto demuestra la robustez del compilador ya que el léxico contiene más de 80.000 entradas con la correspondiente información morfológica relacionada. Las reglas tuvieron que ser revisadas y ordenadas cuidadosamente para pasar de reglas paralelas a reglas secuenciales, ya que como se ha visto en el ejemplo hay reglas interrelacionadas y el orden de las reglas secuenciales es significativo. Debido a ello la mayor dificultad consistió en establecer un orden de las reglas, ya que las 23 reglas complejas existentes abarcan un gran número de cambios, que pueden estar relacionados con cambios en otras reglas. La metodología supuso incluir primero las reglas de carácter morfológico y después las de carácter fonológico.

Hemos comprobado que, como defiende el autor de *foma*, la compilación se realiza más rápidamente que las herramientas de Xerox, y que el número de estados y arcos es similar. Además de las reglas del euskera estándar, tenemos otro sistema de reglas no estándares, con el fin de detectar, y en su caso corregir, errores de competencia y variantes dialectales. Estas reglas también han sido convertidas y probadas con *foma*. Así es posible obtener la forma estándar correspondiente a una forma dialectal o a un error de competencia, y además saber las reglas que se han aplicado. Esto es interesante para sistemas de aprendizaje de idiomas asistidos por ordenador.

En resumen, podemos decir que *foma* es una herramienta que, a pesar de estar todavía en desarrollo, tiene la madurez y robustez suficiente para permitir su uso en herramientas NLP de amplia cobertura. Queremos agradecer al autor de *foma*, Mans Hulden, la ayuda que nos ha ofrecido en este trabajo.

Bibliografía

Alegria I. 1996. Morfología de estados finitos. Revista SEPLN no. 18, 1-26. Donostia.

Beesley K. R. and Karttunen L. 2003. Finite State Morphology. CSLI Publications, Palo Alto, CA.

Hulden M. 2009. Foma: a Finite-State Compiler and Library. EACL 2009. Demo session. pp 29-32.