# Una cascada de transductores simples para normalizar tweets*

## *A cascade of simple transducers for tweet-normalization*

**Iñaki Alegria, Izaskun Etxeberria, Gorka Labaka**
IXA Taldea, UPV/EHU
649 Postakutxa, 20080 Donostia
i.alegria@ehu.es, izaskun.etxeberria@ehu.es, gorka.labaka@ehu.es

**Resumen:** Se presenta un sistema basado en la concatenación de varios transductores o FSTs. Cada uno de los transductores se encarga de completar un hito más o menos simple: ejemplos aprendidos, entidades nombradas, errores básicos, palabras contiguas unidas, onomatopeyas, cambios complejos, cambios en mayúsculas.
**Palabras clave:** Normalización, Transductores, Modelo *noisy-channel*, WFST

**Abstract:** A system where several transducer or FST are combined in cascade is presented. Each transducer manages a simple step: learned examples, named-entities, basic misspellings, collapsed words, onomatopoeia words, more complex changes, lowercase/uppercase letters.
**Keywords:** Tweet-normalization, Transducers, noisy-channel model, WFST

## *1. Introducción*

El sistema que desde el grupo IXA de la UPV-EHU presentamos para la resolución de la tarea de normalización de tweets se basa fundamentalmente en la concatenación de varios transductores o FSTs. Cada uno de los transductores se encarga de completar un hito más o menos simple y la solución final llega de la suma de todos esos hitos. Las características de la tarea y los datos utilizados, así como el estado del arte, no se detallan por falta de espacio pero pueden ser consultados en el sitio web del workshop[1].

La tarea de normalización de los tweets tiene cierta similitud con otras tareas en las que nuestro grupo de investigación viene trabajando (Alegria, Etxeberria, y Leturia, 2010), especialmente con la de normalización de variantes diacrónicas o dialectales en euskera (Hulden et al., 2011).

## *2. Métodos y arquitectura*

Tal y como se ha comentado en el apartado anterior, el sistema que se propone consiste en la concatenación de diversos transductores simples, cada uno con una función muy concreta. La arquitectura del sistema consta así de una serie de etapas secuenciales, cada una de las cuales da respuesta a nuevos casos. Las etapas en las que se divide el sistema son las siguientes: (1) Aprendizaje de los ejemplos disponibles en el corpus de desarrollo; (2) Reconocimiento de entidades; (3) Tratamiento de cambios ortográficos básicos; (4) Tratamiento de palabras adyacentes unidas; (5) Tratamiento de onomatopeyas; (6) Resolución de casos más complejos.

Como veremos, en la última etapa se proponen soluciones alternativas que posteriormente se intentan combinar: distancia de edición, cambios morfofonológicos más complejos y modelo *noisy-channel*.

En varios de los transductores que describiremos hemos utilizado la herramienta *foma* (Hulden, 2009). Se trata de una versión libre equivalente a las herramientas de Xerox (Beesley y Karttunen, 2002) que permite describir y generar transductores.

Si tras la concatenación de todas las respuestas, finalmente hay palabras no-normalizadas para las que no se tiene propuesta alguna, el sistema dará como respuesta la propia palabra de entrada.

El tema de las mayúsculas y su variación también se ha demostrado importante. Debido a esa variación y al pequeño tamaño del corpus de desarrollo se decidió que, salvo en las dos primeras etapas, las palabras a nor-

* Este trabajo ha sido parcialmente financiado por el Ministerio de Economía y Competitividad; proyecto Tacardi (TIN2012-38523-C02-011). Gracias a Josef Novak por su ayuda en el uso de *Phonetisaurus*, y a Mans Hulden por su ayuda con *foma*.

[1] http://komunitatea.elhuyar.org/tweet-norm/

malizar se transforman a minúsculas y que al final del proceso se hace un tratamiento especial de la mayúscula que se describe en la sección 4.3.

### 2.1. Ejemplos aprendidos

El primer transductor ha aprendido las parejas de palabras (pal. no-normalizada, pal. normalizada) anotadas a mano y proporcionadas en el corpus de desarrollo, de modo que si se le pregunta por alguna de ellas, devuelve la respuesta anotada.

Si una misma palabra no-normalizada aparece más de una vez en el corpus se optará por la anotación más frecuente.

### 2.2. Reconocimiento de entidades nombradas

Hay un conjunto importante de entidades nombradas que por su novedad o por otras razones Freeling (Carreras et al., 2004) no etiqueta y deben permanecer sin propuesta de normalización. Se decidió construir un reconocedor de estas entidades basado en la colección no anotada de tweets propuesta por la organización (ver sección 3).

### 2.3. Cambios ortográficos básicos

Basándonos en el conocimiento del idioma y en los ejemplos del corpus de desarrollo hemos obtenido una gramática de cambios básicos mediante la herramienta libre *foma*, que nos permite compilar las reglas en un transductor.

Las reglas incluidas en esta gramática tratan los cambios relativos a tildes, repetición de vocales al final de la palabra (énfasis), pérdida de la letra "d" en participios y cambios fonológicos/ortográficos frecuentes (*qu*→*q*, *qu*→*k*, *h*→*0*, *b*→*v*, *ll*→*y*...). Estas reglas se componen con un léxico obtenido procesando mediante Freeling la ya mencionada colección de tweets además de la lista propia de Freeling.

Como la mayoría de los transductores siguientes, este transductor tiene una característica reseñable: en caso de obtener más de una respuesta se escoge la más frecuente (modelo de lenguaje de unigramas).

### 2.4. Palabras adyacentes unidas

En el corpus de desarrollo aparecen varios ejemplos de palabras adyacentes que aparecen unidas y que deben ser normalizadas. Por ejemplo *Alomejor* o *lasombradelolivo*. El tratamiento de este tipo de palabras se ha llevado a cabo concatenando el léxico y añadiendo ciertas restricciones para evitar sobregeneración extrema; principalmente, limitando la aparición de palabras cortas en la combinación.

A falta de un corpus más extenso para este tipo de variantes, la gramática parece reconocer bastante bien estos casos, pero tiende a corregir como multipalabra palabras que son otro tipo de variantes. Para limitar este problema se ha realizado un ajuste basado en frecuencias (ver sección 3).

### 2.5. Onomatopeyas

También se deben normalizar las onomatopeyas que aparecen frecuentemente en los tweets. Por ejemplo *jajajaja* o *uhuhuh*. Para ello se ha generado un pequeño léxico de sílabas habituales en las onomatopeyas y una gramática simple que permite la repetición de las mismas. Esta gramática se basa en la solución al conocido fenómeno morfológico de reduplicación (Beesley y Karttunen, 2000). Finalmente ha habido que añadir mayor flexibilidad ya que a menudo las repeticiones no son perfectas. Por ejemplo *jajajajaa*.

### 2.6. WFST: Modelo de canal ruidoso (*noisy-channel*)

Para obtener un transductor siguiendo este modelo habitual en reconocimiento de voz se ha utilizado la herramienta *Phonetisaurus*[2]. Esta herramienta *open source* desarrollada por J. Novak permite crear de manera sencilla sistemas de conversión grafema/fonema (G2P) o fonema/grafema (P2G) basándose en autómatas de estados finitos con pesos o WFSTs (Novak, Minematsu, y Hirose, 2012). En la tarea de normalización de tweets, sin embargo, se ha utilizado para construir un sistema de conversión grafema/grafema, siguiendo los pasos que la herramienta exige.

Aunque no hay espacio para describir en detalle la herramienta, merece la pena comentar la simplicidad de su uso, que consta básicamente de dos pasos: (1) alinear las palabras para crear un diccionario de entrenamiento; (2) entrenar un modelo en base a pares de palabras ya conocidas que posteriormente permitirá dar respuesta a nuevas palabras.

Las respuestas del sistema pueden ser palabras no normalizadas que deben ser elimi-

[2]http://code.google.com/p/phonetisaurus/

nadas. Para esto se usa el léxico descrito en la sección 3 y que se utiliza en varios de los transductores.

En cualquier caso la primera palabra normalizada propuesta por G2G puede no ser la palabra más frecuente y sería interesante combinar la probabilidad de transiciones calculada por G2G con la frecuencia. Esto queda como tarea pendiente.

### 2.7. Distancia de edición y cambios complejos

Una primera aproximación que puede verse como un baseline es obtener las palabras que están a la mínima distancia de edición y elegir la más frecuente. Esto se realiza fácilmente usando la opción *med* de la citada herramienta *foma*.

Otra opción que hemos explorado es enriquecer y flexibilizar la gramática de cambios básicos 2.3 añadiendo fenómenos morfofonológicos más complejos (p. ej. la pérdida de caracteres, principalmente vocales) y flexibilizando la aplicación de las reglas para permitir varios cambios en la misma palabra. Compilando estas reglas y componiéndolas con el léxico como se hace en la gramática referenciada en 2.3, se obtiene un transductor de mayor tamaño capaz de corregir variantes más alejadas de la forma normalizada.

## *3. Recursos externos*

Para la realización de nuestro sistema se han utilizado varios recursos externos basados en un amplio corpus de tweets y en un buscador.

### 3.1. Corpus de tweets

Se han recuperado los más de 200.000 tweets identificados por la organización y se han procesado mediante Freeling. Las palabras reconocidas son almacenadas consiguiendo así un diccionario de frecuencias en el dominio. Para solucionar la falta de cobertura de palabras poco frecuentes, son añadidas con frecuencia 1 las entradas del diccionario de Freeling. Con estos datos se genera el léxico, con sus frecuencias, que es utilizado en varias de las etapas descritas en la sección 2.

Las palabras no reconocidas se pueden considerar como variantes habituales en el dominio. Fueron examinadas y se llegó a la conclusión de que la mayoría eran entidades. Tras un repaso y selección de las que tenían una frecuencia mayor que 10 se construyó el diccionario de entidades nombradas que se usa en la segunda etapa del sistema (ver apartado 2.2).

### 3.2. Buscador *Bing*

A la hora de determinar la posibilidad de proponer varias palabras adyacentes como forma de normalizar un caso (sección 2.4), es importante evitar falsos positivos. Aunque se dispone de la probabilidad de cada palabra se decidió que era necesario conocer la frecuencia de la colocación, para lo que se ha utilizado la API del buscador *Bing*[3]. Se debe tener en cuenta que la probabilidad de los términos multipalabra es baja y muchas veces es cero en el corpus de tweets utilizado, por lo que tras una evaluación cualitativa se decidió basar la decisón en el número de apariciones en un buscador.

## *4. Ajustes y pruebas*

Las gramáticas se fueron ajustando en base a los ejemplos del corpus de desarrollo, pero siempre tendiendo a generalizar las reglas en base a morfofonología. También se ajustaron los umbrales de frecuencia para el módulo de palabras adyacentes unidas mencionado en 2.4.

Por otro lado, debido a que el módulo de ejemplos aprendidos (2.1) y el módulo WFST (2.6) se basan en aprendizaje, y que el corpus de desarrollo de los 500 tweets no es muy grande (se cuenta con un total de 775 pares de palabras anotadas), para testear y ajustar el sistema en desarrollo se ha utilizado *cross-validation*, dividiendo los datos en 5 carpetas.

El principal objetivo perseguido en las pruebas realizadas ha sido dar con la combinación idónea de las respuestas obtenidas en cascada tratando de conseguir la unión de todas las respuestas correctas. Las pruebas realizadas y los resultados obtenidos se describen a continuación.

### 4.1. Comparación de los 3 subsistemas

La primera evaluación midió los resultados de unir las respuestas de las 5 primeras etapas con cada una de las respuestas obtenidas por los tres transductores que resuelven casos más complejos y que se han descrito en 2.6 y 2.7. Los resultados se reflejan en la tabla 1. Tal y como puede observarse el mejor resultado lo obtiene la concatenación con el

[3]http://www.bing.com/developers/s/APIBasics.html

transductor PHON (descrito en 2.6), seguido muy de cerca por el transductor foma más complejo (RULES) y quedando en último lugar, como preveíamos, el transductor MED (los dos últimos se describen en 2.7).

| Transductor | Precisión |
|---|---|
| PHON | 66,32 |
| RULES | 65,94 |
| MED | 62,19 |

Cuadro 1: Precisión obtenida con cada transductor de casos más complejos.

### 4.2. Combinación de las respuestas

Combinar las respuestas puede mejorar los resultados ya que hay diferencias entre ellas. Pero no es evidente cómo hacer la combinación. Primero probamos a combinar los 3 transductores mediante un sistema de votación simple en el que se establece una prioridad entre los transductores para los casos de empate. Se han hecho dos pruebas variando la prioridad entre ellos y los resultados obtenidos son los que refleja la tabla 2. Tal y como puede apreciarse, la votación obtiene peores resultados que antes, excepto en el caso del transductor MED.

Se han vuelto a analizar las diferencias de los resultados de los transductores PHON y RULES y al ver que cada uno resuelve casos que el otro no, se ha tratado de afinar más la combinación de respuestas de los mismos. Así, se ha hecho que el transductor PHON no proporcione una única respuesta sino tres posibles (elegidas según diferencias entre probabilides y teniendo en cuenta además la frecuencia de cada una) y después se ha intentado combinar esas respuestas con las de los otros dos transductores mediante un sistema de votación algo más complejo. Los resultados, sin embargo, no han mejorado. Esta combinación queda como trabajo futuro.

| Votación | Precisión |
|---|---|
| PHON-RULES-MED | 63,48 |
| RULES-PHON-MED | 63,74 |

Cuadro 2: Precisión obtenida estableciendo un sistema de votación simple entre los 3 transductores de casos más complejos.

A la vista de los resultados, y dado que las diferencias entre los dos primeros casos de la tabla 1 son muy pequeños, se han enviado dos resultados del test para ser evaluados: uno obtenido utilizando en la etapa final el transductor PHON y el otro utilizando el transductor RULES.

### 4.3. Mayúsculas

Otro aspecto que se ha tratado de ajustar en el proceso de pruebas ha sido el tratamiento de las letras mayúsculas y minúsculas, puesto que acertar en ese aspecto puede mejorar los resultados notablemente. Excepto los transductores de las dos primeras etapas, el resto trabajan siempre en minúsculas. Para decidir finalmente si una palabra debe comenzar con mayúscula o no se han probado dos estrategias sencillas: (1) la respuesta se proporciona en mayúscula si la entrada comienza con mayúscula; (2) comienzan por mayúscula aquellas respuestas que corresponden a una palabra no-normalizada que es comienzo de un tweet. Las pruebas realizadas con el corpus de desarrollo son algo mejores para la segunda opción, que finalmente se ha seguido al realizar el test.

## *5. Resultados de test y mejoras posibles*

Los resultados obtenidos sobre el test se reflejan en la tabla 3, logrando el quinto lugar entre los 13 sistemas presentados.

| Sistema | Precisión |
|---|---|
| PHON | 61,9 |
| RULES | 60,9 |

Cuadro 3: Precisión sobre el corpus de test.

Teniendo en cuenta que es un sistema formado por componentes muy simples, creemos que es un resultado satisfactorio.

En relación con las mejoras posibles, lo cierto es que a pesar de las pruebas realizadas no se ha conseguido acertar con la combinación final idónea que aúne las respuestas correctas de los transductores finales creados para la resolución de los casos más complejos. Esta es la próxima tarea que debemos afrontar.

Adicionalmente deberíamos afinar más en decidir cuándo no se debe modificar la palabra. En el artículo de referencia para el inglés (Han y Baldwin, 2011) esto se resuelve mediante un clasificador.

## *Bibliografía*


Alegria, Iñaki, Izaskun Etxeberria, y Igor Leturia. 2010. Errores ortográficos y de competencia en textos de la web en euskera. *Procesamiento del lenguaje natural*, 45:137–144.

Beesley, K. R y L. Karttunen. 2002. Finite-state morphology: Xerox tools and techniques. *Studies in Natural Language Processing. Cambridge University Press*.

Beesley, Kenneth R y Lauri Karttunen. 2000. Finite-state non-concatenative morphotactics. En *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, páginas 191–198. Association for Computational Linguistics.

Carreras, Xavier, Isaac Chao, Lluis Padró, y Muntsa Padró. 2004. Freeling: An open-source suite of language analyzers. En *LREC*.

Han, Bo y Timothy Baldwin. 2011. Lexical normalisation of short text messages: Makn sens a# twitter. En *ACL*, páginas 368–378.

Hulden, Mans. 2009. Foma: a finite-state compiler and library. En *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics: Demonstrations Session*, páginas 29–32, Athens, Greece. Association for Computational Linguistics.

Hulden, Mans, Iñaki Alegria, Izaskun Etxeberria, y Montse Maritxalar. 2011. Learning word-level dialectal variation as phonological replacement rules using a limited parallel corpus. En *Proceedings of the First Workshop on Algorithms and Resources for Modelling of Dialects and Language Varieties*, DIALECTS '11, páginas 39–48, Stroudsburg, PA, USA. Association for Computational Linguistics.

Novak, Josef R., Nobuaki Minematsu, y Keikichi Hirose. 2012. WFST-based grapheme-to-phoneme conversion: Open source tools for alignment, model-building and decoding. En *Proceedings of the 10th International Workshop on Finite State Methods and Natural Language Processing*, páginas 45–49, Donostia–San Sebastian, July. Association for Computational Linguistics.