# POS Tagger for Spanish

**Carlos Miguel Gómez Gracia** **Héctor Yela Reneses**

A continuación vamos a mostrar el trabajo obtenido a partir de la realización del citado proyecto en la Universidad de Lund (Suecia) la asignatura de Language Processing and Computational Linguistics (EDA 171).

## 1. Introducción:

En primer lugar, vamos a dar la idea general sobre la que se ha desarrollado nuestro proyecto. Para hacer eso, lo primero que debemos decir, es que un llamado POS Tagger es un programa cuya principal función es la de extraer información de un Corpus (texto de gran dimensión en el que cada palabra presenta información adicional de su estructura) para la realización de unas estadísticas que podrá utilizar en nuevas aplicaciones.

En nuestro caso, dichas estadísticas han sido usadas para encontrar las posibles etiquetas de cada palabra en nuevos textos (en nuestro caso, las etiquetas serán las distintas categorías gramaticales) y elegir la más adecuada en cada caso.

Gracias a esta herramienta, mediante un entrenamiento adecuado de nuestra base de datos, deberíamos ser capaces de etiquetar de forma adecuada cualquier frase que pretendamos evaluar. Ese entrenamiento será después clave en nuestro proyecto, porque podremos observar como difieren bastante los porcentajes de éxito si el fragmento de texto que se pretende etiquetar pertenece o no al Corpus que ha servido para entrenar.

Una vez dadas estas pinceladas que dan una idea general del propósito de nuestro proyecto, pasamos a explicar de un modo más detallado el mismo en los siguientes apartados.

## 2. Desarrollo del proyecto:

Nosotros hemos tratado, en primer lugar, nuestro Corpus. Éste procede del departamento de lenguaje natural de la UPC (Universidad Politécnica de Cataluña).

La estructura del mismo constaba de una palabra por línea en la que aparecían de modo sucesivo la propia palabra original del texto, la palabra raíz de la que procede la misma y un conjunto de etiquetas en el que se podían identificar algunas de las propiedades de las palabras, como son su categoría gramatical, el género al que pertenecen o su número. En el caso de que la palabra fuese una forma verbal, también podíamos identificar su propio tiempo verbal. Algunos ejemplos de esta forma original del Corpus que estamos explicando, podrían ser:

| | | |
|---|---|---|
| No | No | RN |
| Quiero | querer | VMIP1S0 |
| Decir | decir | VMN0000 |

En cualquier caso, para el propósito que le hemos asignado a este proyecto, nos ha sido suficiente con utilizar solamente las primera letra de cada una de estas etiquetas originales, que es la que nos indica la categoría gramatical a la que pertenece la palabra presente en el texto. De este modo, hemos podido distinguir los diferentes tipos de palabras que, según su etiqueta, nos encontraremos a la hora de hacer el análisis. Han quedado como sigue:

| | |
|---|---|
| A | Adjetivo |
| S | Preposición |
| N | Nombre |
| V | Verbo |
| R | Adverbio |
| D | Determinante |
| Y | Abreviatura |
| I | Interjección |
| P | Pronombre |
| Z | Números |
| C | Conjunción |
| F | Puntuación |
| X | Desconocido |

| W | Fechas |
|---|---|

Una vez que fue definida con precisión la parte del Corpus sobre la que íbamos a trabajar, se procedió a recoger datos que resultaran de interés para los posteriores cálculos de probabilidades. De este modo, fueron cuatro los elementos que decidimos almacenar en nuestra base de datos:

- C(w): Número de ocurrencias de la palabra w.
- C(t): Número de ocurrencias de la etiqueta t.
- C(w, t): Número de ocurrencias de la palabra w etiquetada con t.
- C(t1, t2): Número de ocurrencias del bigrama de etiquetas (t1, t2), que consiste en la aparición de una palabra etiquetada con t1, seguida de otra etiquetada con t2.

Algunos ejemplos del cálculo del número de palabras en el Corpus son estos:

| 3 corral |
|---|
| 5 corre |
| 3 correcto |
| 1 corrector |
| 10 corredores |
| 3 corren |
| 3 correr |
| 1 correrse |
| 9 corresponde |

También mostramos a continuación un ejemplo de la distribución de las etiquetas contabilizadas en un fragmento del Corpus, donde se puede ver como algunas de ellas aparecen muchas mas veces que otras.

| A 7886 |
|---|
| C 6660 |
| N 22254 |
| P 6249 |
| R 5514 |
| S 14216 |
| V 13795 |
| W 207 |
| Z 233 |

Una vez que fuimos capaces de realizar correctamente la contabilización de palabras y etiquetas, procedimos a aplicar alguna de las fórmulas que nos habían sido proporcionadas durante el curso para el cálculo de diversas probabilidades. De este modo, y debido a que tras una reunión con el profesor que ejercía como tutor de nuestro proyecto decidimos usar el algoritmo de Viterbi, las probabilidades que calculamos fueron las siguientes:

- Probabilidad de transición:
  $P(ti \mid ti\text{-}1) = C(ti\text{-}1, ti) / C(ti\text{-}1)$
- Probabilidad de estado para palabras conocidas:
  $P(wi \mid ti) = C(wi, ti) / C(ti)$
- Probabilidad de estado para palabras desconocidas:
  $P(wi \mid ti) = 1$

Este algoritmo de Viterbi del que hemos hablado, determina el camino subóptimo que debe seguir para cada nodo en el autómata, descartando el resto de nodos, mientras lo atraviesa. Esta autómata es el que se construye teniendo en cuenta las posibles etiquetas que pueden presentar las palabras ambiguas del texto.

En el proyecto desarrollado, la probabilidad de transición que hemos usado se ha calculado mediante bigramas, por lo que nuestro porcentaje de acierto cuando aparecen palabras desconocidas ronda el 80%. En el caso de que hubiésemos usado trigramas para su cálculo, este resultado hubiese mejorado hasta el 90% aproximadamente.

Además de los cálculos de los que he hablado, para elevar el porcentaje de acierto al etiquetar las palabras, hemos usado también un conjunto de simples reglas que en castellano se pueden aplicar de un modo sencillo, pero que serían muy difíciles de identificar en el tipo de análisis que habíamos desarrollado. Unos de los casos más complejos de este tipo que hemos encontrado es el de la palabra “que”. Es un caso muy común su uso en cualquier texto en castellano y no resulta nada fácil distinguir con simples estadísticas cuando debemos identificarlo como un pronombre o como un determinante. Por ello, al hacer un análisis de cuáles eran las palabras en las que más errores se producían, pudimos ver que ésta era con bastante diferencia la que más problemas daba. Aún así, y a pesar de la introducción de estas reglas todavía sería necesario hacer un análisis gramatical mas profundo para su correcto etiquetado.

## 3. Análisis del código

Básicamente lo que hace nuestro código es analizar el corpus adquiriendo estadísticas, uno de los programas analiza todo el corpus mientras que el otro omite analizar la parte que luego usaremos como comprobante.

Una vez obtenidas estas estadísticas, que luego servirán para calcular probabilidades, utilizaremos el algoritmo de Viterbi para ir etiquetando las palabras con su correspondiente etiqueta.

Finalmente comprobamos con la parte del texto que hemos elegido, el porcentaje de acierto y, en el caso de que existan palabras desconocidas, calculamos el porcentaje para los dos tipos de palabras.

Todo este código del que estamos hablando ha sido estructurado de un modo claro y ordenado, de modo que cualquier posible ampliación que se desee realizar sobre el mismo no precise mas que un simple vistazo a los comentarios que en cada fragmento del mismo hemos situado. Este hecho también nos ha sido de gran ayuda durante la realización del proyecto, para tener claro en todo momento el lugar en el que realizábamos cada operación sobre el Corpus.

Sirva como ejemplo de la adecuada estructura del código alguna de sus partes:

```
($ini, $end) = @ARGV;
@mytags = ('A', 'R', 'D', 'N', 'V', 'I', 'Y', 'S', 'P', 'Z', 'C', 'F', 'X', 'W');
open(FILE, "corp.txt") || die "Could not open file corp.txt";
$cnt = 0;
$auxiliar = ""; #fragmento con las palabras sin etiquetar
$comprobante = ""; #fragmento para comprobar tags


#COMPROBAMOS QUE LOS ARGUMENTOS SEAN CORRECTOS
if($ini < 0 || $end >= 106124 || $ini >= $end){
	print "There is a wrong with the arguments\n";
}

else{
	#RECORREMOS EL FICHERO, OBTENEMOS EL TROZO A ANALIZAR Y
LLENAMOS LA BD
	while ($line = <FILE>){
		@linea = split(/ /, $line);

		#CAMBIAMOS LAS MAYUSCULAS POR MINUSCULAS EN LA PALABRA
		$linea[0] =~ tr/A-ZÅÀÂÄÆÇÉÈÊËÎÏÔÖÙÛÜ/a-
zåàâäæçéèêëîïôößùûüÿ/;

		#OBTENEMOS EL TAG DE LA PALABRA
		@palabra = split(/ */, $linea[2]);

		#OBTENEMOS LA FRECUENCIA DE CADA PALABRA
		if(!exists($words{$linea[0]})){
			$words{$linea[0]} = 1;
		}
		else{
			$words{$linea[0]}++;
		}

		#OBTENEMOS LA FRECUENCIA DE CADA TAG
		if(!exists($tags{$palabra[0]})){
			$tags{$palabra[0]} = 1;
```

```
		}
		else{
			$tags{$palabra[0]}++;
		}

		#OBTENEMOS LA FRECUENCIA DE LA DUPLA PALABRA TAG
		if(!exists($wordandtag{"$linea[0] $palabra[0]"})){
			$wordandtag{"$linea[0] $palabra[0]"} = 1;
		}
		else{
			$wordandtag{"$linea[0] $palabra[0]"}++;
		}

		#OBTENEMOS LAS FRECUENCIAS DE BIGRAMAS DE TAGS
		if($cnt != 0){
			if(!exists($bigramtag{"$tagant $palabra[0]"})){
				$bigramtag{"$tagant $palabra[0]"} = 1;
			}
			else{
				$bigramtag{"$tagant $palabra[0]"}++;
			}
		}

		#OBTENEMOS LAS FRECUENCIAS DE TRIGRAMAS DE TAGS
		if($cnt > 1){
			if(!exists($trigramtag{"$tag2ant $tagant $palabra[0]"})){
				$trigramtag{"$tag2ant $tagant $palabra[0]"} = 1;
			}
			else{
				$trigramtag{"$tag2ant $tagant $palabra[0]"}++;
			}
		}

		#OBTENEMOS LAS FRECUENCIAS DE (W1,T1,T2)
		if($cnt != 0){
			if(!exists($mixgramtag{"$wordant $tagant $palabra[0]"})){
				$mixgramtag{"$wordant $tagant $palabra[0]"} = 1;
			}
			else{
				$mixgramtag{"$wordant $tagant $palabra[0]"}++;
			}
		}

		#LLENAMOS EL FRAGMENTO SI ES NECESARIO
		if($cnt >= $ini && $cnt <= $end){
			$comprobante .= "$palabra[0]\n";
			$auxiliar .= "$linea[0] ";
		}

		#GUARDAMOS LA ULTIMA PALABRA DEL FRAGMENTO PARA SABER SI
ES UN PUNTO
		if($cnt == $end){
			$ultima_palabra = $linea[0];
		}

		$tag2ant = $tagant;
		$tagant = $palabra[0];
```

```
		$wordant = $linea[0];
		$cnt = $cnt + 1;
	}


	#METEMOS EN WORDANDTAG PARA CADA PALABRA LA LISTA DE TAGS POSIBLES
	#CUIDADO LUEGO CON LAS PALABRAS SIN TAG; LAS UNKNOW!!!!!!!
	foreach $word (sort keys %words){
		$lineatags = "";
		for ($j = 0; $j <= $#mytags; $j++){
			if(exists($wordandtag{"$word $mytags[$j]"})){
				$lineatags .= "$mytags[$j] ";
			}
		}
		$tagsforword{$word} = $lineatags;
	}
```

## 4. Resultados

Una vez explicado como hemos calculado las diversas probabilidades para realizar un etiquetado correcto y el algoritmo usado, ya podemos explicar que han sido 2 los programas que hemos elaborado en nuestro proyecto.

Por una parte, el primero lo que hace es introducir todo el Corpus en la base de datos para, posteriormente, proceder a analizar una parte del mismo en el que no se van a encontrar palabras desconocidas. Es el fichero llamado POSinc.pl y los resultados obtenidos por él se acercan al 98% de etiquetados correctos. Para invocar a dicho programa es necesario hacerlo del siguiente modo:

*perl –w POSinc.pl lineIn lineOut > exit.txt*

En el otro programa llevado a cabo, el texto es entrenado por una amplia parte del Corpus para después analizar la parte restante, apareciendo de este modo palabras desconocidas. El porcentaje de éxito roza el 80% y esta en el fichero llamado POSdestex.pl. La manera de invocar este programa es:

*perl –w POSdestex.pl lineIn lineOut >exit2.txt*

Este último además nos devuelve el porcentaje de aciertos de las palabras conocidas y el de las desconocidas por el trainer. Para las palabras conocidas el porcentaje ronda el 88% y para las desconocidas el 25%.

## 5. Conclusiones y posibles mejoras

De nuestro experimento y de los resultados obtenidos podemos deducir que un POS que funciona con bigramas y sin casi tratamiento para las palabras desconocidas es una buena herramienta pero no óptima, ya que con un tratamiento sobre esas palabras desconocidas de las que solo acertamos una cuarta parte ahora y con una leve mejoría sobre las conocidas (posiblemente con trigramas, aunque se podria dar el caso que los porcentajes no mejorasen acordes con el esfuerzo y la complejidad requerida para utilizarlos) podríamos aumentar nuestra efectividad quizás hasta un 90%.

Todo y así necesitaríamos de un corpus mayor para conocer el alcance de nuestros resultados y poder trabajar en una posible mejora, habría también que comparar los resultados de nuestro proyecto empleado sobre otras lenguas, ya que quizás el castellano es una lengua más sencilla de Taggear que el inglés.

En cualquier caso, la conclusión que nosotros podemos sacar tanto de la realización del proyecto en sí, como del conjunto de la asignatura, es muy positiva, ya que ninguno de los dos habíamos tenido anteriormente experiencia alguna con el lenguaje de programación PERL y en España no se utiliza de un modo común.

De hecho, las herramientas que habíamos podido llegar a usar para tratar textos eran algunas como Lex, Yacc o Visón, y siempre en asignaturas que tenían una estrecha relación con el mundo de los compiladores, nunca con el fin que le hemos aplicado en este proyecto.

## 6. Agradecimientos

Por último, nos gustaría agradecer a CLiC Centre de Llenguatge i Computació (UB) el hecho de habernos permitido la utilización de su extraordinario Corpus de aprendizaje en español, ya que sin él, no habría habido proyecto, ni resultados ni nada.

Por supuesto, también debemos acordarnos y dar las gracias tanto a la persona que se ha encargado de tutorarnos este proyecto, Richard Johansson, como al profesor encargado de las clases de teoría, Pierre Nugues. Nos han ayudado a sacar adelante esta tarea, especialmente en la parte que nos resultaba más nueva para nuestros conocimientos, todo lo relacionado con el algoritmo de Viterbi. También agradecer a aquella gente de la Universidad de Lund que permite tantas facilidades a los alumnos para trabajar cómodamente, tanto a nivel de medios como de horarios.

Ya para acabar, sólo queda presentarnos de un modo mas formal. Este proyecto ha sido realizado por:

- Carlos Miguel Gómez Gracia, alumno del Centro Politécnico Superior (Zaragoza, España).
- Héctor Yela Reneses, alumno de la Universidad Politécnica de Cataluña (Barcelona, España).