Trabajo final Lingüistica Computacional

Marcos Esteve Casdemunt y David Gimeno Gomez

Octubre 2019

Contents

1 Introducción

Este trabajo final de Lingüística Computacional tiene como objetivo principal evaluar las prestaciones de distintos etiquetadores morfosintácticos presentes en el paquete NLTK (Natural Language Toolkit). Para ello, se procede a realizar una experimentación donde se estudia cómo influyen sobre las prestaciones del sistema parámetros como el tamaño del corpus de entrenamiento o el suavizado para las palabras desconocidas, entre otros. Cabe destacar que todas las precisiones expuestas a lo largo de la memoria se han obtenido mediante una validación cruzada, exceptuando los casos en los que se comente lo contrario, tal y como se ha estudiado en la asignatura. Por último, indicar que se ha empleado el corpus del español cess-esp.

2 Tarea 1

El objetivo principal de esta tarea es evaluar el corpus *cess-esp* utilizando validación cruzada y comparando los resultados obtenidos con el corpus de etiquetas reducidas frente al corpus original. Para ello, emplearemos el etiquetador basado en Hidden Markov Models (HMM) proporcionado en el paquete NLTK.

Precisión	**Intervalo**
0.892963	0.024550
0.895532	0.024289
0.894237	0.024421
0.903238	0.023476
0.897726	0.024062
0.897537	0.024082
0.892107	0.024637
0.898081	0.024025
0.901848	0.023626
0.897949	0.024039

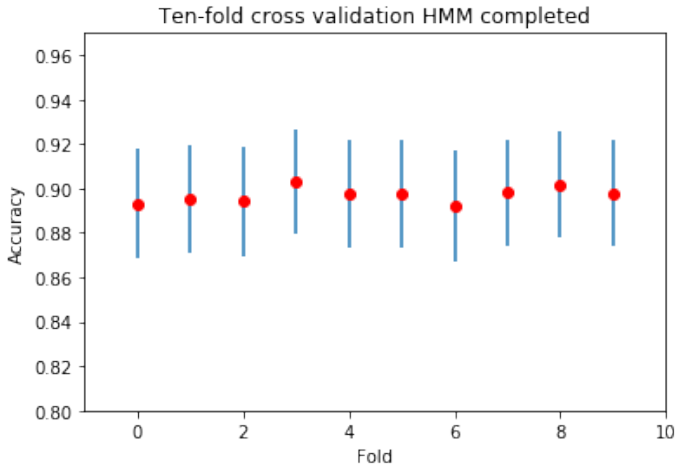

Figure 1: Precisiones con validación cruzada en el conjunto de datos original

Precisión	Intervalo
0.922902	0.021182
0.925816	0.020811
0.927870	0.020544
0.926035	0.020783
0.926081	0.020777
0.922131	0.021279
0.926187	0.020763
0.924996	0.020916
0.929857	0.020280
0.925980	0.020790

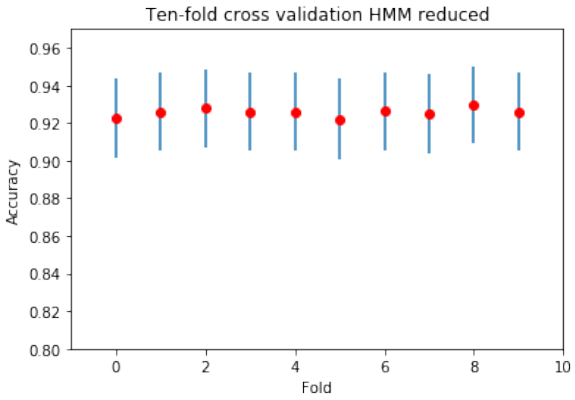

Figure 2: Precisiones con validación cruzada en el conjunto de etiquetas reducido

A la vista de los resultados se observa que mientras el modelo entrenado con el corpus original obtiene una precisión de 89'71%, el modelo basado en categorías reducidas es capaz de alcanzar una precisión de 92'58%. Esto se debe a que al reducir el número de clases a distinguir, el modelo es capaz de adaptarse mejor a los datos obteniendo mejores generalizaciones.

3 Tarea 2

Debido a los resultados en la anterior tarea, de ahora en adelante se ha decido utilizar el corpus reducido dado que genera mejores resultados.

En la tarea actual se ha tratado de evaluar cual es la evolución de la precisión al aumentar el tamaño del conjunto de entrenamiento.

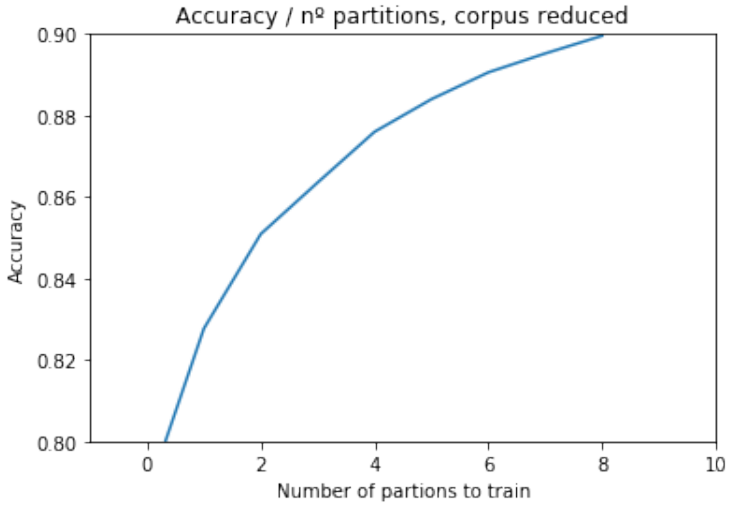

Figure 3: Progresión de la precisión en función de la cantidad de datos

A la vista de los resultados observados en la figura 3 podemos concluir que

la precisión del sistema mejora a medida que aumenta el tamaño del corpus de entrenamiento.

4 Tarea 3

Normalmente, el conjunto de muestras dedicado al entrenamiento no puede contener todo el vocabulario existente en un lenguaje concreto. En otras palabras, nuestro etiquetador tendrá que enfrentarse con palabras que no haya conocido durante su etapa de aprendizaje y, por ello, deberemos solventar este contratiempo mediante técnicas de suavizado que permitan una respuesta por parte del sistema. Debido a que el etiquetador TnT no incorpora este suavizado se ha propuesto el empleo del etiquetador AffixTagger. El suavizado se estimará por el AffixTagger en base al sufijo de la palabra desconocida en cuestión. Sin embargo, es necesario realizar una experimentación para determinar cuál es la longitud del sufijo óptima, es decir, aquella que mejora las prestaciones del etiquetador TnT, tal y como se observa en la figura 4. Cabe destacar que cuando la longitud del sufijo es nula, se está mostrando la precisión del sistema sin ningún tipo de suavizado.

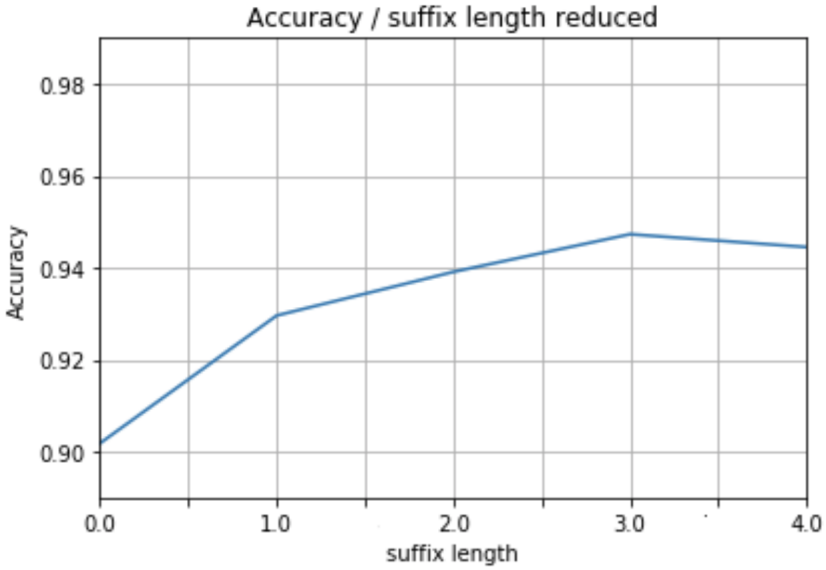

Figure 4: Influencia de la longitud del sufijo sobre la precisión del sistema

A partir de estos resultados, observamos que a medida que aumenta la longitud del sufijo se van mejorando las prestaciones del etiquetador. Sin embargo, una vez sobrepasamos un límite, en este caso la longitud 3, vemos como el sufijo abarca demasiada variabilidad disminuyendo las prestaciones. Por lo tanto, concluimos que la mejor configuración para etiquetador consiste en definir la longitud del sufijo como 3 caracteres, obteniendo así una precisión del 94.73%. Por otra parte, se comprueba que la incorporación de un método de suavizado incrementa la tasa de acierto notablemente.

5 Tarea 4

En esta tarea, se propone la experimentación con otros paradigmas de etiquetado presentes en el paquete NLTK sobre nuestro corpus reducido. En nuestro caso, nos hemos decantado por evaluar el etiquetador de Brill y el de Perceptron. Una vez hayamos obtenido las prestaciones de cada uno de ellos procederemos a comparar estos sistemas con los empleados en anteriores tareas, es decir, el etiquetador TnT y el HMM. Cabe destacar que las evaluaciones se realizan mediante el método Hold-out.

Por otro lado, la técnica de Brill requiere de una inicialización, ya sea por parte de probabilidades estimadas en función de qué etiqueta es más frecuente para una palabra en concreto o mediante otros etiquetadores.

Table 1: Comparación con otros etiquetadores

	Brill Tagger		**Perceptron**	**HMM**	**TNT**
	Unigram Tagger	**HMM**			
Precisión	89.94%	92.99%	96.79%	88.00%	83.00%
Intervalo	2.40	2.04	1.41	2.59	3.00

Concluimos que los métodos evaluados en esta tarea superan de manera significativa las prestaciones de los sistemas evaluados hasta el momento, a excepción del etiquetador de Brill inicialidado mediante Unigram Tagger que presenta un incremento menor.

6 Evaluación de la herramienta Spacy

En esta tarea hemos decido evaluar el paquete Spacy alternativamente al paquete propuesto en el boletín del trabajo. Spacy es una biblioteca de software libre enfocada al procesamiento del lenguaje natural en programas Python. Entre sus utilidades, destacamos Part-of-speech tagging, named entity recognition y la capacidad de emplear Word-embeddings preeentrenados. A diferencia de NLTK, el cual es ampliamente utilizado en educación e investigación, Spacy ocupa lugar en el ámbito industrial.

6.1 Instalación

La instalacion de Spacy consiste básicamente en realizar un comando pip para instalar el paquete en python.

```
pip install -U spacy
```

Una vez instalado Spacy se puede utilizar un modelo preentrenado en distintos lenguajes y con distintas capacidades para utilizarlo en numerosas tareas. También es posible entrenar nuestros propios modelos como se detalla en la documentación [1] . Para utilizar un modelo preentrenado es necesario en primer lugar descargar el modelo. En nuestro caso, descargamos el modelo de capacidad mínima en español proporcionado por la plataforma.

```
python -m spacy download es_core_news_sm
```

[1] https://spacy.io/usage/training

6.2 Uso

Una vez descargado el modelo preentrenado ya se puede utilizar. Para ello importamos Spacy y cargamos el modelo.

```
import spacy
nlp = spacy.load("es_core_news_sm")
```

Una vez instanciado nlp ya podemos realizar tareas de procesamiento lingüístico. Sin embargo es necesario conocer el *pipeline* que sigue la herramienta.

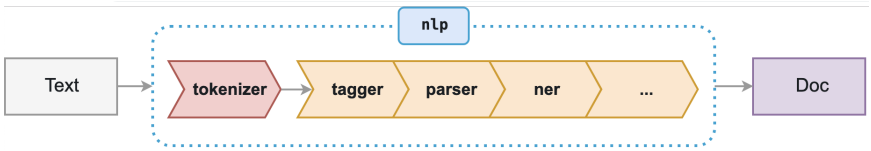

Figure 5: *Pipeline* de Spacy

En la imagen superior se observa el pipeline básico ofrecido por Spacy. Este proceso está caracterizado por un texto de input que pasa por distintas fases como el tokenizador, el tagger para el proceso de POS, el parser, pensado para realizar el análisis sintáctico; y el Name Entity Recognition (NER) para detectar las entidades nombradas. Tras aplicar todas estas fases, obtenemos el texto procesado junto con toda la información lingüística extraída de cada uno de los módulos descritos. Este output se le conoce como documento.

A partir de este documento podemos extraer la información expuesta en los siguientes subapartados. Para ello, nos hemos guiado por la documentación proporcionada en la página web oficial en la que se exponen numerosos ejemplos junto a sus respectivas codificaciones. Tal y como se observará más adelante, el empleo de la biblioteca presenta una gran facilidad de uso, así como una alta funcionalidad.

6.2.1 Tokens

```
#Sacar los tokens
for token in doc:
  print(token.text)
```

```
este
cuento
infantil
.


Ponlo
con
tu
mano
pequeña
y
amable
```

Figure 6: Extracción de Tokens

6.2.2 Part Of Speech

```
for token in doc:
  print(token.text, token.lemma_, token.pos_)
```

```
través través NOUN
de de ADP
la lo DET
tarde tardar NOUN
color color NOUN
de de ADP
oro orar NOUN


      SPACE
el el DET
agua aguar NOUN
nos no PRON
lleva llevar VERB
sin sin ADP
esfuerzo esforzar NOUN
por por ADP
nuestra nuestro DET
parte partir NOUN
, , PUNCT
```

Figure 7: POS tagging

6.2.3 Name Entity Recognition

Con el objetivo de visualizar de manera más clara la funcionalidad del reconocimiento de entidades se ha empleado una noticia de un periódico.

```
from spacy import displacy
import IPython
IPython.display.HTML(displacy.render(doc, style="ent"))
```

En un tuit lanzado a primera hora, Trump PER defendió que Estados Unidos LOC nunca debió haber ido a Oriente Próximo PER ni debe meterse en estúpidas guerras sin fin. Ir a Oriente Próximo MISC es la peor decisión que jamás se ha tomado en la historia de nuestro país, añadió, y ahora estamos lenta y cuidadosamente trayendo a nuestros grandes soldados a casa. El mandatario MISC ha señalado que continúa monitoreando de cerca la situación y que espera que Erdogan PER cumpla sus compromisos. Turquía MISC se ha comprometido a proteger a los civiles, a proteger a las minorías religiosas, incluidos los cristianos, y a asegurarse de que no se produzca una crisis humanitaria, ha explicado. El Consejo de Seguridad de Naciones Unidas ORG , por su parte, anunció que se reunirá este jueves de urgencia para discutir la ofensiva turca en el noreste de Siria LOC , a petición de Francia LOC , Reino Unido LOC , Alemania LOC , Bélgica LOC y Polonia LOC , los cinco países de la Unión Europea ORG (UE ORG) que se sientan actualmente en el organismo.

Figure 8: NER tagging