# POS Tagging
# A neural approach

Ramon Ruiz Dolz
Javier Iranzo Sánchez

25-10-2018

# Índice

# Introducción

- POS Tagging o etiquetado de clases.
- Corpus *cess-esp*:
  - 188650 palabras en español
  - Muestras ya etiquetadas
- Paquete de python *nltk*[1]
- Redes neuronales: Keras [2] + Tensorflow back-end

[1] https://www.nltk.org/index.html
[2] https://keras.io/

# Técnicas clasicas para POS Tagging: Tarea 1

- Comparativa de la repercusión del tamaño del conjunto de etiquetas.
- Conjunto de etiquetas reducido formado por 66 etiquetas.
- Conjunto de etiquetas completo formado por 289 etiquetas.

| **Modelo** | **Categorías** | $\overline{A}$ | **IC 95%** |
|---|---|---|---|
| HMM | Completo | 0.897 | [0.88, 0.914] |
| HMM | Reducido | 0.917 | [0.901, 0.932] |

Table 1: Resultados obtenidos por los modelos basados en HMM con juego de categorías completo y reducido

# Técnicas clasicas para POS Tagging: Tarea 2

- Repercusión del tamaño del set de entrenamiento en accuracy.
- División del corpus en 10 bloques
- Inicialmente set de test y entrenamiento $= 1$ bloque
- En cada iteración el set de entrenamiento $+1$ bloque.

| **Modelo** | **Bloques entrenamiento** | $\overline{A}$ | **IC 95%** |
|---|---|---|---|
| HMM | 1 | 0.815 | [0.793, 0.837] |
| HMM | 2 | 0.854 | [0.834, 0.874] |
| HMM | 3 | 0.872 | [0.853, 0.891] |
| HMM | 4 | 0.883 | [0.865, 0.901] |
| HMM | 5 | 0.892 | [0.875, 0.91] |
| HMM | 6 | 0.900 | [0.883, 0.917] |
| HMM | 7 | 0.904 | [0.887, 0.921] |
| HMM | 8 | 0.909 | [0.893, 0.926] |
| HMM | 9 | 0.914 | [0.898, 0.93] |

Table 2: Resultados obtenidos por los modelos basados en HMM en función del tamaño de entrenamiento

# Técnicas clasicas para POS Tagging: Tarea 3

- Comparativa de la repercusión del suavizado en la *accuracy*.
- Suavizado mediante AffixTagger con longitudes negativas (sufijos).
- Longitudes de sufijos evaluadas 2, 3 y 4.

| **Modelo** | **Suavizado: talla sufijos** | $\overline{A}$ | **IC 95%** |
|---|---|---|---|
| TnT | No | 0.898 | [0.881, 0.915] |
| TnT | 2 | 0.933 | [0.919, 0.947] |
| TnT | 3 | 0.942 | [0.928, 0.955] |
| TnT | 4 | 0.94 | [0.926, 0.953] |

Table 3: Resultados obtenidos por los modelos TnT con distintos suavizados

# Técnicas clasicas para POS Tagging: Tarea 4

- Comparativa del rendimiento de distintos modelos clásicos de POS tagging.
- Brill, CRF y Perceptrón.
- Brill preetiquetado mediante Unigramas y HMM.

| **Modelo** | **Modificación** | $\overline{A}$ | **IC 95%** |
|---|---|---|---|
| HMM | - | 0.917 | [0.901, 0.932] |
| TnT | Suavizado<br>Sufijos talla 3 | 0.942 | [0.928, 0.955] |
| Brill | Preetiquetado<br>HMM | 0.917 | [0.915, 0.919] |
| CRF | - | 0.949 | [0.937, 0.962] |
| Perceptrón | - | 0.961 | [0.951, 0.972] |

Table 4: Resultados obtenidos mediante los modelos de etiquetado clásicos

# POS Tagging con redes neuronales: Feedforward

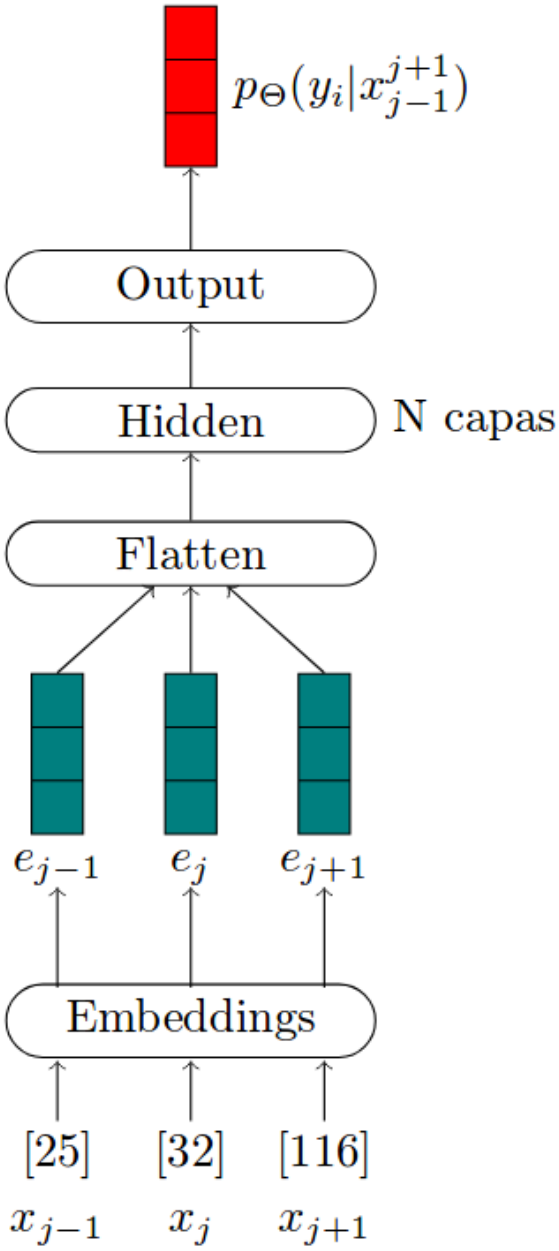
Objetivo: emitir etiquetas $y_1^l$ para la frase $x_1^l$.
Modelo general:

$$\hat{y}_i = \arg\max_{y_i} p_\Theta(y_i | x_1^l, y_1^{i-1}, y_{i+1}^l) \tag{1}$$

Redes feedforward: Relajamos las dependencias.

$$p(y_i | x_1^l, y_1^{i-1}, y_{i+1}^l) := p(y_i | x_{i-window}^{i+window}) \tag{2}$$

# Arquitectura WNN-Tag



# POS Tagging con redes neuronales: Recurrentes

Mantenemos la dependencia en $x_1^I$

$$p(y_i|x_1^I, y_1^{i-1}, y_{i+1}^I) := p(y_i|x_1^I) \tag{3}$$

Longitud variable $\rightarrow$ No podemos aplicar feedforward directamente. Para cada muestra $i$, $c_i$ recoge la información de su dependencia sobre las $x$. De esta manera:

$$p(y_i|x_1^J) = f(c_i; \theta_o) \tag{4}$$

$\theta_o$ es el vector de pesos de la capa de salida.

# POS Tagging con redes neuronales: Recurrentes

¿Como se calculan los $c_i$?
Combinamos la salida de dos capas recurrentes. Una recorre la entrada de izquierda a derecha, la otra al revés.

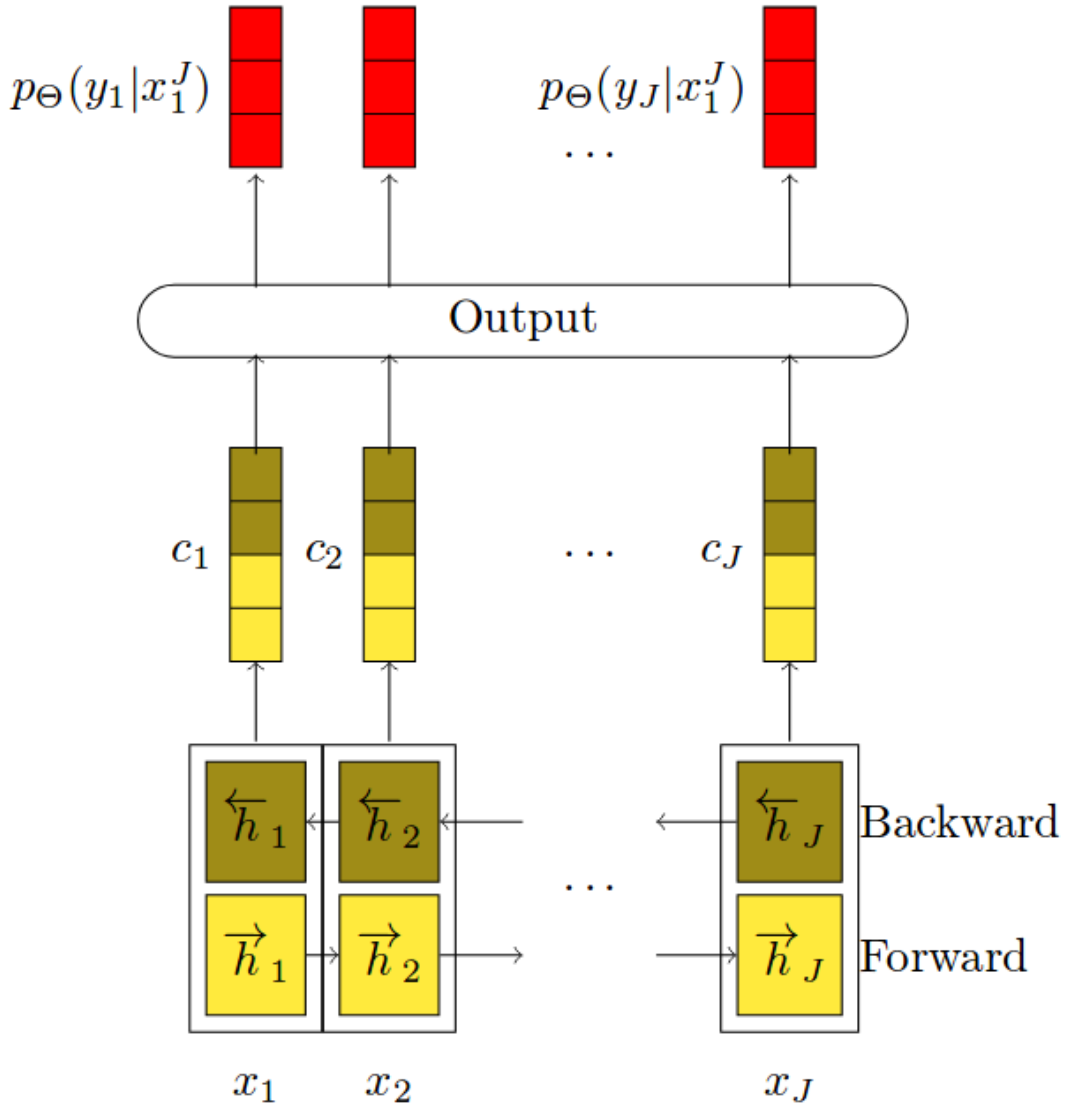
$$\overrightarrow{h_i} = g(\overrightarrow{h}_{i-1}, x_i; \theta_f) \tag{5}$$

$$\overleftarrow{h_i} = g(\overleftarrow{h}_{i+1}, x_i; \theta_b) \tag{6}$$

$$c_i = [\overrightarrow{h_i} \ ; \ \overleftarrow{h_i}] \tag{7}$$

$\theta_f$ es el vector de pesos de la capa forward, $\theta_b$ es el vector de pesos de la capa backward, $g(.)$ la transformación aplicada en cada paso por las capas recurrentes y [ ] es el operador de concatenación.

# Arquitectura BLSTM-Tag



# Resultados

- Embeddings inicializados con vectores glove pre-entrenados
- 20k vocabulario
- Arquitectura
  - WNN-Tag: 200,200,200
  - BLSTM-Tag: 100*2(LSTM)

| Modelo | $\overline{A}$ | IC (95%) |
|---|---|---|
| WNN-Tag | 0.935 | [0.934, 0.936] |
| BLSTM-Tag | 0.938 | [0.937, 0.939] |

Table 5: Resultados obtenidos por los modelos basados en NN

# Conclusiones

- Perceptron obtiene los mejores resultados, pero no hay una diferencia significativa (intervalos de confianza se solapan)
  - Weight averaging
  - Información linguística explícita (sufijos, prefijos)
  - Dependencia en $y_1^{i-1}$
- Muy pocos datos para entrenar las redes neuronales. 6.2M parámetros con solo 6k frases (188k palabras).
  - vs datos otras tareas NLP : 10M-20M frases

# Anexo 1

- **¿Que modelos basados en redes neuronales se han propuesto? ¿Que arquitectura/paradigma sigue cada uno?**
- **¿Cuantas muestras de entrenamiento se generan a partir de una frase del corpus en cada uno de los modelos propuestos?**