POS tagging:
Un análisis de diferentes aproximaciones

Ramon Ruiz Dolz
Javier Iranzo Sánchez

Octubre 2018

Contents

1 Introducción

El etiquetado gramatical o *POS tagging* consiste en asignar una categoría gramatical a cada palabra. Para ello hay que tener en cuenta distintos factores, como por ejemplo el hecho de que una misma palabra pueda pertenecer a distintas categorías gramaticales en función del contexto, o la posibilidad de que aparezcan palabras nuevas en el proceso de etiquetado.

En este trabajo se realiza un estudio y análisis de las diferentes técnicas existentes para afrontar el problema de *POS tagging*. Para aplicar estas técnicas se ha hecho uso del corpus *cess-esp* extraído de la web del paquete *nltk*[1]. Este corpus esta compuesto por 188650 palabras en español analizadas sintácticamente, es decir, ya etiquetadas. Es por esto que, a partir de este corpus se construyen las muestras de test y entrenamiento y, posteriormente se evalúa la precisión del etiquetador.

El trabajo esta organizado de la siguiente forma, en la Sección 2 se ha realizado un análisis de varias técnicas de etiquetado clásicas. Esta sección esta dividida en 4 tareas distintas en las cuales se analizan distintas variables y su influencia en el desempeño de los etiquetadores. En la Sección 3 se ha analizado el funcionamiento de técnicas más avanzadas para afrontar este problema. Concretamente se ha hecho uso de redes neuronales feedforward y recurrentes. Finalmente en la Sección 4 se han comparado las distintas técnicas utilizadas a lo largo del trabajo y su funcionamiento respecto al corpus *cess-esp*.

2 POS Tagging mediante técnicas clásicas

En esta primera sección se han realizado una serie de experimentos haciendo uso de técnicas de *POS tagging* estudiadas a lo largo del curso. La sección esta dividida en 4 tareas distintas. En las dos primeras tareas se ha hecho uso del etiquetador basado en modelos ocultos de Markov. En la tercera tarea se ha utilizado el etiquetador TnT con suavizado basado en sufijos. Finalmente, en la cuarta tarea se ha comparado el desempeño de tres etiquetadores distintos: Brill, CRF y Perceptrón.

2.1 Tarea 1

La primera tarea realizada en este trabajo ha consistido en comparar el efecto de la reducción del conjunto de categorías sobre el proceso de etiquetado. Para ello se ha hecho uso del etiquetador basado en modelos ocultos de Markov. Se han lanzado dos experimentos, uno con el corpus *cess-esp* completo y otro aplicando la reducción de categorías implementada en la sesión anterior. En la gráfica 1 se pueden observar los resultados obtenidos tras realizar la evaluación del modelo mediante validación cruzada. Esta primera gráfica muestra los resultados obtenidos al entrenar el modelo con el conjunto de categorías completo.

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

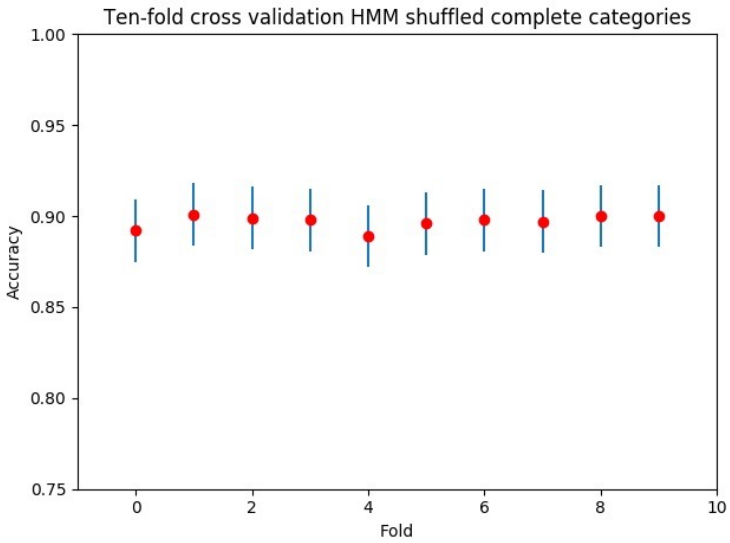

Figure 1: Validación cruzada sobre el corpus barajado mediante el modelo basado en HMM sin reducción de categorías.

Por otra parte, en la gráfica 2 podemos observar los resultados obtenidos al realizar la validación cruzada en el modelo entrenado con el conjunto de categorías reducido.

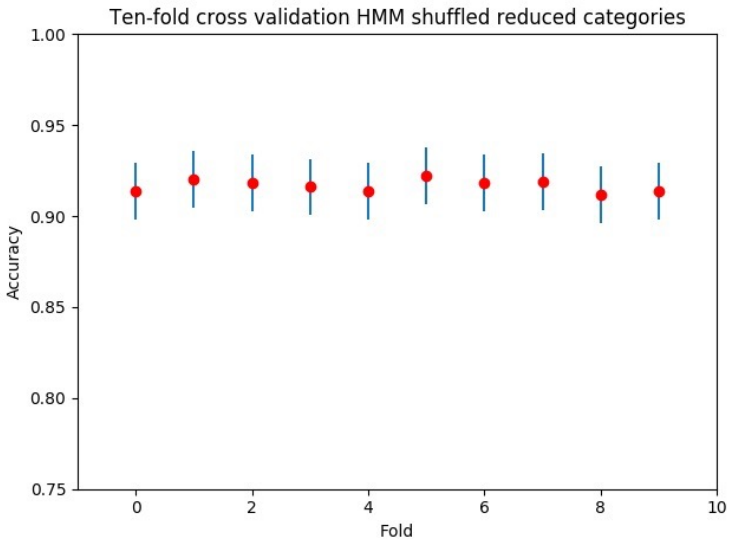

Figure 2: Validación cruzada sobre el corpus barajado mediante el modelo basado en HMM con el juego de categorías reducido.

Como podemos observar, la precisión obtenida al reducir el conjunto de categorías es superior a la precisión alcanzada por el modelo entrenado con el conjunto de categorías completo. En la siguiente tabla se puede apreciar la precisión media (accuracy) de cada modelo entrenado con su intervalo de confianza:

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

2.2 Tarea 2

El objetivo de la segunda tarea realizada mediante técnicas clásicas de *POS tagging* consiste en analizar el efecto del tamaño del conjunto de entrenamiento en la precisión del modelo obtenido. Para realizar este experimento se ha dividido el corpus en 10 bloques del mismo tamaño. Se ha utilizado el clasificado basado en modelos ocultos de Markov. Además, se han realizado 9 entrenamientos distintos, cada uno de ellos aumentando la talla del conjunto de entrenamiento en un bloque hasta alcanzar el tamaño de 9 bloques. Por otra parte, el conjunto de test esta formado por un único bloque de estos 10 bloques totales.

Como se puede observar en la gráfica 3 la precisión obtenida con el mismo

tamaño de entrenamiento y de test a penas se consigue superar el 0.8, conforme el tamaño del conjunto de entrenamiento aumenta la precisión también lo hace. Esto se debe a que el modelo tiene una mayor variedad de muestras para entrenar y, posteriormente comete un menor número de fallos.

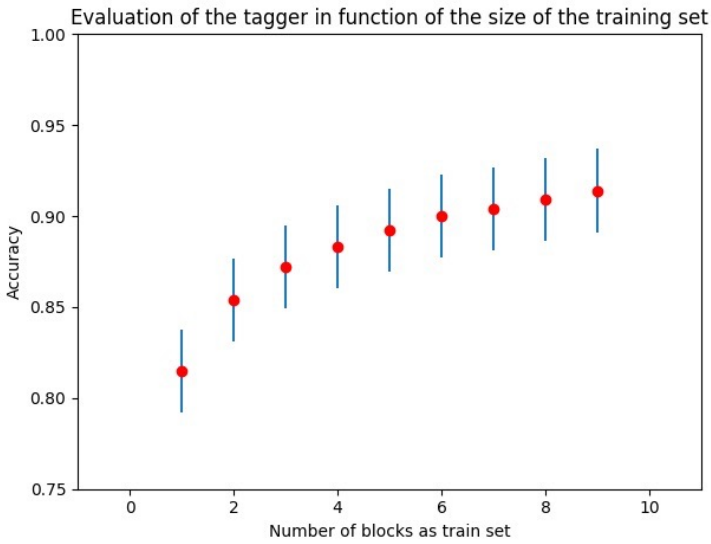

Figure 3: Variación en la precisión del etiquetador en función del tamaño del set de entrenamiento.

En la siguiente tabla se plasma la precisión media y su intervalo de confianza al 95% para cada tamaño de entrenamiento utilizado:

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

2.3 Tarea 3

La tercera tarea consiste en aplicar un método de suavizado basado en sufijos sobre el etiquetador TnT y analizar su repercusión en la precisión. En primer lugar podemos observar la precisión de TnT sin suavizado en la gráfica 4.

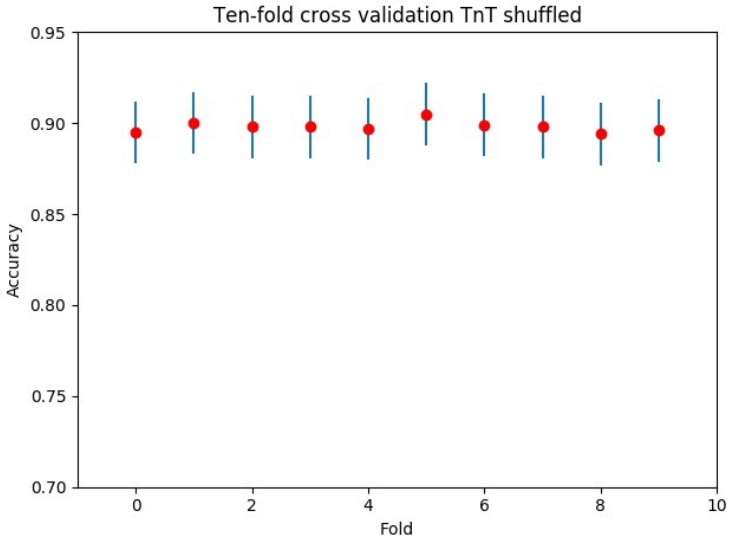

Figure 4: Validación cruzada del etiquetador TnT sin suavizado.

Se ha experimentado con el método de suavizado basado en sufijos para construir un modelo para las palabras desconocidas conocido como *Affix tagger*. Se han realizado tres experimentos distintos modificando la longitud de los sufijos. Se han utilizado sufijos de longitud 2, 3 y 4 obteniendo tres nuevos modelos, en las gráficas 5, 6 y 7 se puede observar el comportamiento de los nuevos modelos respectivamente.

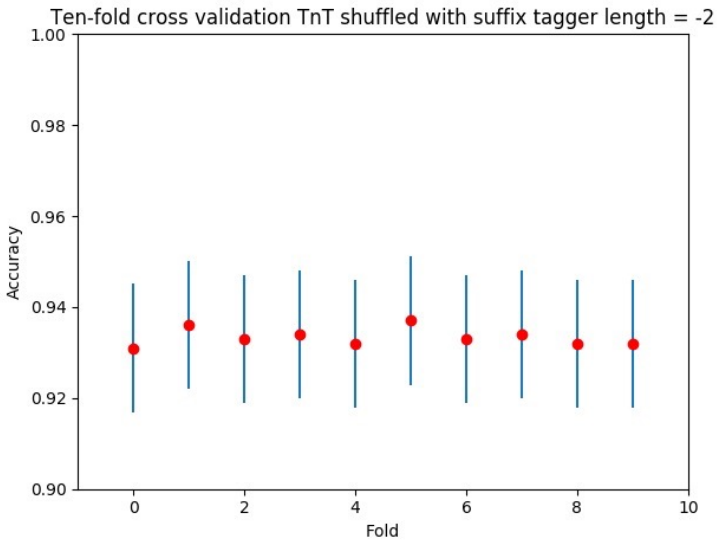

Figure 5: Validación cruzada del etiquetador TnT con suavizado con sufijos de longitud 2.

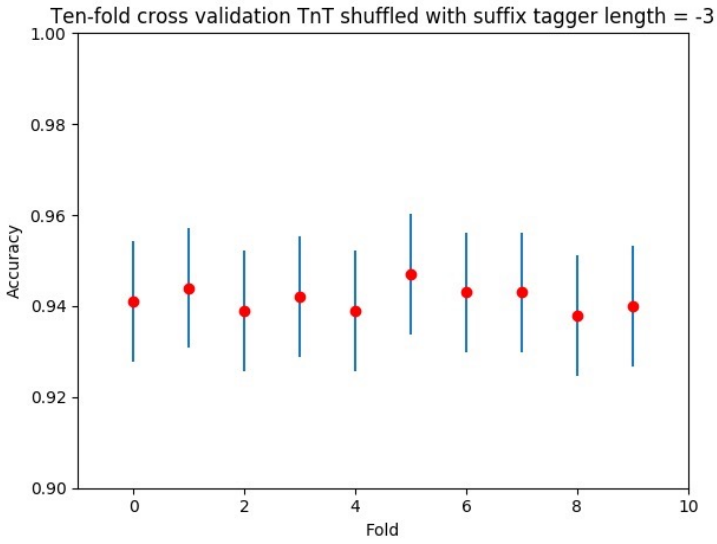

Figure 6: Validación cruzada del etiquetador TnT con suavizado con sufijos de longitud 3.

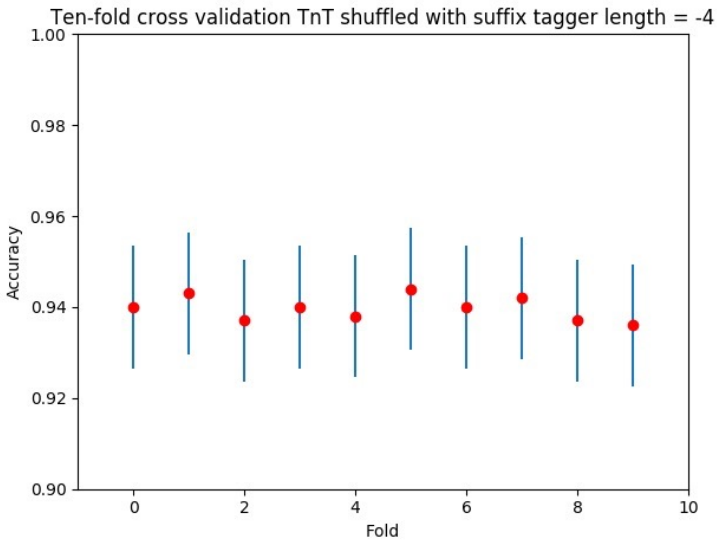

Figure 7: Validación cruzada del etiquetador TnT con suavizado con sufijos de longitud 4.

Mediante este experimento hemos podido observar como los mejores resultados se obtienen con sufijos de longitud 3. La calidad del modelo empeora al aumentar y al decrementar esta talla de sufijos. En la tabla a continuación se puede observar la media de los resultados obtenidos con la distinta talla de los sufijos:

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

2.4 Tarea 4

Finalmente, esta última tarea ha consistido en comparar el funcionamiento de tres paradigmas de etiquetado clásicos, el etiquetador de Brill, el etiquetador CRF y perceptrón. En las siguientes secciones se pueden contemplar los distintos resultados obtenidos con cada uno de estos etiquetadores.

2.4.1 Etiquetador de Brill

El primero de los etiquetadores empleados es el etiquetador de Brill. Este etiquetador es propuesto por el própio Eric Brill en 1992 [3] y consiste en un método inductivo. Este método requiere un etiquetado previo y un set de reglas. El algoritmo hace uso del conjunto de reglas para modificar las etiquetas de las palabras en función de estas con el objetivo de minimizar el error total.

En este trabajo se han lanzado dos ejecuciones distintas. La primera de ellas con el etiquetado inicial realizado mediante el etiquetador basado en unigramas. En la gráfica 8 se puede observar la evaluación del modelo.

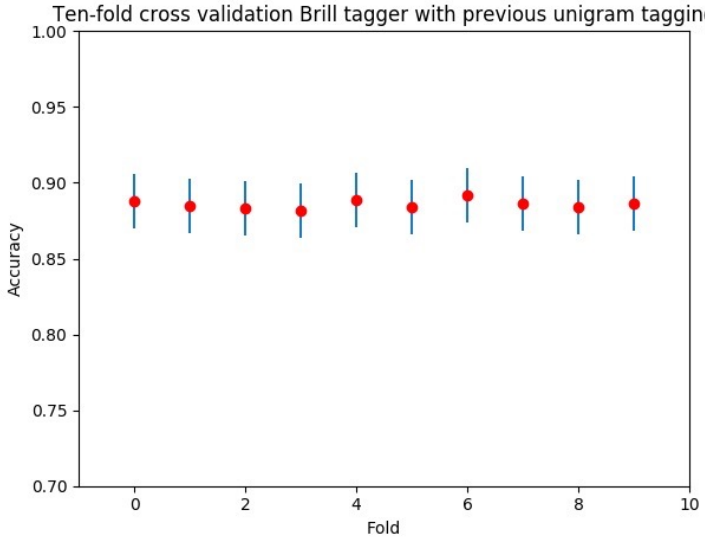

Figure 8: Validación cruzada del etiquetador de Brill realizando el etiquetado inicial mediante unigramas.

Por otra parte, se ha lanzado otro experimento con el etiquetado inicial realizado mediante un etiquetador basado en modelos ocultos de Markov. En la gráfica 9 se pueden observar los resultados obtenidos.

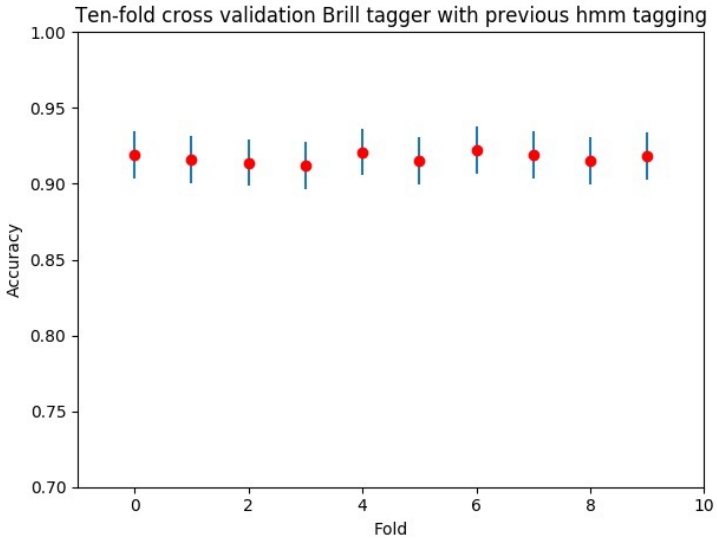

Figure 9: Validación cruzada del etiquetador de Brill realizando el etiquetado inicial mediante HMM.

Cabe destacar, que pese a ser mejores los resultados obtenidos por el etiquetador de Brill preetiquetado con modelos ocultos de Markov, también consume una cantidad de tiempo notablemente superior que el etiquetador basado en unigramas que es prácticamente instantáneo. En la siguiente tabla se puede observar la media de la precisión y su intervalo de confianza obtenidos en el experimento con el etiquetador de Brill:

Modelo	**Preetiquetado**	$\overline{A}$	**IC 95%**
Brill	Unigramas	0.886	[0.884, 0.887]
Brill	HMM	0.917	[0.915, 0.919]

Table 4: Resultados obtenidos por los modelos Brill en función del preetiquetado

2.4.2 Etiquetador CRF

El segundo etiquetador utilizado en esta tarea es el etiquetador basado en *conditional random fields* (CRF) [14]. La base de estos modelos es similar a la de los modelos ocultos de Markov, con la diferencia de que mientras los HMM modelan de forma conjunta la distribución de probabilidad de las etiquetas y las observaciones, el etiquetador CRF modela la distribución de probabilidad de las etiquetas condicionada por las observaciones. En la gráfica 10 se puede apreciar el valor de la precisión evaluando el modelo mediante validación cruzada:

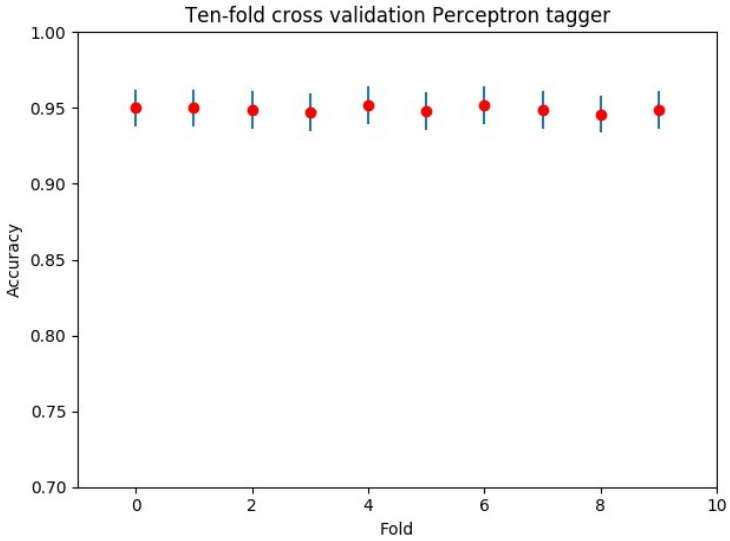

Figure 10: Validación cruzada del etiquetador Conditional Random Field

La media de la precisión obtenida mediante validación cruzada y su intervalo de confianza se ven reflejados en la siguiente tabla:

Modelo	$\overline{A}$	**IC 95%**
CRF	0.949	[0.937, 0.962]

Table 5: Resultados obtenidos por los modelos basados en CRF

2.4.3 Etiquetador perceptrón

El último modelo empleado en esta tarea es el etiquetador basado en perceptrón. El perceptrón [13] consiste en un discriminador lineal con el objetivo de separar las clases linealmente separables. Aplicado a este problema, las clases son las etiquetas. Por lo tanto el perceptron se encarga de encontrar la función discriminante que minimice el número de muestras mal clasificadas, es decir, etiquetadas de forma incorrecta. La información referente al modelo utilizado para construir el etiquetador se describe en [7].

En la gráfica 11 se pueden observar los resultados obtenidos al entrenar el perceptrón y evaluarlo mediante validación cruzada.

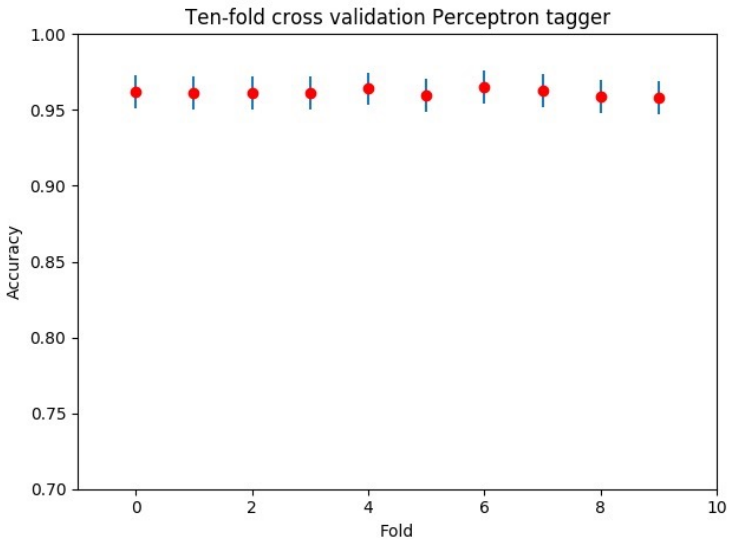

Figure 11: Validación cruzada del etiquetador Perceptrón

Además, en la siguiente tabla se pueden observar los valores relativos a la media de precisión de la validación cruzada y su intervalo de confianza:

Modelo	$\overline{A}$	**IC 95%**
Perceptrón	0.961	[0.951, 0.972]

Table 6: Resultados obtenidos por los modelos basados en Perceptrón

3 POS Tagging con Redes Neuronales

Adicionalmente a las tareas propuestas, también hemos realizado experimentación para comprobar como se comportan los modelos basados en Redes Neuronales. Este apartado se va a centrar en la implementación de modelos para esta tarea y no tanto en una descripción de lo que son las Redes Neuronales en sí. Para una descripción pormenorizada de este tipo de modelos, se recomienda consultar textos como [5].

En las siguientes subsecciones se presentarán los diferentes modelos utilizados para realizar etiquetado. Partimos de que el sistema recibe una frase de entrada $\mathbf{x} = x_1, x_2, \ldots, x_J$, donde x_j es la palabra en la posición j, y el objetivo es producir $\mathbf{y} = y_1, y_2, \ldots, y_I$, donde y_i es la etiqueta que corresponde a la palabra i de la entrada. En su forma más general, se puede asumir que el modelo ideal etiqueta la salida de acuerdo a la siguiente ecuación:

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

Es decir, a la hora de asignar una etiqueta, tiene en cuenta todas las palabras de la frase de entrada, así como el resto de etiquetas de la frase. Esta forma recoge toda la información que puede ser relevante a la hora de asignar una etiqueta, pero es demasiado compleja para que sea de utilidad a la hora de etiquetar. La dependencia sobre y_{i+1}^{I} es especialmente molesta, ya que implicaría que solo podemos etiquetar una palabra si ya hemos etiquetado el resto de la frase.

3.1 Modelos Feedforward basados en ventanas

Ya que la forma general presenta demasiadas dependencias para su cálculo, vamos a relajar estas condiciones y asumir que la etiqueta de una palabra solo dependerá de las palabras de su contexto inmediato. De esta manera, determinaremos la etiqueta a partir de una ventana de (2 x window + 1) palabras.

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

Esta simplificación nos permite afrontar el problema utilizando redes neuronales feedforward. Nuestro modelo estará formado por una capa de embedding, una serie de capas ocultas, y una capa de salida. Mediante un preproceso, se generan J muestras por cada frase x. Las muestras se han construido añadiendo el token "<pad>" para aquellas muestras que lo necesitan al estar cerca del final o el principio de la frase. De esta manera, el proceso de etiquetado consiste simplemente en ejecutar la red para cada muestra, y como salida obtenemos la etiqueta correspondiente a esa muestra.

La única diferencia con un modelo estándar de red neural reside en la capa de embedding. La capa de embedding recibe muestras de dimensiones [(2 x window + 1),1] y emite como salida un representación de forma [(2 x window + 1),dimension embedding]. A continuación esta representación se "aplana" para que adquiera la forma [(2 x window + 1)*dimension embedding,1]. A partir de aquí, la red opera de la misma manera que cualquier otra red, aplicando una serie de transformaciones y emitiendo como salida las probabilidades de las diferentes etiquetas. La arquitectura de este modelo, al que hemos llamado WNN-Tag, se muestra de forma gráfica en la figura 12.

En la literatura encontramos modelos que han utilizado arquitecturas similares para etiquetar, como [12], aunque en ellos la información de las palabras del contexto se suministra como probabilidades condicionales de las etiquetas $p(y_c|x_c)$, estimadadas por Máxima Verosimilitud durante el entrenamiento, en lugar de word embeddings.

3.2 Modelos Recurrentes

La asunción de que la etiqueta de una palabra solo depende de una ventana de tamaño fijo puede resultar insuficiente para capturar algunas relaciones. Si solamente eliminamos la dependencia en las otras etiquetas, la ecuación queda de la siguiente forma:

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

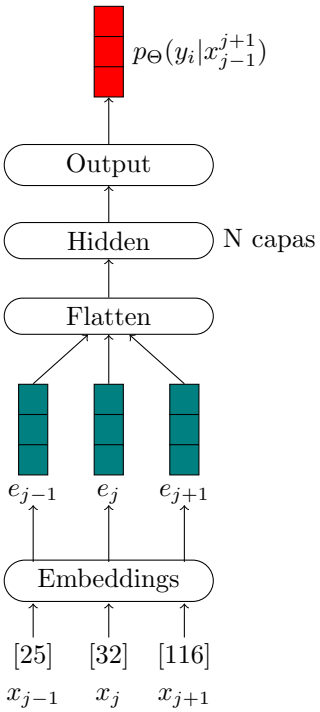

Figure 12: Arquitectura del model WNN. Ejemplo para una red con window=1

Al tener una longitud de entrada variable, ya no podemos aplicar modelos feedforward. Vamos a asumir que, para cada muestra i, la información de la dependencia sobre las x se puede recoger en un vector c_i. Si esto es así, la capa de salida calcula las probabilidades de la manera siguiente:

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

Donde θ_o es el vector de pesos de la capa de salida y $f(.)$ la transformación aplicada por esta capa. Nuestro problema se reduce pues a conseguir estos vectores c_i. Un modelo adecuado para calcular estos vectores son las redes recurrentes.

Los vectores c_i se van a producir combinando los resultados de dos capas recurrentes, una que recorre la entrada de izquierda a derecha, y otra que lo hace de derecha a izquierda.

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

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

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

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

Una vez obtenidos los vectores c, aplicamos la transformación de la capa de salida y obtenemos los vectores de probabilidades. Siguiendo este aproximación, podemos etiquetar una frase entera en una sola activación de la red. La arquitectura de este modelo, al que hemos llamado BLSTM-Tag, se muestra de forma gráfica en la figura 13.

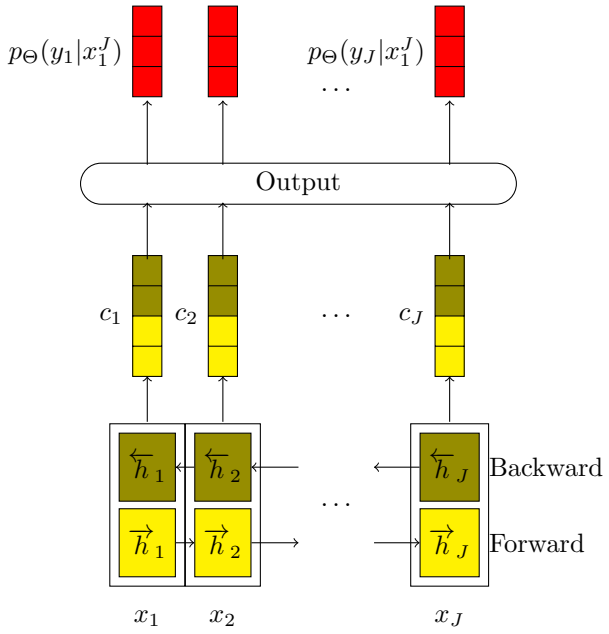

Figure 13: Arquitectura del modelo BLSTM

Existen modelos similares a nuestra propuesta como [16] que incorporan información lingüística adicional de las x como la presencia de mayúsculas/minúsculas.

3.3 Descripción de sistemas

Pasamos ahora a describir la configuración de los sistemas utilizados. Los sistemas presentados aquí han sido implementados utilizando la librería Keras con Tensorflow como back-end. La capa de embeddings se ha inicializado con vectores glove [10] entrenados sobre el corpus SWBC [4], disponibles en [11]. La dimensión de estos embeddings es 300. Todas las palabras se han pasado a minúsculas antes de subministrarlas a la red. Los embeddings de palabras que están presentes en el vocabulario pero para las cuales no hay vector glove disponible se han inicializado con valores de una distribución uniforme [-0.5, 0.5]. En caso de encontrar una palabra que no está incluida en el vocabulario,

Parámetro	WNN-Tag	BLSTM-Tag
Vocabulario	20k	20k
Window	1	-
Optimizador	Adam	Adam
Learning rate	0.01	0.01
Epocas	10	50
Dropout	0.0	0.2
Capas ocultas	[200,200,200]	[100*2 (LSTM)]

Table 7: Configuración de los modelos basados en NN

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

Table 8: Resultados obtenidos por los modelos basados en NN

se sustituye por el token "unk".

En lo que respecta a la evaluación, se ha realizado una validación cruzada en 10 bloques bajo las mismas condiciones que los casos anteriores. Se ha separado un 5% de los datos de entrenamiento como conjunto *dev*. Sobre este conjunto, se ha realizado una búsqueda informal de los hiperparámetros más adecuados, quedando la configuración de los modelos reflejada en la Tabla 7. Aunque estos valores son los que mejor resultados han dado, hay que destacar que la mayoría de hiperparámetros excepto el tamaño del vocabulario tenían efectos pequeños (<1% de accuracy) en el rendimiento del modelo.

Los pesos de las redes se han entrenado con el optimizador Adam [9], con los parámetros propuestos por los autores de la técnica. La función de pérdida aplicada es la entropía cruzada entre la salida del sistema y la muestra de entrenamiento, normalizada por el número de salidas (1 en el caso de WNN-Tag, J en el caso de BLSTM-Tag). Las capas de la red recurrentes están formadas por unidades LSTM [6]. En el caso de la BLSTM, se ha utilizado regularización Dropout [15]. La tabla 8 muestra los resultados obtenidos por estos modelos.

4 Comparativa y conclusiones

Para finalizar el trabajo se han extraído los resultados obtenidos de las mejores versiones de cada modelo etiquetador. En la Tabla 9 se pueden observar estos resultados.

Modelo	**Modificación**	$\overline{A}$	**IC 95%**
HMM	-	0.917	[0.901, 0.932]
TnT	Suavizado Sufijos talla 3	0.942	[0.928, 0.955]
Brill	Preetiquetado HMM	0.917	[0.915, 0.919]
CRF	-	0.949	[0.937, 0.962]
Perceptrón	-	0.961	[0.951, 0.972]
WNN-Tag	-	0.935	[0.934, 0.936]
BLSTM-Tag	-	0.938	[0.937, 0.939]

Table 9: Resultados obtenidos por cada uno de los modelos utilizados en este trabajo en sus versiones con mejor rendimiento

Por lo tanto podemos concluir que, sobre el corpus *cess-esp* el modelo que nos ha permitido alcanzar una mayor precisión en el etiquetado sintáctico de palabras ha sido el Perceptrón. Si bien sus resultados de accuracy son los más altos entre todos los modelos utilizados, la diferencia de rendimiento no es muy grande respecto al resto, y su intervalo de confianza solapa con el de otros etiquetadores como TnT, CRF, WNN-Tag y BLSTM-Tag. Debido a esto, no podemos afirmar que exista una diferencia de rendimiento significativa que nos permita asegurar que es mejor que el resto.

En lo que respecta a los modelos basados en redes neuronales, no han obtenido un rendimiento tan alto como esperábamos. Creemos que hay varias razones que pueden explicar estos resultados. En primer lugar, el número de muestras de entrenamiento es muy reducido comparado con el número de parámetros de los modelos. Contamos con 6030 frases formadas por 188k palabras etiquetadas, mientras que nuestros modelos tienen entre 6.2M y 6.3M de parámetros. Si bien es cierto que cerca de 6M de estos forman la capa de embeddings, preentrenada con los vectores glove, esto todavía nos deja con entre 200k y 300k en el resto de capas de la red, parámetros que son necesarios estimar directamente a partir de los datos. Las redes neuronales han demostrado en situaciones en las que tenemos disponibles muchos datos de entrenamiento que nos permiten entrenar modelos grandes, situación que no se ha dado en este caso, en el cual tenemos un número muy reducido de datos de entrenamiento comparados con las condiciones óptimas para las redes neuronales. Como comparación, en otras tareas de NLP como traducción automática, es habitual utilizar corpus que contienen decenas de millones de frases [2].

En comparación, el modelo de etiquetador Perceptrón, equivalente a una neurona, presenta varias características que vale la pena destacar. En primer lugar, este etiquetador conserva la dependencia en y_1^{i-1}, factor que puede ser relevante a la hora de etiquetar palabras. En segundo lugar, el modelo implementa weight averaging, por lo que los parámetros finales del modelo consisten en una media de los valores de los pesos en las diferentes iteraciones del entrenamiento. Se conoce que está técnica ayuda a reducir "ruido" del proceso de entrenamiento, y versiones modificadas se han utilizado con éxito para mejorar el rendimiento

de sistemas modernos basados en redes neuronales. [8, 1]. Por último, este modelo incorpora información lingüística de manera explicita, los sufijos y prefijos de las palabras del contexto que rodea a la palabra a etiquetar. Si bien cabe esperar que esta información esté disponible en los word-embeddings, tenerla disponible de manera explícita facilita el proceso de optimización.

5 Anexo 1

Una persona que haya prestado atención a la presentación debería ser capaz de responder las siguientes preguntas:

- **¿Que modelos basados en redes neuronales se han propuesto? ¿Que arquitectura/paradigma sigue cada uno?** R: WNN, basado en redes feedforward. BLSTM, basado en redes recurrentes.
- **¿Cuantas muestras de entrenamiento se generan a partir de una frase del corpus en cada uno de los modelos propuestos?** R: WNN, se genera 1 muestra por cada palabra de la frase. BLSTM, se genera una muestra por frase.

References

[1] Parnia Bahar, Tamer Alkhouli, Jan-Thorsten Peter, Christopher Jan-Steffen Brix, and Hermann Ney. Empirical investigation of optimization algorithms in neural machine translation. In *Conference of the European Association for Machine Translation*, pages 13–26, Prague, Czech Republic, June 2017.

[2] Ondřej Bojar, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Philipp Koehn, and Christof Monz. Findings of the 2018 conference on machine translation (wmt18). In *Proceedings of the Third Conference on Machine Translation*, pages 272–307, Belgium, Brussels, October 2018. Association for Computational Linguistics.

[3] Eric Brill. A simple rule-based part of speech tagger. In *Proceedings of the third conference on Applied natural language processing*, pages 152–155. Association for Computational Linguistics, 1992.

[4] Cristian Cardellino. Spanish Billion Words Corpus and Embeddings, March 2016. Accessed 24/10/2018.

[5] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[6] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[7] Matthew Honnibal. A good part-of-speech tagger in about 200 lines of python, September 2013. Accessed 24/10/2018.

[8] Marcin Junczys-Dowmunt, Tomasz Dwojak, and Rico Sennrich. The amuedin submission to the wmt16 news translation task: Attention-based nmt models as feature functions in phrase-based smt. In *Proceedings of the First Conference on Machine Translation*, pages 319–325, Berlin, Germany, August 2016. Association for Computational Linguistics.

[9] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[10] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[11] Jorge Perez Rojas. Glove embeddings from sbwc. https://github.com/uchile-nlp/spanish-word-embeddingsglove-embeddings-from-sbwc. Accessed 24/10/2018.

[12] Mannes Poel, Egwin Boschman, and Rieks op den Akker. A neural network based dutch part of speech tagger. In *BNAIC 2008 Belgian-Dutch Conference on Artificial Intelligence*, page 217, 2008.

[13] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[14] Fei Sha and Fernando Pereira. Shallow parsing with conditional random fields. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 134–141. Association for Computational Linguistics, 2003.

[15] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[16] Peilu Wang, Yao Qian, Frank K. Soong, Lei He, and Hai Zhao. Part-of-speech tagging with bidirectional long short-term memory recurrent neural network. *CoRR*, abs/1510.06168, 2015.