Antonio Norelli

Natural Language Processing *a.a. 2016/2017* *prof. Roberto Navigli*

Homework 2: Supervised POS Tagger with LSTM

Abstract

Given a sentence we want to find the correct part-of-speech (POS) tag for each word. This is a classification problem with sequential data. The objective of the homework is to address this problem on the English data of the Universal Dependencies TreeBank with a Long Short-Term Memory Recurrent Neural Network (LSTM), that is a very effective classifier for this type of problem.

Model description

A LSTM accepts items of a fixed dimension organized in sequences of a fixed length. If we unroll the recurrent neural network, these two quantities define respectively the number of neurons on every sub-input layer and the number of sub-input layers. Due to these constraints a pre-elaboration of the data is required, because we have words and sentences of different length. The words are mapped in a 300-dimensional vector space using a technique called word embedding, that has also been demoed as a powerful representation for characterizing the statistical properties of natural language. More details are given in the next paragraph. Concerning the sentences, instead of considering as sequence the single sentence *s*, all the sub-sentences of *s* of a fixed length (`max_length`) are considered as sequences. In this way the problem of the fixed length is solved and moreover the learning data is increased (~8 times). Sentences with length lower than `max_length` are simply discarded.

A standard feed-forward neural network assumes that all inputs are independent of each other, a standard recurrent neural network, such as a LSTM, can exploit the dependency of an input from the previous ones, a bidirectional recurrent neural network, such as a bidirectional LSTM, can exploit also the dependency of an input from the next ones. In POS tagging future information is useful, there exist sentence patterns also in the right-to-left direction, this information becomes crucial for the first words where past information is scarce or absent. So taking as input the sequences of length `max_length` composed by the 300-dimensional word-vectors, the first layer of the network is a bidirectional LSTM that uses biases for the input vectors, has tanh as activation function, returns an output for every word-vector (`return_sequences=True`) as is natural in a POS tagging problem, and has 2x100 neurons for each sub-output layer. Each neuron of every sub-output layer goes into every one of 17 output neurons (17 as the number of Universal POS tags), this architecture is called Time Distributed Dense layer. Newly are used biases and the activation function is softmax. Softmax is used because our classes are mutually exclusive and the exponential behaviour of softmax lead near to 1 the higher output neuron and near to 0 the others, furthermore the sum of all 17 neurons is 1 and so a probabilistic interpretation of the results is possible. Now it is also clear why it is not used a smaller output of only 6 neurons that are sufficient to represent 17 different states, with this choice it is not possible to exploit the mutually exclusive characteristic of the classes and moreover the neural network would have to learn in a single layer how to POS tag and how to represent the output, leading to worse results.

Optimization

The model used to produce word embeddings is Word2vec, there are used the pre-trained vectors obtained using as training data a part of the Google News dataset (~100 billion words). The resulting mapping of the words in the 300-dimensional vector space have nice properties that are useful in the POS tagging problem. For instance it is possible to define a distance (cosine distance) that captures the semantic proximity between words. This results in closeness of similar words and allows to do vector operations like $vector(\text{"}Paris\text{"}) - vector(\text{"}France\text{"}) + vector(\text{"}Italy\text{"})$ and obtain a vector that is very close to $vector(\text{“}Rome\text{”})$. A problem of the pre-trained model used is that some words are missing, including very common words like “a” or “to”, punctuation and numbers (all words in the training set that are not included in the model are in the file stop-words.txt ordered by frequency). The solution used is explained in detail for each case in the file Word_embedding.py, in the following are reported few examples.

- **Word "a"** is almost interchangeable with "the", they are both articles and has about the same distribution of POS tags in the training set. "the" is included in the model, so "a" is simply mapped in the same word-vector of "the".
- **Word "to"** has not an equivalent word like "a". So the properties shown above are exploited. In the training set "to" is classified 63% PART and 35% ADP, "n't" and "within" are classified respectively 100% PART and 100% ADP in the training set, so "to" is mapped in the vector $0.64vector(\text{"n't"}) + 0.36vector(\text{"within"})$.

Since the described model of a single hidden layer composed by the bidirectional LSTM has had a good performance without evidence of underfitting, the tuned parameters do not include the number of hidden layers and in general the macro architecture of the neural network. The focus was instead on the number of output neurons of the LSTM layer and on the `max_length` parameter. The neural network was trained for each configuration up to 80 epochs, with `max_length` = 6, 8, 15 and number of output neurons of the LSTM layer = 2x100, 2x300. During the training it was recorded the accuracy on the dev set. The accuracy score on all the dev set is suitable to measure the performance of the classifier because the expected distribution of the predicted tags is not highly unbalanced. The data and the plots are in the last pages. It is possible to see that lower values of `max_length` performs better, this is not surprisingly, the resulting training set of lower `max_length` has more samples and probably the information from far words is not so informative for the one in analysis. Similarly the network with only 2x100 output neurons on the LSTM performs better, so more neurons are not needed to capture the full complexity of the problem.

Experiments

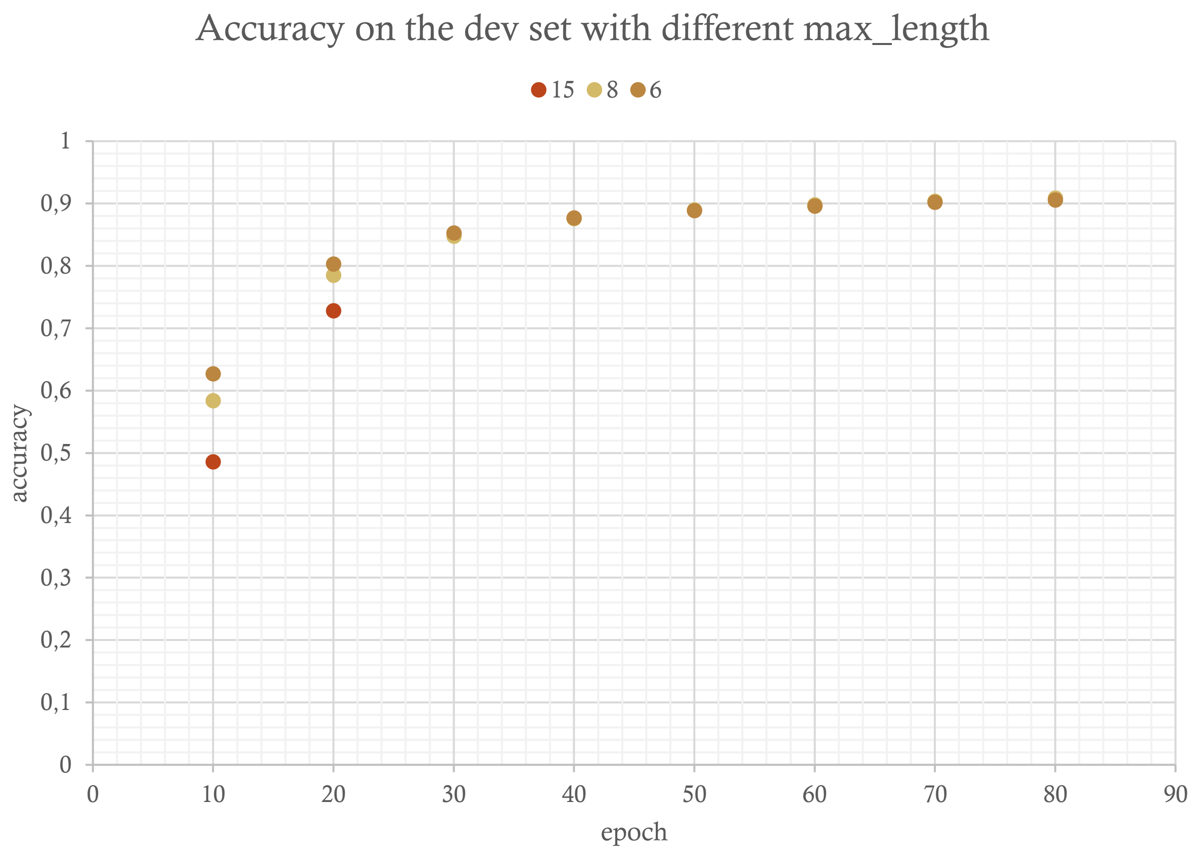
The chosen configuration is with `max_length` = 6 and 2x100 output neurons on the LSTM layer. This configuration is the one implemented in the code and tested on the test set, each predicted POS tagged sentence of the test set is reported in pos_tagged_sentences.txt. This configuration was trained for 100 epochs. The scores obtained on the test set are a precision, recall and F1 of 90.15%, and a coverage of 1. The equivalence of the first three scores and the coverage fixed to 1 are caused by the nature of the English language where a word can have one and only one POS tag and by the fact that the described classifier returns always a tag between the 17 possible ones. The results are satisfactory, more than 90% of words are correctly classified.

In order to have a more complete view on the performance of the classifier there were computed the confusion matrices (normalized and not normalized). In the last page are reported the corresponding heat maps. The diagonal is darker as expected, given the good scores obtained. Looking at the not normalized matrix it is possible to see the distribution of the POS tags, the more frequent classes are NOUN, VERB and PUNCT. The normalized confusion matrix is useful to compare the performance of the classifier among different classes. Not surprisingly the maximum precision class is PUNCT, punctuation is composed by unique characters and is not ambiguous at all. Over the 95% of precision there are also PRON, DET and ADP. All these classes share the particularity of being composed by few standard and common words, pronouns, articles and adpositions, these makes simple the classification. Most of the errors are due to ambiguity of words, given the context these could belong to a certain class or another. So looking at the more frequent errors it is possible to find which classes have more words in common. Observations are very predictable, the most common mismatch of a VERB is with NOUN, of an AUX is with VERB, of a SYM is with PUNCT. Classes with worst predictions are INTJ, SYM and X, a first problem for these classes is that in the training set there are very few words classified in them, moreover in the learning sets SYM and X, given their large definition, often are used to collect words that has a hard collocation. Take for instance the sentence of the test set at row 7495, composed by the two words "Michael Olsen@ENRON", they are classified as PROPN and PROPN while the gold standard says that they are X and X, a choice at least questionable, especially for the first word.

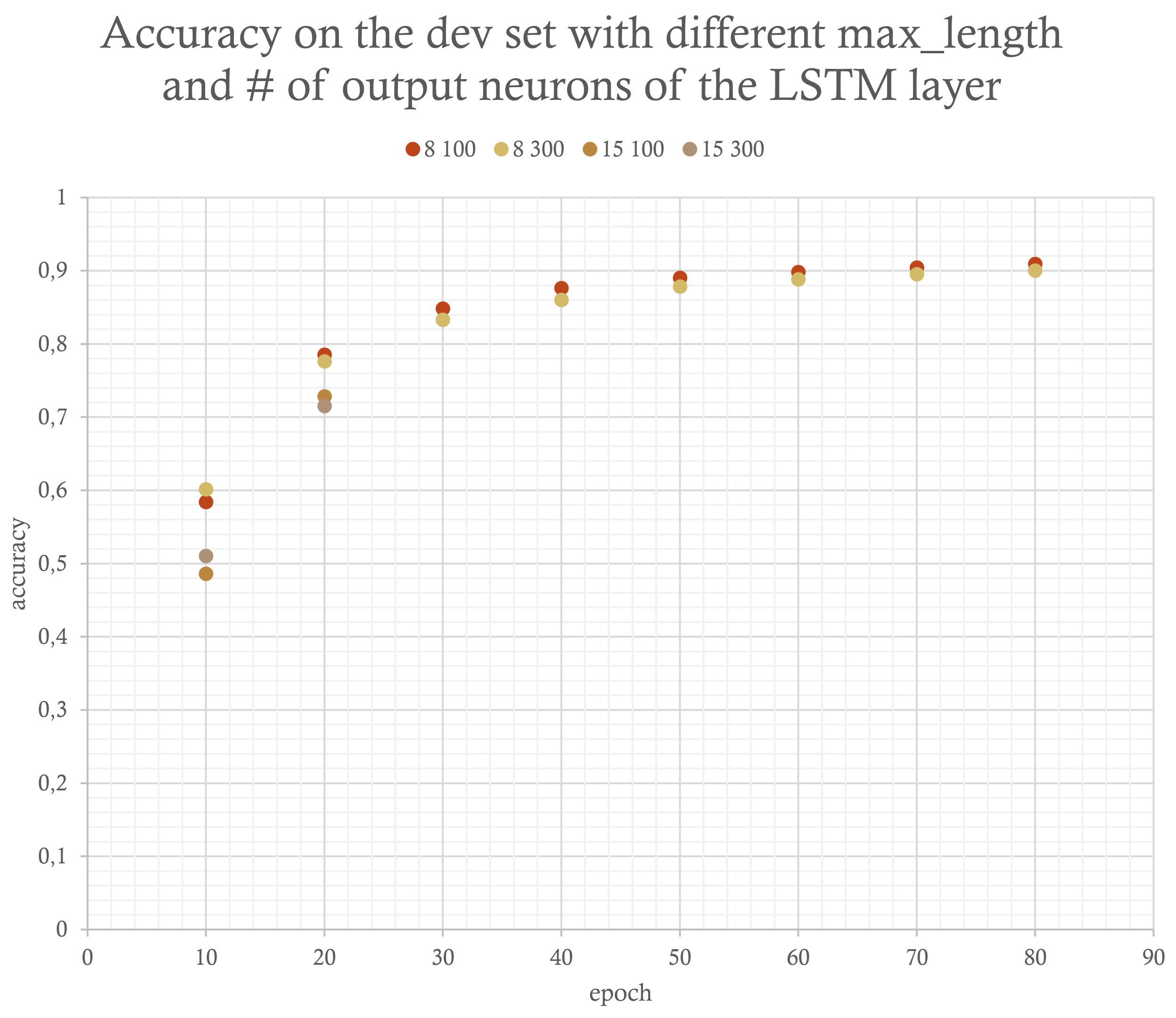
Extra

In order to use the same code for the multilingual task it is used an Italian pre-trained word2vec model (source: github.com/Kyubyong/wordvectors) in addition to the English one. The training, dev and test data are merged into single files containing all the English and Italian entries. Since the Italian pre-trained model contains less words of the other language than the English model, the vectorization of a given word is first tried using the Italian model and then, if the word is not in the vocabulary, it is used the English model as before. The chosen configuration for the multilingual POS tag is the same as the English POS tag (`max_length` = 6 and 2x100 output neurons on the LSTM layer). The results are worse, the precision falls to 88.00%. On one hand the Italian pre-trained model used is of lower quality than the English one, bigger and trained on sentences of Google News, that are nearer to the sentences of the learning data, but on the other hand Italian POS tagging problem is simpler than the English one. Most likely I think that the problem is in using two different pretrained models, in this way we are merging two independent 300-dimensional vector spaces into one, causing closeness of completely different vectors and losing the property of capturing the semantic proximity of two words with a defined distance in the vector space. So a good choice to improve the performance and exploit multilingual benefits could be representing words as a one-hot-encoding and add an embedding layer.

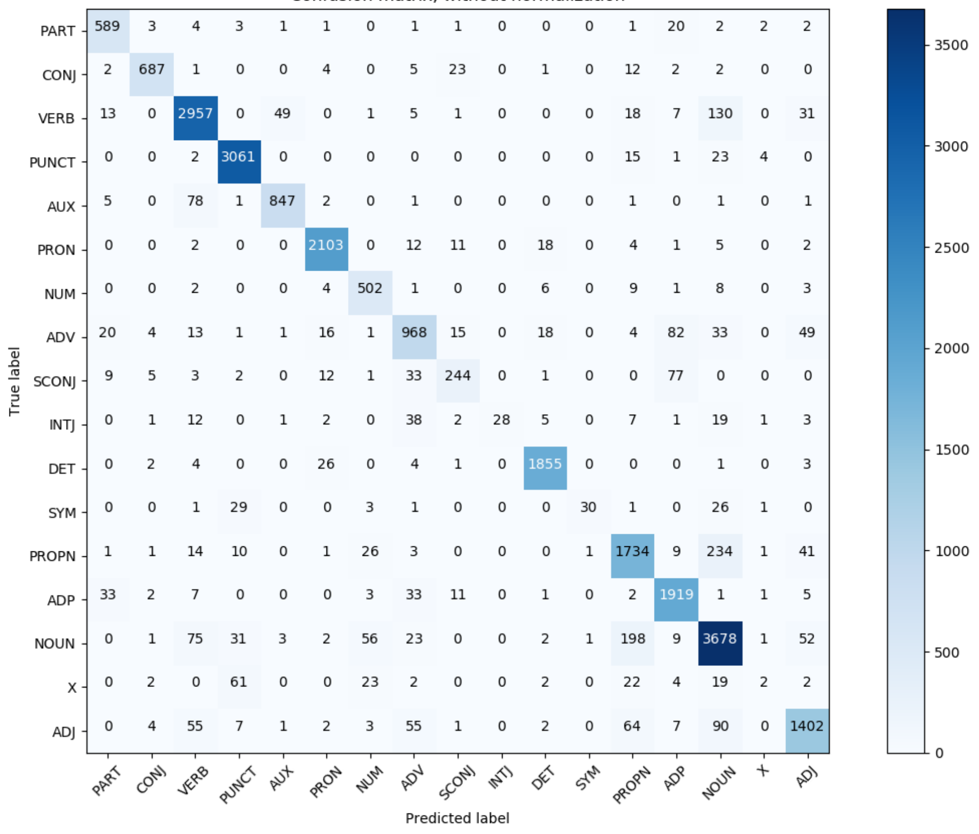
MAX_LENGTH	**15**	**8**	**6**
#SAMPLES	71838	126954	146729
EPOCH			
10	0,486	0,584	0,627
20	0,728	0,785	0,803
30		0,848	0,853
40		0,876	0,877
50		0,89	0,889
60		0,898	0,896
70		0,904	0,902
80		0,909	0,906



MAX_LENGTH	8	8	15	15
#OUTPUTNEUR	100	300	100	300
EPOCH				
10	0,584	0,6011	0,486	0,51
20	0,785	0,776	0,728	0,715
30	0,848	0,833		
40	0,876	0,86		
50	0,89	0,878		
60	0,898	0,888		
70	0,904	0,895		
80	0,909	0,9		

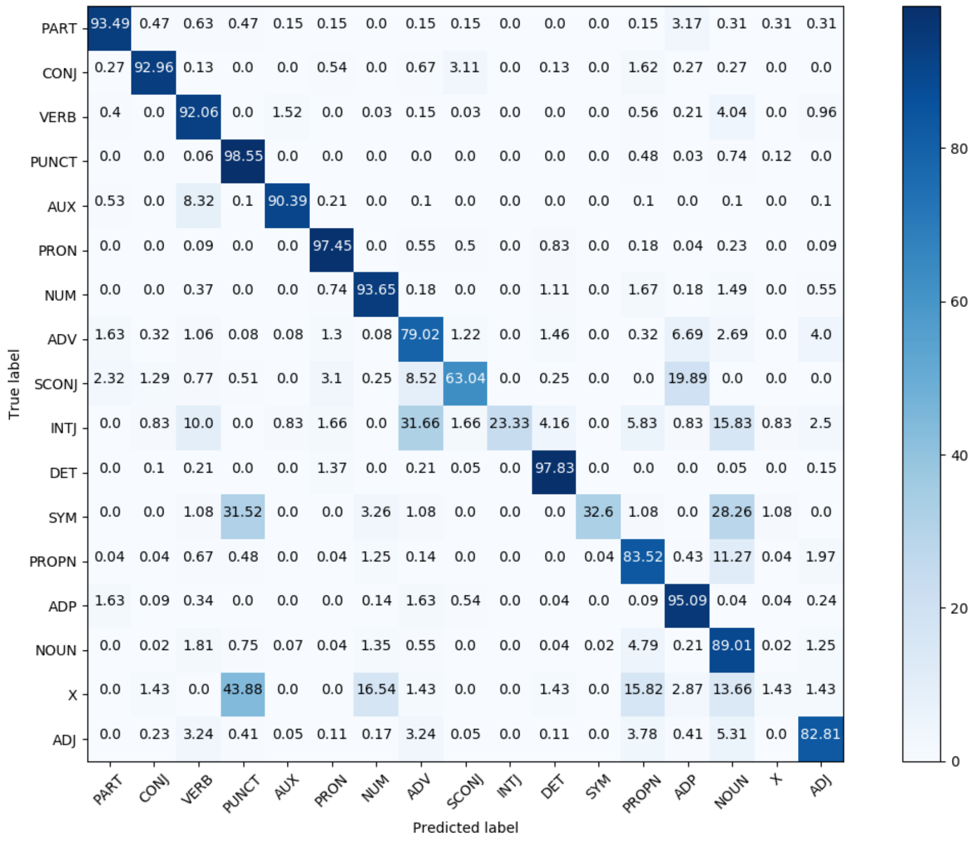


Confusion matrix, without normalization



True label \ Predicted label	PART	CONJ	VERB	PUNCT	AUX	PRON	NUM	ADV	SCONJ	INTJ	DET	SYM	PROPN	ADP	NOUN	X	ADJ
PART	589	3	4	3	1	1	0	1	1	0	0	0	1	20	2	2	2
CONJ	2	687	1	0	0	4	0	5	23	0	1	0	12	2	2	0	0
VERB	13	0	2957	0	49	0	1	5	1	0	0	0	18	7	130	0	31
PUNCT	0	0	2	3061	0	0	0	0	0	0	0	0	15	1	23	4	0
AUX	5	0	78	1	847	2	0	1	0	0	0	0	1	0	1	0	1
PRON	0	0	2	0	0	2103	0	12	11	0	18	0	4	1	5	0	2
NUM	0	0	2	0	0	4	502	1	0	0	6	0	9	1	8	0	3
ADV	20	4	13	1	1	16	1	968	15	0	18	0	4	82	33	0	49
SCONJ	9	5	3	2	0	12	1	33	244	0	1	0	0	77	0	0	0
INTJ	0	1	12	0	1	2	0	38	2	28	5	0	7	1	19	1	3
DET	0	2	4	0	0	26	0	4	1	0	1855	0	0	0	1	0	3
SYM	0	0	1	29	0	0	3	1	0	0	0	30	1	0	26	1	0
PROPN	1	1	14	10	0	1	26	3	0	0	0	1	1734	9	234	1	41
ADP	33	2	7	0	0	0	3	33	11	0	1	0	2	1919	1	1	5
NOUN	0	1	75	31	3	2	56	23	0	0	2	1	198	9	3678	1	52
X	0	2	0	61	0	0	23	2	0	0	2	0	22	4	19	2	2
ADJ	0	4	55	7	1	2	3	55	1	0	2	0	64	7	90	0	1402

Normalized confusion matrix



True label \ Predicted label	PART	CONJ	VERB	PUNCT	AUX	PRON	NUM	ADV	SCONJ	INTJ	DET	SYM	PROPN	ADP	NOUN	X	ADJ
PART	93.49	0.47	0.63	0.47	0.15	0.15	0.0	0.15	0.15	0.0	0.0	0.0	0.15	3.17	0.31	0.31	0.31
CONJ	0.27	92.96	0.13	0.0	0.0	0.54	0.0	0.67	3.11	0.0	0.13	0.0	1.62	0.27	0.27	0.0	0.0
VERB	0.4	0.0	92.06	0.0	1.52	0.0	0.03	0.15	0.03	0.0	0.0	0.0	0.56	0.21	4.04	0.0	0.96
PUNCT	0.0	0.0	0.06	98.55	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.48	0.03	0.74	0.12	0.0
AUX	0.53	0.0	8.32	0.1	90.39	0.21	0.0	0.1	0.0	0.0	0.0	0.0	0.1	0.0	0.1	0.0	0.1
PRON	0.0	0.0	0.09	0.0	0.0	97.45	0.0	0.55	0.5	0.0	0.83	0.0	0.18	0.04	0.23	0.0	0.09
NUM	0.0	0.0	0.37	0.0	0.0	0.74	93.65	0.18	0.0	0.0	1.11	0.0	1.67	0.18	1.49	0.0	0.55
ADV	1.63	0.32	1.06	0.08	0.08	1.3	0.08	79.02	1.22	0.0	1.46	0.0	0.32	6.69	2.69	0.0	4.0
SCONJ	2.32	1.29	0.77	0.51	0.0	3.1	0.25	8.52	63.04	0.0	0.25	0.0	0.0	19.89	0.0	0.0	0.0
INTJ	0.0	0.83	10.0	0.0	0.83	1.66	0.0	31.66	1.66	23.33	4.16	0.0	5.83	0.83	15.83	0.83	2.5
DET	0.0	0.1	0.21	0.0	0.0	1.37	0.0	0.21	0.05	0.0	97.83	0.0	0.0	0.0	0.05	0.0	0.15
SYM	0.0	0.0	1.08	31.52	0.0	0.0	3.26	1.08	0.0	0.0	0.0	32.6	1.08	0.0	28.26	1.08	0.0
PROPN	0.04	0.04	0.67	0.48	0.0	0.04	1.25	0.14	0.0	0.0	0.0	0.04	83.52	0.43	11.27	0.04	1.97
ADP	1.63	0.09	0.34	0.0	0.0	0.0	0.14	1.63	0.54	0.0	0.04	0.0	0.09	95.09	0.04	0.04	0.24
NOUN	0.0	0.02	1.81	0.75	0.07	0.04	1.35	0.55	0.0	0.0	0.04	0.02	4.79	0.21	89.01	0.02	1.25
X	0.0	1.43	0.0	43.88	0.0	0.0	16.54	1.43	0.0	0.0	1.43	0.0	15.82	2.87	13.66	1.43	1.43
ADJ	0.0	0.23	3.24	0.41	0.05	0.11	0.17	3.24	0.05	0.0	0.11	0.0	3.78	0.41	5.31	0.0	82.81