Analyse Textuelle

Année 2018-2019

Word Embedding - Prediction of unknown words

Élèves : BOUTHEMY Marin & NEROMA Kossi

19 mai 2019

1 Introduction

When we have looked at the sample document, we found that there were an important number of "UNK" tag for unknown words. Our goal was then to try to use the word embedding of the others words in order to predict what can be the unknown words.

A first approach was to use a Word2Vec model and try to guess the unknown words by using the similarities distance with its neighbour. To enhance our prediction, we then decided to take into account the Part-of-Speech (POS) tag of each word (if it is a noun, verb, etc...) and with a LSTM neural network, try to predict what can be the POS tag of the unknown word for a more accurate prediction.

By doing this, we were able to replace all the unknown words into the document and then re-train the Word2Vec model to get more accurate predictions.

2 Prediction of unknown words

We first use a Word2Vec model from the Gensim package and we trained it on some sentences from Wikipedia. Then we were able to get the vectorial representation of each words.

Suppose now that we have the following sentence :

I want to UNK banana and apple.

To try to approximate the unknown word, a first idea is to take the words around ('want', 'to', 'banana', 'apple') and get the most similar word to them.
We did that and obtain a pretty accurate and similar prediction to the real words (around 90% of similarity). However, in some cases, the word was not linked to the context to the sentence.

For example, in the previous sentence, we can guess that the 'unknown' word is probably a verb because there is a 'to' before it.
That is the reason why, we decided to take into account the POS tag of each word and try to detect what can be the POS of the missing word.

Imagine that the missing word is in reality the verb *nibble* (a synonym of eat) then it makes sense (and this is more correct) to suggest another verb in order to replace this unknown word (such as eat).

2.1 Implementation of LSTM

To do so, we first associate every word to its POS tag thanks to the NLTK package, we get the following for the previous sentence :

'pronoun' 'verb' 'TO' 'UNK' 'noun' 'Conjunction' 'noun'

We do a one-hot encoding to each of the categorical variables and then feed a Long Short Term Memory (LSTM) neural network with the four POS tag around the unknown around.

We chose to use a LSTM neural network in order to take into account the temporal dependency of a sentence. For example it makes sense that :

The order *('verb' 'to' 'unk')* is different than *('to' 'verb' 'unk')*

Because in the first one, 'unk' is certainly a verb which is false in the second one.

Furthermore, this neural network was outputting the density probability for each POS category and we selected the highest. We use the categorical cross-entropy as loss function and the accuracy as a metric.
However, the results that we obtain on the test set were not that good (12% of correct predicions of POS tag) but we do believe that it might be because the hyperparameters or the architecture of the neural network was not the best one. Another possibility is the fact that there were too many categories (45) where we could have restricted only to the most important ('verb', 'noun', 'conjunction', ...).

Finally, we combined both of the previous techniques to try to get the most accurate words. For each words, we were first predicting its POS tag and then we selected the most similar words with the same POS tag among the top 10.

3 Conclusion

Finally, we obtained that the results with or without the POS tag prediction were pretty similar (90 91%) for each case and we think that this is because the POS tagging was not that accurate and most of the time was not able to predict the correct POS tag.

However, we believe that using a double approach : one neural network for the POS tagging and another one (Word2Vec) for the word similarities tend to enhance the prediction of unknown words and once it is trained correctly, it is quite easy and fast to use.
For the results of the Spearman ranking and Analogy score, we use the Word2Vec model on the

training set (with the UNK tags replaced) and we get a score of 0.51 for the first one. You can find the results for the analogies in the outputs attached.