SkipGram

Aymane Benayada - Driss Debbagh Nour - Soufiane Hadji - Mehdi Mikou

18th February 2019

1 Introduction

The results obtained by our very first model were extremely poor compared to the ones we got using our latest model. The reason behind that is the large spectrum of improvements that can be applied on such a model.

Indeed, several steps were followed and each one can be improved, not only by changing the hyperparameters but also the whole methodology.

Below are some variations we experienced and some of the crucial decisions we took to improve our model.

2 Tokenization

During the tokenization of our data set, we wanted to remove the most frequent words. The choice of the threshold from which we consider that a word is very frequent and decide to remove it, is a tough task. So before removing any word, we wanted to try first an existing library and see if it adds value.

Indeed, we made the big assumption that removing the most frequent words is similar to remove the stopwords. (it is not the case for example for a book where usually the main character is called several times).

We used then the library nltk to remove the stopwords from our vocabulary. And the results were very interesting because they were very poor compared to the results when we keep the stopwords.

We wanted to conclude and keep the stopwords and then the most frequent words, but to make sure that we take the right decision we still implemented a function to remove words that appear more than a certain threshold. We tried few values for the threshold, and we decided from the results obtained then that we will not remove the most frequent words.

3 Hyperparameters

Many hyperparameters can be modified to improve the embedding. Applying a grisearch over all the following parameters can be a greedy method to obtains the best parameters. Unfortunately, it needs (for a personal computer) a lot of time.

Another issue would be the way of evaluating the results obtained. Indeed a human annotation is necessary to evaluate the robustness of the embedding.

However, we still varied some hyperparameters and evaluate our embedding by computing the most similar words to some reference words chosen beforehand.

After a quick review of the first sentences of our training dataset "news.en-00001-of-00100.txt", we chose to use the following words as references:

- president
- financial
- city

3.0.1 Size of the Sample

Obviously one of the most important parameters of our model. We observe a very notable impact on the embedding using the reference words. Since training our model takes a lot of time, we decided to train our model on two different data sets:

- 10,000 sentences : 219,305 words and 4,901 unique words
- 50,000 sentences (including the 10,000 sentences of our first training data set) : 1,094,214 words and 14,033 unique words

For the same hyperparameters, here are the ten most similar words to "financial" for each model

Financial	10k model	50k model	Similarity
1	union	economic	0.8605254987941997
2	web	banking	0.8543459085572801
3	across	housing	0.8391836329291854
4	east	global	0.8367757803069268
5	economic	institutions	0.8249920995141922
6	west	development	0.7974591339002757
7	rose	crisis	0.7927191095780013
8	afghan	energy	0.7923018373003
9	beijin	markets	0.7898427369024803
10	wales	european	0.7891532793944722

For the following variations, we used the 10,000 sentences model for a matter of time.

The basic parameters are: nEmbed=100, negativeRate=5, winSize=5, minCount=5, learning rate=0.01, epochs=5, batchsize=500

Whenever we made a variation on an hyperparameter, we kept the basic values for the other hyperparameters.

3.0.2 Window size

We tried five values for the window size {3; 5; 7; 9; 11} which represents respectively {+- 1; +-2; +-3, +-4, +-5}

Here above the results obtained when computing the 10 most similar words to "president":

Window Size	3	5	7	9	11
1	director	chief	chief	secretary	chief
2	california	secretary	secretary	minister	barack
3	chief	executive	minister	chief	obama
4	chairman	prime	john	barack	john
5	bank	angeles	executive	reuters	secretary
6	america	deputy	former	leader	minister
7	leader	director	richard	prime	sen
8	london	los	prime	angeles	mike
9	office	former	los	citing	washington
10	life	minister	director	council	citing

Evaluating the results is subjective. One interesting thing is to see that the word "chief" always appears among the top 3. Another curious observation is the appearance of both "barack" and "obama" on the top 3 in the case where minSize is equal to 11. We can see that the word president is very influenced by the general context of the data set. Indeed, the data set provides sentences from the American news. Taking a window size of 11 makes the word "president" loose a bit of generality and becomes very specific to the context.

3.0.3 Negative rate

As for the window size, we varied the value of the negative rate and computed the 10 most similar words to "president" (so we can also compare its impact with the one of the window size).

We gave 3 different values to the negative rate {2, 5, 8} and hereafter the obtained results:

Negative rate	2	5	8
1	chief	chief	chief
2	korea	secretary	executive
3	foreign	executive	secretary
4	secretary	prime	angeles
5	executive	angeles	minister
6	los	deputy	prime
7	saturday	director	state
8	south	los	los
9	director	former	director
10	smoking	minister	south

The first delightful observation is that the most similar word to "president" is the same no matter the negative rate chosen.

We also observe completely different similar words between 2 and {5, 8} starting from the second similar word. We can conclude that there is a threshold that we need to exceed to have interesting results.

Concerning the two choices 5 and 8, there isn't huge differences among the most similar words. But we can still observe better scores when computing the similarity for the case where the negative rate is equal to 8.

3.0.4 Rare words pruning

Instead of fixing a value no matter the size of our data set, we can choose a ratio of the size of the data set. For example, a minSize of 5 for a 50k sentences data set, is equivalent to choose a ratio of 0.0004% of the size of the data set.

3.0.5 Embedding space dimension

We only tried two value for this hyperparameter, but we still can make some conclusions over it. Here below the 10 most similar words to "president":

Embedding Dimension	100	300
1	chief	chief
2	secretary	prime
3	executive	executive
4	prime	minister
5	angeles	obama
6	deputy	secretary
7	director	state
8	los	director
9	former	john
10	minister	former

As for the variation of the negative rate, we observe in both cases the same most similar word to president. Globally, the results for the highest dimension are better. The result was expected.

Why don't we make the dimension the highest possible? A first obvious reason is because it takes much more time to train our model, but the issue that we can face by taking a very high value for the dimension is that it will overfit our training data set. We can see for example when we take a dimension of 300, the appearance of obama which means that we get nearer to the context of our training data set and loose some generality.

3.0.6 Batch Size

Our first approach consisted on actualizing our vectors after computing the gradient for each pair (positive or negative). We faced an issue which was the divergence of our gradient. We decided then to actualize our vectors only after computing the gradient for a certain number of pairs. We tried three values for the batch size "200", "500" and "2000" and noticed a slightly better results when we chose "500". The batch size shall be high enough but not too high.

However, increasing the size of batchs can help us avoid the gradient to diverge.

3.0.7 Learning rate

The learning rate can be very impactful when the size data set is too high. Indeed, the gradient can sometimes diverge. Hence we need to revise our learning rate (decrease its value).

4 Method 1 v.s. Method 2

We tried two different methods as explained in the README. The results were completely different since the second method gave us much better results than the first one.

We decided to send only the code for the method that worked best.

Both methods shared a lot of common parts. The main difference was the way we consider each word of our data set whenever it appears in a pair.

Indeed each unique word of our data set will have one or two embeddings depending on the method.

4.1 Method 1

Each unique word will have one unique embedding. Here again two methods were applied.

The first one consists on computing the gradient and actualising the embedding of both the target word and the context. The results were very poor. We decided then to proceed to the second method.

We considered that a pair (w1, w2) where w1 is the target word and w2 is the context will later become a pair (w2, w1) where w2 will take the role of the target word and w1 the context. Based on this observation, we decided to compute the gradient and actualise the embedding of only the target word. Indeed computing the gradient and actualizing the embedding of the context too will lead to a duplication of the pair (w1, w2) since they both play a symmetric role on the expression of the loss function used [1]. The impact of (w1, w2) would then be the same as (w2, w1).

The results were much better using this method, but they were even better to the following method.

4.2 Method 2

We had the idea of implementing this method after actualizing the embedding of the target word only on each pair.

Whenever we treated a pair we still wanted to compute the gradient according to the context, considering that it might impact our model. Indeed when actualizing the embedding of the target word, we take the information that the w2 is the context of w1 and use it to compute the new embedding of w1, but never the information that w2 was once the context of the w1.

From this point of view, we decided to make two embeddings for each unique word of the data set: one for the word as a target word and one as a context. So for each pair we compute both the gradient according to the target word and then the context. We actualize the embedding of the word 1 as a target word and the embedding of word 2 as a context.

Finally, as mentioned before, the results were much better than the first method.

References

[1] Omer Levy Yoav Goldberg. *word2vec Explained: Deriving Mikolov et al.'s Negative-Sampling Word-Embedding Method.* 2014.