ELMO for Word Sense Induction

Vanessa Skliarova
Vanessa.Skliarova@skoltech.ru

1 Introduction

Word sense induction (WSI) is the problem of grouping occurrences of an ambiguous word according to the expressed sense of this word. Lexical ambiguity, when single word has several meanings, is one of the fundamental properties of natural languages, and is among the most challenging problems for NLP. RUSSE-2018: Word Sense Induction was organized to compare approaches to WSI. The goal of this competition is to compare sense induction and disambiguation systems for the Russian language and choose the best for each dataset. Adjusted Rand Index (ARI) was used as a quality measure of a clustering for each word. There are several approaches to solve this task. The most important ones include generation of possible substitution for target words in particular context using neural network models and then clusters vectors of obtained words. Other approaches include building embeddings for contexts and then use different algorithms of clusterization. From the previous competition it was found, that such methods as Affinity and Agglomerative clusterization gave the best results. In this paper we will consider ELMO embeddings for WSI task.

2 Methodology

In competition RUSSE-2018: Word Sense Induction were provided baseline model and three datasets: wiki-wiki, bts-rnc and active-dict. As baseline model authors proposed adagram model and as evaluation metric - adjusted rand score (ARI).

Attempts to improve the baseline model using another lemmatization library pymorphy2 or imporved preprocessing with removing stopwords and foreign words didn't bring a lot improvement in test score.

So was decided try to apply more advanced algorithms to solve this task. Final algorithm includes following steps: data preprocessing, building context embeddings and clusterization obtained vectors.

2.1 Preprocessing

For preprocessing were used such libraries as pymorphy2, nltk and re. First of all it was decided to remove from sentences all punctuation and foreign words. All words were lemmatized with pymorphy2 library. For noise reduction it was decided to delete all stopwords and numbers. Also the different hypothesis about result improvement with removing prepositions, pronouns were checked. Unfortunately, this techniques didn't improve the result.

Described preprocessing steps were applied to all datasets.

2.2 Embeddings

For embeddings it was decided to use DeepPavlov library with pretrained models. Experiments were set for BERT sentence model and contextualized ELMO models, pretrained on wiki, news and twitter datasets. Scores for wiki dataset you could find in table 1.

ELMO model pretrained on wiki gave the best result on each dataset. This model takes as input word context and returns the contextualized embeddings for each word. The main problem was to build one vector for each context using its words embeddings.

At this step there appeared several hypothesis, how to do that. The easiest approach was to average all vectors in context. The second approach was for each context average only target words in this context. For example, for context "There is a beautiful castle in this forest. In this castle lives the dragon" and for word "castle", we will find two occurrences of this word in context and average them. The third approach was to use not only vectors of target words, but also it's neighbours from left and right side to bring more sense to model. From the previous example, it means to average "beautiful, castle, in, this, castle, lives" vectors. But this approach didn't improve the result. Final and the most promising approach was to use k-nearest neighbours for target words in each sentence and average them. This approach was applied for number of neighbours equal to 3 and 5, and equal to 1, if the context is very short. Overall, the best result was obtained in first two cases and gave 0.6520. Other score results you could find in table 2.

Also there were ideas to consider weighted neighbours vectors, where weights depend on distance. Maybe it will be a good solution to find in each context not only the closest vectors, but also remote. And finally vectors with the same sense should be distant to all remote vectors for this cluster. It seems promising approach to improve clusterization.

2.3 Clusterization

After obtaining context vectors the next step was properly clusterize them. Such methods as Agglomerative, Affinity and Kmeans clusterization with 2 clusters were tested. In this part the main problem was to decide how many senses does each word have. This information could be find in the dictionary, but in this task were used silhouette scores. Comparing results of clusterization algorithms the best choice was to use Agglomerative with number of clusters obtained maximizing silhouette scores, and cosine affinity with the use of average linkage.

3 Results

For wiki datasets were used pretrained ELMO model for embeddings, averaging all words in context for building context embeddings and Agglomerative clusterisation with silhouette score for find word senses. For clusterization were used cosine affinity with the use of average linkage. Final results you could find in table 1. Code of this model is in this colab link:

https://colab.research.google.com/drive/11Jphwfsdq6vLYQkyR62ApsU-E02sMLDA?usp=sharing

As wikipedia language seems to be the most similar to dictionary language it was decided to use ELMO, pretrained on wiki, not on news or twitter, for other datasets. So described model was applied to bts-rnc and active-dict dataset.

For bts-rnc dataset defined earlier clusterization with proposed parameters gave very low score, but with default parameters it gave the score similar to baseline. For building context vector were used several approaches. First of all averaging all words in sentence, using 5-nearest neighbours, and using k-nearest neighbours. For k-nearest neighbours was decided to change k depends on the length of context. So for length more than 15, we choose 10 neighbours, for length more than 6 - 5 neighbours, in other case - only 1 neighbour. This values were chosen after comparing the score results. This gave a great improvement, comparing to the first and second approach, but still got less, than baseline score. You could see all results in tables 4.1 and 4.2. Code for proposed earlier model you could find in following colab link:

https://colab.research.google.com/drive/1ybKYuINRSpYhztF4E2IbWrUR5EmoZ1Dn?usp=sharing

For active-dict were tested all above described methods, but results were very bad. See table 5. This dict has the smallest average context length and maybe provided models were not enough to catch senses in this dataset. SO for this dataset it's better to use baseline adagram model. Link to this model on colab is following:

https://drive.google.com/file/d/1Wy3by-dzRWPqn1bxLwHjihJyE6WkGOwg/view?usp=sharing

Wiki dataset got the best score on WSI task, because firstly, embeddings were trained on this dataset and secondly, the length of contexts was rather big and sometimes in one context there were more than 1 target word. It helps model to understand more dependences for target words. Also the number of clusters is equal to 2 in most cases for wiki dataset.

For participation in competition login: vanessa was used.

Table 1:pretrained embeddings on wiki dataset

Model	ARI
Authors_baseline	0.6278
BERT	0.61
ELMO_on_wiki	0.652
ELMO_on_news	less 0.1

Table 2: methods of building context embeddings on wiki dataset

Algorithm	ARI
Avg_target_words	0.652
Avg_all_words	0.652
Avg_neighbours	0.616
3-nearest_neighbours	0.59
5-nearest_neighbours	0.616

Table 3: clusterization methods on wiki dataset

Algorithm	ARI
Agglomerative_with_best_params	0.652
Agglomerative_with_default_params	0.59
Kmeans_2_clusters	0.652
Affinity_clusterization	less 0.3

Table 4.1: models on bts-rnc dataset with proposed clusterization parameters

Model	ARI
Authors_baseline_bts-rnc	0.26
Obtained_baseline_bts-rnc	0.22
ELMO_model_bts-rnc_5-nearest_neighbours	0.02
ELMO_model_bts-rnc_avg_all_words	0.06

Table 4.2: models on bts-rnc dataset with default clusterization parameters

Model	ARI
ELMO_model_5-nearest_neighbours	0.18
ELMO_model_k-nearest_neighbours	0.241
ELMO_model_avg_all_words	0.176

Table 5: models on active-dict dataset

Model	ARI
Authors_baseline	0.17
Obtained_Baseline	0.1596
ELMO_model_different_variants	less 0.06