Word Sense Disambiguation

Ihar Shulhan

February 2019

Word Sense Disambiguation

The goal of assignment is to tackle a problem of word sense disambiguation. The data comes from Senseval competition. The data includes a target word, context where it was used and a target sense id. In order to predict a sense of a word Wordnet dictionary was used, that contains all sense of the word and their dictionary definitions.

Most common guess

In order to build a baseline solution, we can try to use probabilistic approach. For each target word in a train set, distribution of different senses is calculated. Then the model simply always predicts that a sense is the one with the highest probability. Described approach achieves an accuracy of 0.516, which, as we would see later, is a benchmark that is hard to beat.

Simplified Lesk

The idea of simplified lesk algorithm is to use dictionary definitions in order to find if a definition uses the same words as in the context. One can define different functions that give a score of similarity between two texts.

In order to remove irrelevant information, stopwords provided by NLTK library are eliminated from both texts and target word doesn't contribute to the score. All words are converted to their base norm. Following approaches were tested:

Simple comparison

The score of similarity is simply number of shared words between two text normalised by a length of the longest sentence. This approach achieves an accuracy of 0.341.

The low score can be explained looking at misclassification matrix. It can be seen, that an algorithm chooses different sense, because senses described in Wordnet definitions are very close and words used in those definitions are similar. Example: A sense "art%1:04:00::
" is classified as "art%1:06:00::" in 35 out of 41 cases, but if we look at the dictionary definitions: "art%1:06:00::" - the products of human creativity; works of art collectively, "art%1:04:00::
" - the creation of beautiful or significant things, we can see that definitions are very close and probably the first one is more likely to intersect with some context then the second one.

Word2Vec

Previous approach has a downside of giving no score for synonyms. If both sentences describe the same things using different words, the score would be zero. In order to tackle this issue, Word2Vec

embeddings were used. For each word we get a vector provided by embeddings and calculate a cosine similarity between means of vectors for two texts.

This approach achieves an accuracy of 0.289. The issue in this case comes from the fact, that the algorithm seem to favour one sense over all samples, it means that cosine similarity score is not evenly distributed among different senses. Further research can be done in order to normalise similarities based on unique score values generated for each sense.

Machine Learning Approach

Another idea is to use machine learning in order to train a model which is able to predict a sense. A model would try to predict which of the sense provided by WordNet has the highest probability given current context.

The model uses scores calculated by word2vec similarity algorithm, probability distribution of senses for current target word and target word directly. Context was omitted, as it didn't provide a considerable accuracy boost, but led to high overfitting.

Two gradient boosting implementation were used CatBoost and LightGBM. Tested using 5-fold cross-validation, they achieved following results:

- CatBoost - accuracy: 0.5369 +/- 0.0145
- LightGBM - accuracy: 0.5329 +/- 0.0112
- Stacked CatBoost and LightGBM - accuracy: 0.5452 +/- 0.0089

Further research

One needs to notice, that machine learning approach wasn't fully explored as other machine learning algorithms should be tested and more extensive feature engineering performed. Originally, we've used only definitions provided by Wordnet to compare with a sample context, however we can also use train data to compare current context and those context that were found in a train dataset with the same sense.