Word-in-Context Disambiguation

Farooq Ahmad Wani* Homework 1

Abstract

Word in context disambiguation is the task of addressing the disambiguation of polysemous words, without relying on a fixed inventory of word senses. In this paper, I will present three different binary classification methods and compare their results to determine whether the indicated target word has the same meaning or not in two context sentences.

1 Introduction

In our daily life when we talk to people we often use the same word for different meanings and because the human race is intelligent and has very strong linguists we can easily distinguish the meaning of these polysemous words. This distinguishing of polysemous words is not easy for machines. In this paper, I will discuss three methods to check whether the meaning of the word is the same or not in the given context of two different sentences.

2 Data Processing

The training data given is in the form of JSON format in which each object of JSON has two sentences, lemma of the target word, the pos tag of the target word, etc. After parsing this data into the required format I had a list of tuples where each tuple has two sentences and the label. Before sending the data for tokenization **I replaced the target word with the same lemma of a word in both sentences. I did this so that distance should be only because of the context of surrounding words.** Then processed data is passed for tokenization where every token is replaced **by pre trained word embedding of the glove vector.** I tested to remove the stopwords and used the nltk pos tags, lemma, and other things but they didn't prove helpful to increase accuracy.

3 Vector Treatment

After the sentences are converted into the vector format each having the size of a number of words in **sentence * glove word embedding size** . For the first method, these vectors are passed to **hierarchical pooling**. This pooling is done to protect some of the contexts even if the simple method is applied after that. This pooling is followed by **max pooling** which generates the single vector for the whole of the sentences. Then every two sentences are concatenated by adding the **separator zero**. For the other two methods, the vector treatment is different after the word vector embedding the vectors are sent to the **collate** function to generate different vectors which will be discussed in sections to follow.

4 First Method

After processing the training data and converting it into the vectors using the **Hierarchical pooling,** I used the simple neural network of the three layers deep having 101,100 and 10 neurons respectively. I have used **Relu,** function to add non-linearity to the network and sigmoid function to get the output. I tested with the **batch normalization, dropout, and res net** but these techniques didn't improve the accuracy. After passing the vectors to the above designed neural network for 100 epochs with a batch size of 32, I tested on the validation set and the maximum accuracy I was getting every time is **64 percent**. then after tuning the hyper parameters and decreasing the size of glove vectors my accuracy on the validation set jumped to **70 percent.** the main tuning which helped more was increasing the **momentum** and decreasing the learning rate, besides that using SGD rather than Adam optimizer.

5 Second Method

In this method, I have used the **LSTM** , followed by the three layers of deep neural network. The vectors which were used for LSTM were processed differently before passing to it. I have written collate function which will be called by the data loader and return the **zero padded batches** so that size of every sequence will be the same in every batch. The embedding is done by the **using the pre-trained glove word to vectors** , besides processing the vectors differently the function also returns the length of word sequences in each sentence. Every two sentences in training data are concatenated by the **Zero paddings.** Thus collate function returns concatenated vector of the two sentences and the lengths before the zero paddings of every sentence and the point of concatenation. During the training process **two outputs have been taken from the LSTM**, one at the sequence length of the first sentence and another at the sequence length of the second word. **The euclidean distance of these two vectors has been passed to the followed layers of the neural network.** After much of the hyperparameter tuning and a lot of training the maximum accuracy was **68.6 percent.**

6 Third Method

In this method **Bi directional LSTM** has been used followed by the two layers of the ANN. For this method collate function is designed differently which returns not only the concatenation of the word embedding indices of the two sentences in context but also returns the initial sequence length, first sentence sequence length, concatenation point length, and the second sentence sequence lengths. All this data is being passed to a bidirectional network during the training where it helps me to take two outputs. **After taking the two outputs from the bi-directional LSTM each of which is of length 20 I have taken the euclidean for the first half of the outputs in the forward direction and then in the reverse direction for another half.**This type of the operation has been done so to save the both forward and the backward context of the sentences. The single distanced output is being passed to the followed layers of ANN and the final output is generated. The maximum accuracy of this method reached is **67 percent.**

7 Comparison

There was not that much difference in the accuracy of the three methods but the maximum accuracy attained by bi directional LSTM was faster compared to the other two. The decrease in loss was much faster in bidirectional compared to the other two methods, but I finally choose the first method because the predictions are quicker than the other two and have 1 percent more accuracy compared to the other two methods.

Extras

- I replaced the target word with the same lemma of a word in both sentences. I did this so that distance should be only because of the context not because of different forms of the same word.
- Used glove pre-trained vectors for the word to vector embedding.
- I have used the Hierarchical pooling to save the context even for simpler methods.
- Used zero separator for concatenation of the two sentences in context.
- Tried batch normalization,skip connections (res net) ,dropout etc.
- momentum
- padded every sequence vector separately then concatenated for LSTM using zero pad took two different outputs from LSTM at respective sequence lengths
- Returned random seed for the words not found in embedding.
- euclidean distance of the outputs in LSTM for rest of ANN
- bidirectional LSTM
- Four outputs for bidirectional LSTM and each one divided into two halves then added first two for forwarding context and next two for backward context and took the euclidean distance of these to get a single vector.

References

- Class lectures and notebooks by Roberto Navigli, Sapienza University of Rome
- Baseline Needs More Love: On Simple Word-Embedding-Based Models and Associated Pooling Mechanisms by Microsoft Research Team
- Enhancing Sentence Embedding with Generalized Pooling by Zhen-Hua Ling,Qian Chen, and Xiaodan Zhu