Final Project: WSD with BERT
Felipe Caldeira and Shalom Alarape

Anonymous ACL submission

Abstract

To figure out a words meaning, it requires the context of the words around it. By replicating a paper for WSD, we developed a predictor function that takes in a sentence and a specific word that you want a definition/sense of and returns the correct definition/sense. In order to do this, we used DistilBERT and MLP classifier. After training and validating our classifier, we then tested the model on four datasets which gave us some great F1 scores. Using that model, we then created the predictor function. This function takes in a sentence and a word that you wants to figure out the sense of, and returns the definition/sense of the word. With the function, we saw high accuracy when it comes to find the sense/definition of words in sentences.

1 Introduction

Fall. Is that the word being used as an action or the season? What about date? Is it the activity, the month and day, the food, or aging yourself? These words and many more are considered homonyms which are words that are spelled the same but have different meanings. When it comes to NLP, this part of languages can be quite difficult. How can we a computer decide what sense a word has?

Our goal is to create a model that, given a word in a sentence, it can come up with the sense of the word. In other words, certain words have more than one meaning based on what the context is. Let us look at the word bat. In one situation, it describes an inanimate object used in a game. While at other times, it is used to describe an animal. For us to be able to make this distinction on a computer, we are using recreating a paper by Jiaju Du, Fanchao Qi, Maosong Sun titled ***'Using BERT for Word Sense Disambiguation'*** (Du et al.,2019).

Why does word sense even matter for NLP? This is an important thing to distinguish accurately because, without the correct understanding of a word it may confuse the reader/listener. Having the wrong sense of a word can make a sentence hard to comprehend or give an incorrect summary. The sense of the word is helpful specifically to NLP because it can effect the programs ability to summarize or answer question in regarding to what it hears or reads.

2 Data

We got our data used different datasets from different sources (Raganato et al.,2017). For testing, we used four different ones: Senseval 2 (Edmonds and Cotton,2001), Senseval 3 (task 1) (Snyder and Palmer,2004), SemEval 2013 (task 12) (Navigli et al.,2013), and SemEval 2015 (task 13) (Moro and Navigli,2015).

For training we used SemCor (Miller et al.,1994) which uses senses from an inventory called WordNet 1.4. For our validation we used SemEval 2007 (task 17) (Pradhan et al.,2007) which uses the WordNet 2.1 sense inventory and is the smallest of the datasets. SemEval 2013 and 2015 both use WordNet 3.0 while the Senseval 2 and 3 use WordNet 1.7.

The data sets are in the format of .xml and .txt files. In the .xml file (which is what we iterate on) it give us a sentence with certain ones tagged as instance which are the ones that the sense is trying to be found. Within one sentence there can be multiple instance words. Additionally, each of these words are given their type (Noun, Number, Conjunction, etc..). And for the instance words, they also have an ID to go with them in the same way that sentences do.

3 Methods

The method that we used follow very closely to what was discussed in ***'Using BERT for Word***

Sense Disambiguation' (Du et al.,2019). The first step was to acquire all the data and put them into our file. This includes doing the stuff with Java Scorer. After that, we moved on to work on the encoder. The encoder is one essential part of the process because it will tokenize the sentence and then give us the calculated hidden states for each token.

In our first few attempts at this replication, we used the original BERT encoder. However, this led to our program running for several hours for each epoch. In order to progress with the project better, we decided to use the DistilBERT encoder, a lightweight version of BERT with similar performance. With both versions, there needed to be a tokenizer, a config, and a model. For the tokenizer, we had it truncate and pad the tokenized sentence. This allowed us to have similar lengths for all the tokens. For the configuration, we made sure it allowed us to see the hidden states in order to use them later when sending them to the feature vector.

Once that was completed, there were some key functions that we would need in order to create our classifier. One was to create the feature vector. To do this, we need to average the values of the hidden state for the particular tokens that make up the word we are trying to find the sense for. The formula is shown below

$$\mathbf{f} = \frac{1}{k}\sum_{l=0}^{k-1}\mathbf{h}_{j+l},$$

A few other functions we created were getting the encoded index of the sentence. Another one we created was to evaluate how the model did in regard to the dataset. Here is where we used F1 as our scoring metric. F1 takes into account the accuracy (precision) and the recall of the guessed sense. This is a good metric because we are not only looking to be accurate but we are also looking to reduce the number of false negatives.

Next, we created the Multi-Layer Perception (MLP) general classifier structure that we will then train and validate. The MLP takes in the feature vector of size H and passes it through two layers. Layer 1 (L1) is a fully-connected linear layer with an H x H mapping. The value of that then gets passed into ReLU (Rectified Linear Unit) as the activation function. The result of that gets passed to Layer 2 (L2), which outputs the probabilities of a polyseme's senses. Because In L2, which is in the form of a dictionary, the value from the ReLU gets passed in. However, the main difference here is that L2 is based on the number of senses of each word. So the dictionary holds the polyseme of each word. And for each polyseme, it creates a forward network taking in the number of senses and H. Once we find the value, we take the softmax to get the sense distribution for that specific word. The formula for the MLP structure is shown below

$$\mathbf{p} = \text{softmax}(L_2(\text{ReLU}(L_1(\mathbf{f})))),$$

This leads us into the training, validating, and testing part of the classification. For the training process, we are aiming to minimize the loss between the actual label of the word and the sense distribution that is calculated using the layered formula we discussed above. The formula for the loss is:

$$L = -\frac{1}{M}\sum_{m=1}^{M}\sum_{s=1}^{|S_m|}[\mathbf{y}_m]_s \log[\mathbf{p}_m]_s,$$

We also needed an optimizer and a scheduler to handle our training data. After training, we then did validation. To do this, we evaluated the MLP tho the validation dataset, which gave us the validation loss and F1 score. Once validation was complete, we then did testing with the four datasets (Senseval 2 and 3 (task 1), SemEval 2013 (task 12), and SemEval 2015 (task 13) where we looked at their F1 scores using the same evaluate function that we used in the validation process.

With all this information and a good model, we created a predictor function. This function takes in the sentence and the word(polyseme) and gives us the predicted sense. To do this, we first made sure the sentence is lower case and then split it. Then we find the index of the word we are looking for in the string. We then get the feature vector and send that to the MLP that we trained. We then get the index of the sense we want. We then use that index to output the sense from the list of senses of that word. We tested this out on a few sentences and found high accuracy.

4 Results

4.1 Training, Validating, and Testing

Original learning rate

Dataset	F1 Score (percentage)
Senseval 2	53.1328
Senseval 3	48.1868
SemEval 2013	49.1312
SemEval 2015	49.2943

Updated learning rate

Dataset	F1 Score (percentage)
Senseval 2	53.6645
Senseval 3	50.9689
SemEval 2013	48.7358
SemEval 2015	50.5910

4.2 Predictor Function

Try it out!

```
predictSense("This is such a lovely day!", "lovely")
```

```
Predicted sense for lovely:
        appealing to the emotions as well as the eye
```

```
predictSense("Could you stand up please?", "stand")
```

```
Predicted sense for stand:
        be standing; be upright
```

```
predictSense("Let's take a stand against the government.", "stand")
```

```
Predicted sense for stand:
        hold one's ground; maintain a position; be steadfast or upright
```

5 Discussion

Let's first look at the testing results. For all the datasets, the F1 scores were around 50% for both the learning rates. For the second learning rate, we tried to reduce it to improve the F1 scores. This worked for all but one dataset, SemEval 2013 which actually decreased by around 0.4%. This might be something to work on in the future in order to fine-tune the classification process. Now for the predictor, the results show high accuracy. This is very evident in the example of the word stand. The predictor function was able to tell the difference between standing as a person remaining in a position versus choosing a side on a specific topic.

6 Group Work Breakdown

The assignment was twofold: coding and write up. Felipe was the main coder for the project. This occurred because there was not a good way for us to co-code. That being said, Felipe and Shalom worked on finding papers for our project. Additionally, as a team, we discussed a few tricky parts of the project and tried to figure them out together. Both members of the group can explain the code in detail. In other words, both are knowledgeable about the code. In regards to the write-up, Shalom was the leader of that. That being said, both Shalom and Felipe worked on editing and cleaning up the paper.

References

Jiaju Du, Fanchao Qi, and Maosong Sun. 2019. Using bert for word sense disambiguation. *arXiv:1909.08358 [cs]*. ArXiv: 1909.08358.

Philip Edmonds and Scott Cotton. 2001.SENSEVAL-2: Overview. In *Proceedings of SENSEVAL-2 Second International Workshop on Evaluating Word Sense Disambiguation Systems*, pages 1–5, Toulouse, France. Association for Computational Linguistics.

George A. Miller, Martin Chodorow, Shari Landes, Claudia Leacock, and Robert G. Thomas. 1994.Using a semantic concordance for sense identification. In *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*.

Andrea Moro and Roberto Navigli. 2015.SemEval-2015 task 13: Multilingual all-words sense disambiguation and entity linking. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 288–297, Denver, Colorado. Association for Computational Linguistics.

Roberto Navigli, David Jurgens, and Daniele Vannella. 2013.SemEval-2013 task 12: Multilingual word sense disambiguation. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 222–231, Atlanta, Georgia, USA. Association for Computational Linguistics.

Sameer Pradhan, Edward Loper, Dmitriy Dligach, and Martha Palmer. 2007.SemEval-2007 task-17: English lexical sample, SRL and all words. In *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, pages 87–92, Prague, Czech Republic. Association for Computational Linguistics.

Alessandro Raganato, Jose Camacho-Collados, and Roberto Navigli. 2017. Word sense disambiguation: A unified evaluation framework and empirical comparison.

Benjamin Snyder and Martha Palmer. 2004.The English all-words task. In *Proceedings of SENSEVAL-3, the Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, pages 41–43, Barcelona, Spain. Association for Computational Linguistics.