Mathieu-Joseph Magri 260 928 498
COMP 550 November 16th, 2024

Report – Programming Assignment 2

In this assignment, the main goal was to test and compare a variety of word sense disambiguation (WSD) algorithms and models. The first algorithm implemented was Lesk's algorithm. Two versions of this algorithm were used, one where the algorithm was fed the context and the target lemma and another enhanced version where the part-of-speech (POS) tag was also included. Then, as a baseline, we evaluated the most frequent sense (according to Wordnet) of each word and compared it to the answer keys.

Following the creation of those models, I decided to implement a Naïve Bayes classifier, more precisely, a Multinomial Naïve Bays (MNB) classifier. The reason for this choice is that a NB classifier works well with small datasets (like the one we currently have), is easy to extend with additional linguistic or semantic features, is easy to understand and implement and it also works well with labelled data which is our case here. However, to do this, I needed to generate an entire new dataset. I created my own data with ChatGPT. I asked ChatGPT to create sentences that featured the work "bank" and to include the WordNet definition used for that instance of "bank". The specific word prompt used was: "Generate 150 instances of the word bank used in a sentence as well as its wordnet sense." In this model, I decided to focus on single word WSD with regards to the MNB classifier given that this sort of classifier was not fit to work on a dataset that aimed to disambiguate more than one word rather than only one word. As mentioned previously, the chosen word I decided to do was "bank" given that it is a word with a variety of definitions as well as POS tags. With regards to the MNB model, we first compared countVectorizer against Tfid vectorization. Then we lemmatized the data to see how it influenced performance. Finally, we did some hyperparameter tuning with cross-validation on the created development set which was chosen due to the fact that it is a rather useful method when we don't have much data available.

Finally, a BERT model was implemented on both the original development and test set. BERT is a pretrained LLM model that can be used for a variety of NLP tasks such as WSD. Essentially, the BERT word embeddings further analyze the relationship between the lemma we are trying to discern, and its context. Therefore, with this added analysis, BERT should lead to an increase in performance.

All models were evaluated on both the development set and test set and MNB was evaluated on the created data. Where applicable, the development set was also used to do some hyperparameter tuning (MNB). The following table is an agglomeration of all our accuracy results. Accuracy was calculated by comparing the

model's output, a synset of the form "group.n.01" and then compared to the dev_keys and test_keys which were converted from sense keys of the form "group%1:03:00::" to synset defintions.

Table 1: Accuracies of all the developed models for this assignment.

Model	Accuracy (Development Set)	Accuracy (Test Set)
Lesk	34%	34%
Lesk with POS tags	37%	38%
Most Frequent Sense	67%	62%
MNB with CountVectorizer	Not Applicable	50%
MNB with Tfid	Not Applicable	47%
MNB Lemmatized	Not Applicable	43%
MNB with tuning	Not Applicable	50%
BERT	49%	47%

As we can see from these results, unfortunately, not one model was capable of beating the most frequent baseline. This can be due to a variety of things. Firstly, with regards to Lesk, there are many possible things that can be done to increase its performance such as using POS tags in the method. Although this was done, accuracy only increased by about 3-4 points. The MNB classifier performed much better but still not great. Tuning the model and using CountVectorizer seemed to maximize its performance at 50%. BERT also did not perform as I had hoped as it only obtained about 47% accuracy on the test set. There are many things that can be done to a BERT model to increase its performance such as including glosses within the model and simply fine tuning the model on a WSD dataset like SemCor, but they were not done here. Perhaps with those additional features, BERT's performance would beat the most frequent sense baseline. Furthermore, it is important to note that for all models, training data was limited, and this can also have an impact on performance given that not a lot of data was available. This is especially true with the MNB classifier given that the work "bank" has over 10 meanings which means that for the model to be properly trained, it would require a good amount of sentences with all the possible meanings for the work "bank", but I simply generated 150 cases, all definitions included. There could have also been ways to compute overlap in ways that could have increased our accuracy but that was not explored further with any of the models. Finally, it was a purposeful decision not to remove stop words nor punctuation from any of the datasets given that stop words and punctuation can be clues to a word's meaning.

Note: In the submission zip, I included the Jupiter file as well as the python file in case the python file does not run, although it should work.