Programming Assignment 2: Word Sense Disambiguation

Hamza Rashid

November 15, 2024

1 Introduction

remember to compare the accuracies

Word Sense Disambiguation (WSD) is the task of determining the correct sense of a word in context. In this assignment, Lesk's algorithm and other methods were implemented to resolve word senses in the SemEval 2013 Shared Task #12 dataset, using WordNet v3.0 as the lexical resource. This report presents the methodology, results, and analysis of these experiments.

2 Methods

2.1 Baseline Methods

For Most Frequent Sense (MFS), we utilized NLTK's `wordnet.synsets()` method, choosing the synset whose sum of lemma frequencies was the largest. This method yielded 55.93% accuracy on the test set that was provided. Preprocessing considerations included lowercase conversion and underscore removal when dealing with multi-word phrases, the former decreasing the accuracy, and the latter being necessary for all methods in this assignment. Moving on to Lesk, its use of token level comparisons led to various preprocessing experiments on the dev set: stopword removal, puncuation removal, and lowercase conversion. Since the dataset was biased toward international politics, we saw a performance decrease with lowercase conversion. Punctuation removal slightly increased the accuracy, which could owe to NLTK's `word_tokenizer` seperating punctuation from the adjacent word, leaving more opportunity for unwanted overlap. Similarly, stopword removal improved the accuracy, trading quantity for more meaningful overlap. In the end, Lesk's best accuracy of 30.41% was acheived with stopword and punctuation removal, and case retention. The relatively poor performance is likely caused by the specialized dataset.

2.2 Custom Methods

2.2.1 Method 1: Named Entity Overlap (NEO)

Explain the use of a pretrained language model (e.g., BERT) for WSD, including how contextual embeddings were utilized to resolve word senses. For NEO, we perform a variant of Lesk on the named entities of the context and synset example or definition (when no example text exists). Our intuition is that an example is liklier to contain meaningful entities to be compared against. A key challenge faced in this approach is the sparsity of entities. As a result, we experimented with various models varying in size, speed, and number of entities used in their pre-training. The latter addresses the sparsity directly, while the former two explore the tradeoffs between speed and accuracy.

2.2.2 Method 2: Transformer embeddings L2 distances (TeL2d)

Describe the design and implementation of the second method, highlighting its uniqueness compared to the first method. Our hypothesis is that a transformer's contextualized embeddings can be used to accurately identify the sense of a word, given its context and synsets' examples (if they exist) or definition. We use a pre-trained transformer model to generate document embeddings, and select the synset whose example text embedding minimizes L2 distance with the context embedding. This method continues in spirit of NEO, attempting to generalize Lesk to account for implicit similarities. There are a considerable number of synsets for which an example text does not exist, in which case, we use the definition. We experimented with the same preprocessing decisions as in the Lesk method, as well as model selection and hyperparameter tuning on the dev set. Punctuation and stopword removal had adverse effect on accuracy, meaning that higher quality embeddings depended on textual richness. It is then reasonable to postulate that the underlying dataset's lemmatization is not an ideal preprocessing decision in this setting. For the purposes of the assignment, we left the dataset unchanged.

representations depends on the surrounding textual richness, at least in the task of WSD. We compared three transformers varying in both in size and pre-training task.

indicates that textual richness improves perhaps owing to the dependence of This leads one to believe that without the underlying dataset's lemmatization, this method would have performed better.

3 Experimental Setup

3.1 Dataset

Explain the use of the SemEval 2013 Shared Task #12 dataset, including handling multi-word expressions and the separation of dev and test sets.

3.2 Evaluation

Discuss the use of accuracy as the evaluation metric and how lemma sense keys were mapped for evaluation purposes.

4 Results and Discussion

4.1 Baseline Results

Present and analyze the results of the MFS and Lesk methods, comparing their performance.

4.2 Custom Methods Results

Discuss the performance of the custom methods and compare them with the baselines.

4.3 Analysis

Analyze the successes and challenges of the models, providing specific examples of correct and incorrect predictions.

4.4 Improvements

Offer suggestions for improving the methods, such as refining context representation or leveraging additional data.

5 Conclusion

Summarize the findings and highlight the key takeaways from the experiments.

References

- Navigli, R., & Jurgens, D. (2013). SemEval-2013 Task 12: Multilingual Word Sense Disambiguation. *Proceedings of the 7th International Workshop on Semantic Evaluation.*
- WordNet Documentation: `https://wordnet.princeton.edu/documentation/senseidx5wn`