Natural Language Processing

Programming Assignment 2
COMP 550

Authors: *Ling Fei Zhang*

Instructor:	*Jackie Cheung*
Department:	Computer Science
Date:	November 20, 2023

1 Overview

This assignment focuses on implementing word sense disambiguation algorithms using `wordnet`'s senses. Four algorithms are explored:

1. **The most frequent baseline**. This serves as a baseline by selecting the #1 indicated sense in the synset from `wordnet`.
2. **Lesk's algorithm**. Utilizes dictionary definitions of a word's senses, disambiguating based on the highest overlap score between the current context and dictionary definitions.
3. **A bootstrapping algorithm (Yarowsky's)**. The idea is to apply a minimally supervised (semi-supervised) machine learning approach. The high level idea is to manually label a small set of examples, repeatatively train a supervised model, and add the highly confident classifications to the seed set.
4. **Baseline + Lesk**. This hybrid approach integrates the baseline method with Lesk's algorithm. Essentially, it involves applying a weight penalty to less frequent senses.

2 Baseline Versus Lesk

Implementing the baseline method is straightforward, as we simply take the first sense from `wordnet`. Lesk's algorithm, from `nltk`, was also easily integrated. However, we made a few adjustments to enhance its performance. Firstly, we ensured lemmatization of the word for disambiguation. Secondly, we computed the POS tag for the word to narrow down the list of senses. Although we initially attempted to remove stopwords from the context, this negatively impacted the development set's performance. Thus, we retained the context words as they are. Below we show the performance of the two algorithms

Method	Accuracy
Baseline Method	0.62
Lesk Method	0.38

This result is quite surprising to us, as Lesk's performance is significantly lower than the baseline method.

3 Bootstrapping Algorithm

To implement the bootstrapping algorithm, we acquired training data from the `OntoNotesv5` dataset [1], comprising over 10k English training sentences. The training dataset is provided in `train.txt`.

To select the five lexical items for word sense disambiguation, we identified the top five lemmas with the highest counts in the testing set. This approach aimed to ensure a more equitable evaluation of the algorithm's performance. A quick count revealed the top five words as [('game', 23), ('year', 22), ('player', 21), ('team', 19), ('case', 18)], with associated numbers indicating the word's frequency in the test set. For each word, we manually labeled a starting seed set for binary sense classification. Each sense comprised 2 to 4 sentences obtained from a mix of `wordnet.synset.examples()`, ChatGPT 3.5 generation, and sentences created manually. We applied *Yarowsky*'s algorithm with a default training epochs of 10, using a Naive Bayes classifier. In each iteration, we retained data with a confidence level of 85% or higher, incorporating it into the seed set. During testing, a prediction was considered correct if the predicted synset matched one of the sense keys' senses. The following presents the algorithm's results, including a weighted average across all five words.

Word	Accuracy
game	0.96
year	0.95
player	1.0
team	0.84
case	0.67
Weighted Average	0.89

Challenges During the implementation, numerous challenges surfaced with the algorithm, leading to incredibly poor results. Initially, the word `year` achieved only 5% accuracy. Upon debugging, we noted that accuracy decreased as the word's frequency in the training set increased. This issue came from our large training dataset of 5000 sentences, resulting in frequent occurrences of each word and numerous senses per word. Given that the models were trained for binary sense classification, exposure to multiple senses worsened our classifier's performance. Consequently, we reduced the training data size to 800 sentences and made slight adjustments to the starting seed, significantly improving accuracy to 89% on average. It's worth noting that the accuracy for 'case' is considerably lower than others. This is potentially due to more instances in the training data classifying 'case' as `case.n.01`. In other words, the model had limited training for the `lawsuit.n.01` classification, impacting its performance during testing.

4 Baseline + Lesk

This hybrid algorithm incorporates Lesk's algorithm and introduces a weight penalty for less frequent senses. Lesk's algorithm was implemented from scratch, comparing a word's context to each sense's definition in `nltk`. We assigned weights to the first 5 senses—[1, 0.6, 0.5, 0.4, 0.3]—which were multiplied by the overlap count. For any remaining senses, a weight of 0.3 was assigned. These weights were determined through accuracy testing in the development set. Lemmatization was also applied to both the context and definition sentences for each word. Although we experimented with removing stop words, similar to `nltk`'s Lesk algorithm, it appeared to slightly worsen the model's performance. After calculating overlaps for all senses, the sense with the highest count is selected as the best sense. Here, we present the result of this hybrid approach.

Method	Accuracy
Baseline + Lesk	0.52

The hybrid method demonstrates improved accuracy compared to Lesk alone but falls short of the baseline method. We initially expected this method to combine the strengths of both approaches, but it seems that this approach rather averages their performance.

5 Conclusion

In our specific dataset, Lesk's algorithm seems to perform poorly. Conversely, the bootstrapping algorithm significantly outperforms the other models. However, such a comparison is ultimately biased, as it would be unfair to compare a simple heuristic model against a semi-supervised model trained on a dedicated dataset.

References

[1] Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. Towards robust linguistic analysis using OntoNotes. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 143–152, Sofia, Bulgaria, August 2013. Association for Computational Linguistics.