Comparing Different Methods of Word Sense Disambiguation

Problem Setup

The goal of this project was to implement, compare and contrast different approaches to tackling the word sense disambiguation problem. I used four approaches. Two were versions of Lesk's algorithm, the first of which used the most frequent sense baseline and the second used NLTK's implementation. The other two involved implementing a supervised classifier for each word to be disambiguated. Both implementations of Lesk's algorithm used the SemEval 2013 Shared Task #12 dataset while both supervised classifiers used the SemCor corpus, a sense tagged subset of the Brown corpus.

Experimental Procedure and Parameter Settings

NLTK's Lesk's algorithm was not modifiable, hence there was no experimentation involved with that algorithm. There were two main aspects of Lesk's algorithm which I experimented with in my own implementation: the preprocessing on the input sentences and the method of determining overlap. For preprocessing I tested out lemmatization, stemming, leaving the sentences unchanged, and removing stopwords. In total, I experimented with 15 different preprocessing combinations.

A key part of the most frequent sense version of Lesk's algorithm is determining the overlap between a synset's definition and the context surrounding the word to be disambiguated. I experimented with two versions of computing overlap. The first version involved filtering out words that were found in both the context and sense definition meaning that the overlapping word could not be counted twice if it showed up again in the definition. The second version involved leaving overlapping words in the definition, thereby allowing the same word to contribute more than once to the overlap score. Filtering out overlapping words led to marginally better performance so this was the method I used. There was some other minor preprocessing needed for this implementation such as resolving collocated lemma's by replacing the underscore with a space before calculating overlap.

I chose to implement 12 different Multinomial Naive Bayes (MNB) classifiers, each of which was trained to disambiguate a single word. In order to decide which words to disambiguate, I found the 100 most common lemmas and their POS in the SemCor corpus, determined the accuracy that would be achieved if the lemma's most common sense was used to disambiguate it and picked 6 nouns and 6 verbs that had a wide range of baseline accuracies and were not stopwords. This information can be seen in Figure 1. I spent a lot of time experimenting with which features to use but eventually settled on a representation that contained enough information for the model but was not so complex that it led to lengthy training times. Let (..., w-2, w-1, w, w+1, w+2, ...) be the sentence context of the word w to be disambiguated, my feature representation was as follows: w-2, w-1, w+1, w+2, (w-2, w-1), (w-1, w+1), (w+1, w+2) where (x, y) represents the collocation of the word x and the word y with an "_". If an attribute was outside the range of the sentence then that attribute was filled with an "_". Once the features were vectorized I used a TfidfTransformer to balance the weight of high frequency features and low frequency features.

In order to compare the performance of my supervised classifiers I implemented 12 Decision Tree (DT) classifiers using the same features and words as the 12 MNB classifiers. However, unlike for the MNB classifiers I used bootstrapping to generate m random n sized subsets of the training set by sampling from the training set with replacement. I then used bootstrap aggregating to fit m Decision Tree models on the m subsets and aggregated the individual predictions using voting to form a final prediction. I used scikit-learn's BaggingClassifier to help me with this task and experimented with different values for m, settling on 50 as it gave the best balance of performance and efficiency.

In order to find the best hyperparameters for my supervised models I used scikit-learn's GridSearchCV to perform 5-fold cross validation on the training set (70% of the full dataset). For MNB, a smoothing value (alpha) of 1.0 and fit_prior=True led to the best performance. For the BaggingClassifier a max_samples value of 0.5 combined with a max_depth value of 5 for the DT classifier led to the best performance.

Results and Analysis

My implementation of Lesk's algorithm managed 53.31% accuracy, significantly outperforming NLTK's implementation which got an accuracy of 34%. Preprocessing was detrimental to accuracy as my best results were achieved with no preprocessing and removing stopwords. This surprised me but after some more experimentation I discovered that running my algorithm without computing overlap (i.e. always selecting the most frequent sense) led to an accuracy of 61.45%, significantly better than the versions that computed overlap. Therefore, it makes sense that no preprocessing led to a better accuracy as without preprocessing, words in the context and sense definitions with the same lemma but different endings would not contribute to the overlap score while with stemming and lemmatization they would. This means that the most frequent sense was more likely to be selected without preprocessing, leading to improved accuracy.

Both implementations of Lesk's algorithm shared similar failures and successes. One of the most notable failures was that both algorithms would often predict the synset "nation.n.02" for the lemma "country" when the correct synset was "state.n.04". The definitions for both senses are quite similar but this led to 16 failed predictions in NLTK's implementation and 17 in mine, leading to a non-trivial effect on the accuracy. Interestingly, this was not an issue in my implementation that always selected the most frequent sense. My algorithm performed best on lemmas with very few sense definitions. For example, all 13 instances of "united_states" or "usa" and all 21 instances of "player" were correctly disambiguated by my implementation. These lemma's only have 2 and 5 different senses respectively making them easier to disambiguate compared to a lemma like "game" with over 10 senses which my algorithm incorrectly disambiguated 20 times.

The results for the supervised classifiers varied. Five of the 12 words were best disambiguated by MNB, four were best disambiguated by bootstrapped DT, and three were disambiguated with equal accuracy by MNB, DT, and the most frequent sense baseline. See Figure 1 for more details on the results.

Bootstrapped DT seemed to outperform MNB when disambiguating words with fewer sense definitions. This may be explained by my choice to use a max_depth hyperparameter of 5 for the DT. Using a larger max_depth value would likely lead to improved performance as the model becomes more complex and can generalize to more situations like having many possible senses for the same lemma. However, it makes the model a lot costlier to train hence I settled on the value of 5. When the baseline accuracy was already high to begin with (meaning the most frequent sense was often the correct sense) my supervised models did not offer much improvement in terms of accuracy. This could possibly be fixed by using more complex features that incorporate the POS of different parts of the context as this information may be vital in terms of identifying a situation where the most frequent sense is not the correct choice. This approach was used by Escudero et al. in 2000 and led to large improvements in accuracy over the baseline.

It is also important to consider the possible effects that bootstrapping had on the results for my supervised classifiers. It is reasonable to assume that the DT classifier would perform worse without bootstrapping as there would be more variance and thus the model would have a worse fit on the data. If I had more time, I would have made two MNB and two DT classifiers with bootstrapping and without bootstrapping to gauge the effects it has on performance.

Lemma	say	make	know	take	use	find	man	time	year	day	thing	way
Multinomial Naive Bayes	**0.894**	**0.505**	0.604	**0.293**	0.849	**0.343**	**0.727**	**0.558**	**0.952**	**0.606**	0.235	0.516
Bootstrapped Decision Tree	**0.894**	0.422	**0.608**	0.216	**0.886**	0.289	**0.727**	0.462	**0.952**	0.566	**0.247**	**0.604**
Baseline Accuracy (test set only)	**0.894**	0.312	0.604	0.125	0.781	0.234	**0.727**	0.404	**0.952**	0.515	0.198	0.505
Frequency in SemCor	1950	1443	881	691	729	669	572	519	420	328	268	302
Number of WordNet Senses	11	49	11	42	6	16	11	10	4	10	12	12

Figure 1: 12 words to be disambiguated by supervised classifiers. Red columns are verbs, yellow columns are nouns.

References

Escudero, G., Màrquez, , & Rigau, G. (2000). Naive Bayes and Exemplar-Based approaches to Word Sense Disambiguation Revisited. *Proceedings of the 14th European Conference on Artificial Intelligence, ECAI,* 421-425. doi:arXiv:cs/0007011