Word Sense Disambiguation

Name: Anshita Saxena, Matricule Number: 20240044

1 Problem Setup

This study aims to create models and conduct experiments for the word sense disambiguation task. The approach involves using heuristic models like NLTK Lesk, a custom model implementation, and a semi-supervised model like Yarowsky's bootstrapping model. The practical significance lies in accurately recognizing word senses in conversations, social media, and websites. The research addresses questions about model baselines, custom implementation, and suggestions for improvement.

2 Dataset Generation

No extra data was needed for the most frequent baseline and Lesk models, both of which are heuristic approaches like my custom implementation. However, for the bootstrapping model, additional text collection was performed. Since ChatGPT data sources haven't been disclosed, the dataset used comes from NLTK Reuters Corpus and MASC Word Sense Sentence Corpus, excluding content from SemEval/SemCor datasets (BabelNet, Wikipedia, WordNet). NLTK Reuters only has sentences without senses, so I manually annotated a few of them with two types of senses. MASC Word Sense Sentence Corpus includes annotation information, such as senses, senseID, annotatorID, and sentenceID, spread across three files. I wrote preprocessing code to retrieve sentences with their senses. Table 1 in the Analysis Section records information for words and sentences. Special thanks to Saif Gazali for contributing to the additional text collection.

3 Analysis and Model Discussion

My custom implementation follows a heuristic approach, incorporating parameters such as context sentences, ambiguous words, part-of-speech tags, and boolean choices for hypernym/hyponym and cosine. Using the WordNet dictionary, it retrieves synsets and their definitions for the given senses. When the hypernym/hyponym boolean variable is true, it adds lemmas connected with the hypernym/hyponym of synsets. The implementation assesses overlaps by checking either the maximum word intersection or the maximum cosine similarity between the context and synset definitions to assign the sense. After experimenting with various models, we selected the one with the highest accuracy on the dev set and applied it to the test set for final predictions.
Inspired by the Yarowsky algorithm, the bootstrapping approach aims to identify ambiguous words in both the development and test sets, crucial for assessing model performance and refining classifier iterations (Multinomial Classifier) for accurate results on the test set. Initially, nine words ('Friday', 'united_states', 'year', 'deal', 'world', 'week', 'action', 'country', and 'time') were identified in both datasets, but some had only one sense. Table 1 details the final ambiguous words with multiple senses in both dev and test sets, along with word senses and the total sentences used in training. Notably, words like "power" and "execution" share senses with "major_power" and "death_penalty," respectively, as per WordNet. Sample output is illustrated in Figure 1. The selected senses were consistently applied to evaluate accuracy on the development set and subsequently employed on the test set. Each word's senses were labeled with 20 sentences, categorized into two senses (10 sentences each), and fed into the iterator. The classifier predicts unlabeled sentences and only those exceeding the user-provided confidence parameter are selected. In successive iterations, newly chosen sentences, along with the initially labeled ones, undergo classification. Accuracy is calculated at each iteration using the development set until performance stabilizes, and the final classifier is employed for predictions on the test set.

Word	Senses Used	Corpus Used	TS
deal	1:10:00::, 1:04:02::	Nltk RC	100
time	1:11:00::, 1:28:05::	MASC	214
power	1:14:00::, 1:07:02::	MASC	102
level	1:07:00::, 1:26:01::	MASC	217
sense	1:09:05::, 1:09:04::	MASC	217

Table 1: Selected words for bootstrapping, along with their senses, data sources (MASC for MASC Word Sense Sentence Corpus and Nltk RC for Nltk Reuters Corpus), and total sentence count (TS).

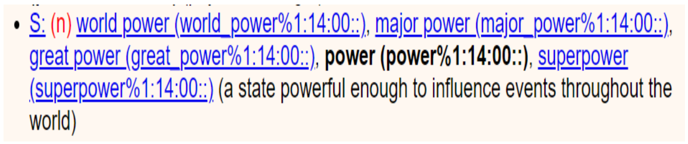
• S: (n) world power (world_power%1:14:00::), major power (major_power%1:14:00::), great power (great_power%1:14:00::), **power (power%1:14:00::)**, superpower (superpower%1:14:00::) (a state powerful enough to influence events throughout the world)

Figure 1. WordNet power and major_power sense

4 Results

All experiments and their outcomes are documented in the results_log_file.txt (78 experiments), included in the submission. Between the Most Frequent Sense model and Lesk model, the former outperformed the latter. The Lesk model achieved approximately half the accuracy, specifically 38%, for lemmatization only and without all three preprocessing steps (lemmatization, stop words removal, and punctuation removal) on the test set, compared to the Most Frequent Sense model, which achieved 62% accuracy on the test set. In the custom implementation, the model reached a maximum accuracy of 64% using the word intersection overlap technique and 56% using cosine similarity. The best accuracies for the bootstrapping model are outlined in Table 2 for various words. Table 3 displays the best results for the other three models.

Word	Parameters	Accuracy
deal	I=2, C=0.9	Dev: 70%, Test: 75%
time	I=5, C=0.9	Dev: 63.63%, Test: 66.66%
power	I=4, C=0.9	Dev: 91.67%, Test: 100%
level	I=3, C=0.7	Dev: 72.72%, Test: 50%
sense	I=4, C=0.9	Dev: 72.72%, Test: 100%

Table 2: Bootstrapping results for all words: "sense" and "deal" achieved this performance with preprocessing (lemmatization, punctuation removal, and stop words removal), while "time," "level," and "power" achieved this performance without any preprocessing. "I"=number of iterations, and "C"=confidence threshold for sentence selection.

5 Model Success and Failures with suggestions for Improvement

In experiments, it is noted that preprocessing (lemmatization, stop words, punctuation) has no impact on the most frequent baseline model because the lemma's sense is not influenced by these factors. However, for the NLTK Lesk model, applying preprocessing to the context proved successful in reducing word sparsity. Since overlap calculations in NLTK consider words, this preprocessing aids in maximizing the word intersection between the context and definition, resulting in increased accuracy. In my custom implementation, which follows Lesk intuition learned in class, calculating the intersection of word sets proved more effective than using cosine similarity. Additionally, the inclusion of hypernym and hyponym, defining the superset and subsets of lemmas, did not contribute to accuracy improvement in either case—word intersection or cosine similarity.

Carefully labeling a small dataset with a balanced, large number of unlabeled sentences, categorized between two senses (not necessarily having the exact same sense), yields good accuracy. Conversely, an unbalanced number of example sentences between two senses does not result in high accuracies. Experiments indicate that preprocessing plays a crucial role, boosting accuracy for the word "time" from 50% to 66.67%. In experiments with the word "level," it was observed that reducing the confidence level of the prediction score for sentences increases accuracy but may introduce inconsistency when applying the algorithm to an unseen dataset.

To enhance model performance, one can improve by incorporating more example sentences with up-to-date definitions and senses, given the rapid updates in languages. Adding more surrounding words, such as longer sentences, enriches the context for disambiguation. Incorporating sense frequency counts allows for assigning more weight to specific senses. Given additional time for this assignment, exploring hybrid models that combine multiple approaches becomes a viable possibility.

Model	Parameters	Accuracy
Most Frequent Model	No preprocessing used (Lemmatize, stop words, punctuation)	Dev: 67.525% Test: 62.344%
Nltk Lesk Model	1.lemmatization 2.without lemmatization, stop words removal, removed punctuation	1.Dev:37.628% Test:38.206% 2.Dev:38.659% Test:38.413%
Custom Model using word intersection and cosine similarity for overlap	Lemmatized, stopwords removal, punctuation removal, word intersection overlap, without hypernym/hyponym	Dev: 64.432% Test: 57.655%

Table 3: Recorded results for all three models.

6 REFERENCES

1. Wordnet
2. Wordnet Tool
3. Lesk Algorithm
4. WordNet Word Structure
5. WordNet Sense Structure
6. Yarowsky Algorithm
7. Word Sense Disambiguous Dataset Article
8. Word Sense Disambiguous Dataset Information
9. MASC Word Sense Sentence Corpus, tab-separated format
10. Nltk Corpus Information