Assignment 2

Enzo Benoit-Jeannin

I. Introduction

This report presents an analysis of Word Sense Disambiguation (WSD) methods as applied to the SemEval 2013 Shared Task #12 dataset [1] and the FEWS dataset [2], highlighting the development and evaluation of a baseline and other WSD techniques.

II. Methodology

The transition from Python 2 to Python 3 necessitated a key modification in the to_ascii function to handle string encoding properly, ensuring contexts were strings instead of byte sequences.

A. Pre-Processing

As suggested in the assignment guidelines, the preprocessing steps involve tokenization, stop word removal, and lemmatization, with a focus on maintaining words that contain either periods "." or hyphens "-" like "u.n". The process also lowers all text to ensure uniformity. This step is performed for both the SemEval 2013 Shared Task dataset and the FEWS dataset.

B. Baseline Method and Lesk's Algorithm

We first evaluated two methodologies: the Baseline method, which selects the most frequent sense of WordNet, and the implementation of Lesk's algorithm in the NLTK library. A key step in model evaluation involved converting dataset labels from "word%1:03:00" format (lemma sense keys format which encodes part of speech, category, and sense) to WordNet v3.0's "word.n.03" format (synset numbers). We developed a function to parse lemma sense keys and match them to WordNet synset numbers, ensuring accurate model performance assessment for these two models. Moreover, the given dataset occasionally featured keys with multiple senses. To address this, we used the development set to compare the efficacy of using all listed senses against the first sense in each key, alone. Our findings, detailed in the results section, revealed that incorporating all possible senses for model evaluation yielded higher performance metrics. This is logical, as including more potential labels increases the likelihood of correctly matching at least one of the senses, thus enhancing the model's accuracy.

C. Bootstrapping Method

For our bootstrap approach, we identified the six most frequent words to disambiguate in the test set (which were *game, year, player, team, case, and country*) to construct six separate models based on Yarowsky's algorithm [3]. To follow Yarowsky's methodology, we needed to define a seed set for each of our models. Given the rarity of these words in the given test set, we opted to create a custom seed set to have as many words as possible to test the method. Utilizing ChatGPT 3.5 [4], we generated a dataset of 200 sentences for each word, ensuring representation of all possible senses as defined in WordNet. The prompt used for each word was as follows: *"I need to create a seed set for the word: ... . This seed set should contain sentences showcasing different senses of the word. This word has the following possible senses: ... . Please produce a CSV file with at least 200 sentences, each labeled according to these WordNet tags."*. This approach allowed us to build a robust seed set for each word, which is essential for the bootstrap method to perform effectively. For each of the six words, we trained a logistic regression model using the sklearn library [5], following Yarowsky's methodology. The process involved training each model on its respective seed set, then predicting on the test set. Predictions with a probability above 0.6 were considered confident and added to the seed set. This cycle of training and augmenting the seed set was repeated, either until reaching a maximum number of iterations which we defaulted to 15, or until all test examples were classified with confidence exceeding the 0.6 threshold.

D. Supervised Learning Method

In contrast to the previous methods, we explored a fully supervised learning approach using the FEWS dataset [2], which provides extensive training examples and labels for Word Sense Disambiguation task. By following the assignment guidelines and for performance optimization, only 1MB of this dataset was utilized (out of the 20MB available). We fine-tuned the "bert-base-uncased" model from the Hugging Face library and used its tokenizer. An important feature of the dataset is the use of $< WSD > ... < /WSD >$ markers to denote the words requiring disambiguation. Moreover, the labels conform to the synset numbering format we discussed in class. We partitioned the 1MB dataset into 80%

for training, with the remaining 20% equally divided between validation and testing. The model was trained for one epoch, using a batch size of 16 for both training and evaluation phases.

III. Results and Discussion

Throughout our evaluation, accuracy served as the metric for assessing model performance.

A. *Baseline vs. Lesk's Algorithm*

The comparative analysis between the baseline and Lesk's algorithm yielded insightful results. The baseline model, attained a surprisingly high accuracy of 60% on the test set. On the other hand, Lesk's algorithm demonstrated a much lower accuracy of 19% on the test set. It's also worth noting that these accuracy results consider all possible senses in the dataset keys when computing the accuracy. Indeed, when only the first sense in each key is considered, the accuracy of the baseline model is 60% on the validation set, and the accuracy of the Lesk's algorithm is 22%. On the other hand, when considering all possible keys, the validation accuracy changes only for the baseline model which increases to 62%. Therefore, we chose to evaluate the models using all senses in the given keys. An example of model outputs can be shown here: "Lesk's: united_states_government.n.01; Baseline: united_states_government.n.01; Actual: ['u.s.%1:15:00::']". The baseline method, while straightforward, does not account for contextual nuances. Lesk's algorithm, on the other hand, faces challenges due to its reliance on dictionary definitions, which may not accurately reflect the real-world context of word usage. A possible enhancement could be to include the definitions of words found within the primary definition of the target word. This could better reflect the context of a word.

B. *Bootstrapping Method*

The bootstrapping approach showed very contrasting results: each model achieved either 0% or 100% accuracy on their corresponding test set. The test set for each model was specific to instances where the target word matched the one the model was trained on, resulting in relatively small test sets. A notable issue with bootstrapping was its tendency to get 'stuck'. For instance, in the test set for the word 'country', the majority of instances shared the same sense, leading to a repetitive pattern in the model's predictions. Once the model confidently labeled a single instance, it tended to apply the same label to all subsequent similar contexts, culminating in extreme accuracy outcomes. This suggests a potential limitation of the bootstrapping methodology in contexts where sense diversity is limited. An important thing to note is that this experiment was limited to the size of the test set and future explorations could include using a bigger one. Moreover, we could try to stop the algorithm main loop by checking if the model parameter freeze or not instead of a maximum number of iterations.

C. *Supervised Learning Method*

The supervised learning method, yielded more promising results with an accuracy of 42% on the validation set and 37% on the test set. Note that this approach is not directly comparable to the previous models implemented as it was trained on a different dataset. In the context of Word Sense Disambiguation, fine-tuning in supervised learning offers the advantage of transferring learned features from large, diverse datasets potentially enhancing model performance with relatively less data. A potential avenue for future work could include utilizing the entire FEWS dataset, which might allow the model to capture a broader range of language nuances and improve its ability to accurately disambiguate word senses. However, this might require a lot of computational power.

References

[1] R. Navigli, D. Jurgens, and D. Vannella, "SemEval-2013 task 12: Multilingual word sense disambiguation," in *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, S. Manandhar and D. Yuret, Eds. Atlanta, Georgia, USA: Association for Computational Linguistics, Jun. 2013, pp. 222–231. [Online]. Available: https://aclanthology.org/S13-2040

[2] T. Blevins, M. Joshi, and L. Zettlemoyer, "Fews: Large-scale, low-shot word sense disambiguation with the dictionary," in *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics*, 2021. [Online]. Available: https://blvns.github.io/papers/eacl2021.pdf

[3] D. Yarowsky, "Unsupervised word sense disambiguation rivaling supervised methods," in *33rd Annual Meeting of the Association for Computational Linguistics*. Cambridge, Massachusetts, USA: Association for Computational Linguistics, Jun. 1995, pp. 189–196. [Online]. Available: https://aclanthology.org/P95-1026

[4] OpenAI, "Chatgpt 3.5 (september 25 2023 version) [large language model]," https://chat.openai.com.

[5] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.