Naive *Named Entity Recognition (NER)* Implementation

Yinan Gao, Joshua Lee

November 2024

Introduction

Named Entity Recognition (NER) is a critical task in Natural Language Processing, where the objective is to identify and classify entities within a text into predefined categories such as persons, organizations, locations, and more. It serves as a foundational component in many downstream applications, such as information retrieval, question answering, and knowledge extraction [2]. The task is challenging due to the variability in how entities are mentioned in natural language (e.g., abbreviations, synonyms, or cultural variations).

Dataset: CoNLL-2003

In this naive implementation, we used the CoNLL-2003 dataset[3], a widely recognized benchmark in NER. The CoNLL-2003 dataset was introduced as part of the shared task in the Conference on Computational Natural Language Learning (CoNLL) 2003. It includes texts from the Reuters Corpus and contains named entity annotations for four categories:

- **PER**: Person names
- **ORG**: Organizations such as companies, teams, or government bodies
- **LOC**: Geographical locations such as cities, countries, or landmarks
- **MISC**: Miscellaneous entities, such as nationalities, political affiliations, or other unique mentions that do not fall into the first three categories

Each entity is tagged using the BIO tagging scheme, where tags such as `B-PER` and `I-PER` denote the beginning and inside of a person entity, respectively. Tokens outside any named entity are marked as `O`.

Dimensions and Statistics

The dataset is divided into three parts: training, validation (development), and test sets. Below are its key dimensions:

- **Training Set**: Contains 14,987 sentences with approximately 203,621 tokens.
- **Validation Set**: Includes 3,466 sentences with approximately 51,362 tokens.
- **Test Set**: Comprises 3,684 sentences with approximately 46,435 tokens.

Evaluation Metrics

In evaluating our NER implementation, we used accuracy, precision, recall, and F1-score. Since the output of NER is not simply binary (positive/negative), we define: if entity is detected (i.e. type returned is not O), it's positive; else it's negative. If the prediction matches the actual type, it's a true case. Thus, we were able to calculate the classification metrics for NER models.

In general, accuracy gives an overall measure of correct predictions, while precision and recall provide insight into the Conservative or aggressive nature of the model when labeling specific entities. The F1-score, as a balance between precision and recall, helps dealing with imbalanced datasets. Additionally, since we have those values for each entity type, we got to know how the algorithm is performing for certain entity types.

Implementation Approaches and Results

Since our naive implementation considers input texts as discrete single words, we took of the BI tagging of the data when loading them. Then, we implemented a probabilistic classification using word frequencies based on the training set. This dictionary is referred to when classifying a word that exists in the training data. Additionally, incorporating the gensim word embedding module, we were able to generate an averaged word vector for each entity type based on this dictionary. Thus, when encountering a new word that is not present in the training data, the model checks its similarity to the averaged entity-type vectors using cosine similarity, and classifies it according to the similarity values.

The general performance metrics of testing this model on the test set of CoNLL-2003 is summarized in table 1. With an accuracy of 0.9, the naive model decently predicts the given words in the test set. Besides that, we also tried to generate performance metrics for each type of entity, and the result is listed in table 2. Here we can see the prediction of organizations is the worst and is holding the general performance of the model back. The reason here we think is that the

name of organizations in the training set utilizes a lot of daily words, thus the averaged word vector for organizations in our model deviates more toward daily words, causing the model to recognize a lot of normal words (O) as organizations (ORG). The low precision but higher recall value for ORG is consistent with this speculation.

Metric	**Accuracy**	**Precision**	**Recall**	**F1-Score**
Value	0.9028	0.5924	0.8516	0.6987

Table 1: Performance Metrics for Naive NER

Entity	**Precision**	**Recall**	**F1-Score**
LOC	0.7809	0.7387	0.7592
MISC	0.7871	0.5839	0.6704
PER	0.8306	0.6152	0.7069
ORG	0.3670	0.6294	0.4636

Table 2: Performance metrics for each entity in Naive NER.

Third Party Tool: SpaCy

SpaCy is a python natural language processing model, which has a mature implementation of NER. SpaCy's NER pipeline uses a statistical model based on machine learning to recognize entities, exploiting pre-trained model en_core_web_sm. These models rely on word embeddings and deep learning architectures like Convolutional Neural Networks (CNNs) or Transformers to make predictions about the entities in a sentence.

While it performs quite well identifying named entities in daily texts, it includes too many classifications of entities than that included in CoNLL-2003. The entity types SpaCy would return includes but is not limited to: NORP, PRODUCT, EVENT, WORK OF ART, LANGUAGE, PERCENT, MONEY, TIME, etc. Therefore, we had to perform a transformation of its output to make sure the format matches up when testing it on CoNLL-2003 test set. Based on the way the CoNLL-2003 dataset is labeled, we marked the "NOAP" and "LANGUAGE" types in SpaCy as "MISC", and labeled both "LOC" and "GPE" in SpaCy as "LOC" in CoNLL. The performance metrics of testing the transformed SpaCy NER model on CoNLL2003 test set is listed in table 3.

Metric	**Accuracy**	**Precision**	**Recall**	**F1-Score**
Value	0.9146	0.7219	0.6981	0.7098

Table 3: Performance Metrics for SpaCy NER

Discussion and Comparison of Results

In general, the naive NER and SpaCy NER achieved similar accuracy values on the CoNLL-2003 test set. However, the precision and recall scores reveal important insights into the strengths and weaknesses of both methods. The naive NER exhibits significantly lower precision compared to recall, indicating that the model is overly aggressive, misclassifying too many outside (O) words as entities. This issue is likely related to the word embedding algorithm used in the implementation of word classification. To address this, a potential improvement for the naive model could involve adjusting its classification preference to favor outside (O) labels more conservatively. In contrast, SpaCy NER demonstrates a better balance between precision and recall, which can be attributed to its use of pre-trained embeddings and deep learning models.

One of the primary reasons SpaCy outperformed the naive NER implementation is its ability to leverage deep learning models that consider the context within a document when classifying words. Context is essential for resolving ambiguity. For example, the word "APPLE" could refer to an organization or a fruit, depending on the surrounding text. Since the naive model cannot consider contextual information, its ability to correctly label entities is significantly limited. Moreover, the gensim Word2Vec model used in the naive approach does not handle capitalized words, forcing all input text to be converted to lowercase. This is problematic, as capitalization often carries crucial information about an entity's type. In this respect, SpaCy's ability to account for capitalization provides a distinct advantage.

Lastly, it is important to note that the comparison between the naive NER and SpaCy NER is not entirely fair. The naive NER is trained on the same dataset it is evaluated on, while SpaCy NER relies on pre-trained models. As a result, the naive NER's performance on the test set is likely overestimated, as it benefits from potential overfitting to the CoNLL-2003 data. Conversely, SpaCy NER's evaluation is limited to the four entity types in the CoNLL-2003 dataset, despite being capable of identifying a much broader range of entity types in real-world applications. This constraint likely underestimates SpaCy's true potential in Named Entity Recognition tasks.

Considering these factors, it is clear that SpaCy NER is far superior to the naive NER for performing real-world NER tasks, as it is better equipped to handle the complexities and nuances of natural language.

Bias Discussion

Both the naive NER and SpaCy NER are susceptible to biases that could impact their performance in real-life tasks. The naive NER, relying on word embeddings generated from the training dataset and gensim, may struggle with out-

of-vocabulary words or entities that deviate from those seen during training, leading to poor generalization in diverse linguistic contexts. Additionally, its reliance on lowercased inputs removes critical features like capitalization, which may disadvantage certain languages or cultural naming conventions. SpaCy, while more robust due to contextual awareness and better ability to handle word cases, is still subject to biases inherent in its training data, such as over-representation of Western-centric entities or systemic stereotypes embedded in large corpora. Both models could disproportionately underperform on texts from underrepresented languages, dialects, or domains, potentially perpetuating biases in applications like hiring, legal document analysis, or public policy decisions[1].

References

[1] Dartmouth College. Zeroing in on the origins of bias in large language models, 2024. Accessed: 2024-11-25.

[2] DataCamp. What is named entity recognition (ner)?, 2024. Accessed: 2024-11-25.

[3] E. F. Tjong Kim Sang and F. De Meulder. Introduction to the conll-2003 shared task: Language-independent named entity recognition, 2003. Accessed: 2024-11-25.