Dhanya Srinivasan
951660903
dps6276@psu.edu

Midterm Project – Named-Entity Recognition Task

Problem Definition:

Named-Entity Recognition (NER) is a task in Natural Language Processing (NLP) that classifies named entities into predefined categories such as person names, organizations, locations, medical codes, time expressions, quantities, monetary values, and more. NER's primary objective is to comb through unstructured text and identify specific chunks as named entities, subsequently classifying them into predefined categories. This conversion of raw text into structured information makes data more actionable, facilitating tasks like data analysis, information retrieval, and knowledge graph construction.

Dataset:

For this task, we make use of the CoNLL-2003 dataset, which was released as a part of CoNLL-2003 shared task: language-independent named entity recognition. The data consists of eight files covering two languages: English and German. For each of the languages there is a training file, a validation file, a test file and a large file with unannotated data. For this task, we make use of the training, validation and test data in the English language.

The English data was taken from the Reuters Corpus. This corpus consists of Reuters news stories between August 1996 and August 1997. For the training and development set, ten days worth of data were taken from the files representing the end of August 1996. For the test set, the texts were from December 1996. The preprocessed raw data covers the month of September 1996.

The CoNLL-2003 shared task data files have four columns separated by one space. Each word has its own line, and each phrase ends with an empty line. Each line begins with a word, followed by a part-of-speech (POS) tag, then a syntactic chunk tag, and finally a named entity tag. The chunk tags and named entity tags use the form I-TYPE, which indicates that the word is contained within a phrase of type TYPE. Only if two phrases of the same type immediately follow each other will the first word of the second phrase include the tag B-TYPE to indicate that it begins a new phrase. A word with the tag O is not part of a sentence. This HuggingFace dataset, however, uses the IOB2 tagging system, whereas the original dataset employs IOB1.

An example of 'train' looks as follows.

Dhanya Srinivasan
951660903
dps6276@psu.edu

```
{

    "chunk_tags": [11, 12, 12, 21, 13, 11, 11, 21, 13, 11, 12, 13, 11, 21,
22, 11, 12, 17, 11, 21, 17, 11, 12, 12, 21, 22, 22, 13, 11, 0],

    "id": "0",

    "ner_tags": [0, 3, 4, 0, 0, 0, 0, 0, 0, 7, 0, 0, 0, 0, 0, 7, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],

    "pos_tags": [12, 22, 22, 38, 15, 22, 28, 38, 15, 16, 21, 35, 24, 35,
37, 16, 21, 15, 24, 41, 15, 16, 21, 21, 20, 37, 40, 35, 21, 7],

    "tokens": ["The", "European", "Commission", "said", "on", "Thursday",
"it", "disagreed", "with", "German", "advice", "to", "consumers", "to",
"shun", "British", "lamb", "until", "scientists", "determine", "whether",
"mad", "cow", "disease", "can", "be", "transmitted", "to", "sheep", "."]

}
```

The data fields are described as follows:

- id: a string feature.
- tokens: a list of string features.
- pos_tags: a list of classification labels (int).
- chunk_tags: a list of classification labels (int).
- ner_tags: a list of classification labels (int)

The data split is as follows:

- train: 14041 examples
- validation: 3250 examples
- test: 3453 examples

Methodology:

1. **Pre-trained Word Embeddings:**
   To generate word embeddings, we make use of the Word2Vec model, which has been pre-trained on the Google News dataset. This model was downloaded using the GenSim downloader.

2. **Vocabulary and Tag Mapping:**
   As a first step, we define mappings for vocabulary words and NER tags.

   a. **Vocabulary Mapping:**
      - We initialize a vocabulary dictionary with two special tokens:
        - "<PAD>": Assigned index 0 (used for padding shorter sequences).

    - "<UNK>": Assigned index 1 (used for unknown words).
- We then update the vocabulary with words from the training dataset:
    - We extract all unique words from the train split of the dataset.
    - If a word exists in the Word2Vec model, it is assigned a unique index (idx + 2 to leave space for <PAD> and <UNK>).
    - Otherwise, the word is mapped to <UNK> (index 1).

b. **Tag Mapping:**
We map numeric indices to NER tags:
- 'O': No entity.
- 'B-PER', 'I-PER': Beginning and Inside of a **Person** entity.
- 'B-ORG', 'I-ORG': Beginning and Inside of an **Organization** entity.
- 'B-LOC', 'I-LOC': Beginning and Inside of a **Location** entity.
- 'B-MISC', 'I-MISC': Beginning and Inside of a **Miscellaneous** entity.

Using the above vocabulary and tag mappings, we create an embedding matrix with an embedding dimension of 300.

3. **Data pre-processing:**
Based on the vocabulary mapping generated earlier, the words are tokenized, and the NER tags are converted into tensors. These token and tag sequences are all padded to be the same length, where tokens are padded with 0 and NER tags are padded with -1.

4. **NER Model and Training:**
We create a simple Convolutional Neural Network (CNN) with 2 convolutional layers and one last fully connected linear layer. This NER model is trained with

- 100 convolutional filters in each convolutional layer.
- Filter sizes: 3, 5, 7

The training data is taken in batches of size 32, and the model is trained for 100 epochs. We use cross-entropy loss and an Adam optimizer with a default learning rate of 0.00001. At every epoch, the model is evaluated using the validation dataset.

5. **Evaluation of model:**
Lastly, the model is evaluated using the test dataset.

In-Depth Analyses/Experiments:

1. **Word2Vec Embeddings Analysis:**
As already explained, we use pre-Word2Vec embeddings to represent input text. Word2Vec was trained on a large corpus of text, using the skip-gram model. The

Dhanya Srinivasan
951660903
dps6276@psu.edu

embeddings capture semantic relationships between words. We visualized the embeddings using t-SNE, and the results showed that semantically similar words were clustered together. For instance, names of people like "Pedro" and "Janet" were grouped closely, indicating that Word2Vec effectively captured semantic meaning. The t-SNE plot is shown in Figure 1.

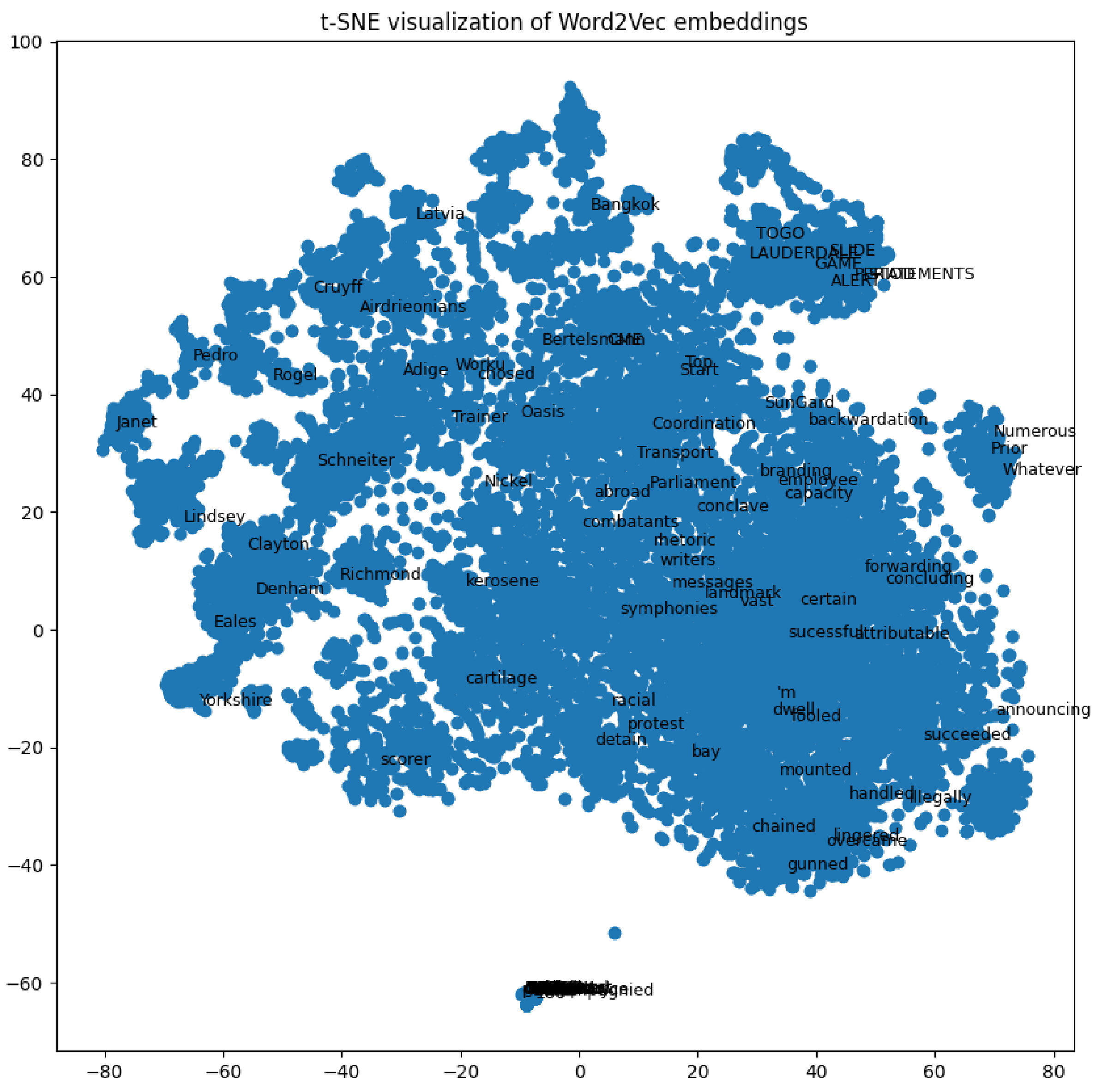

Figure 1: t-SNE visualization of Word2Vec embeddings

2. **Training vs Validation Loss:**
As we know, when the model begins to overfit, the validation loss begins to increase from its lowest point. To determine the optimal setup, where the model is not overfit on the training data, we trained the model for 15 epochs, with a learning rate of 0.001, and then plotted the training and validation losses against each other. As we can see in

Dhanya Srinivasan
951660903
dps6276@psu.edu

the Figure 2, the validation loss begins to increase after reaching its lowest point at the 5th epoch.

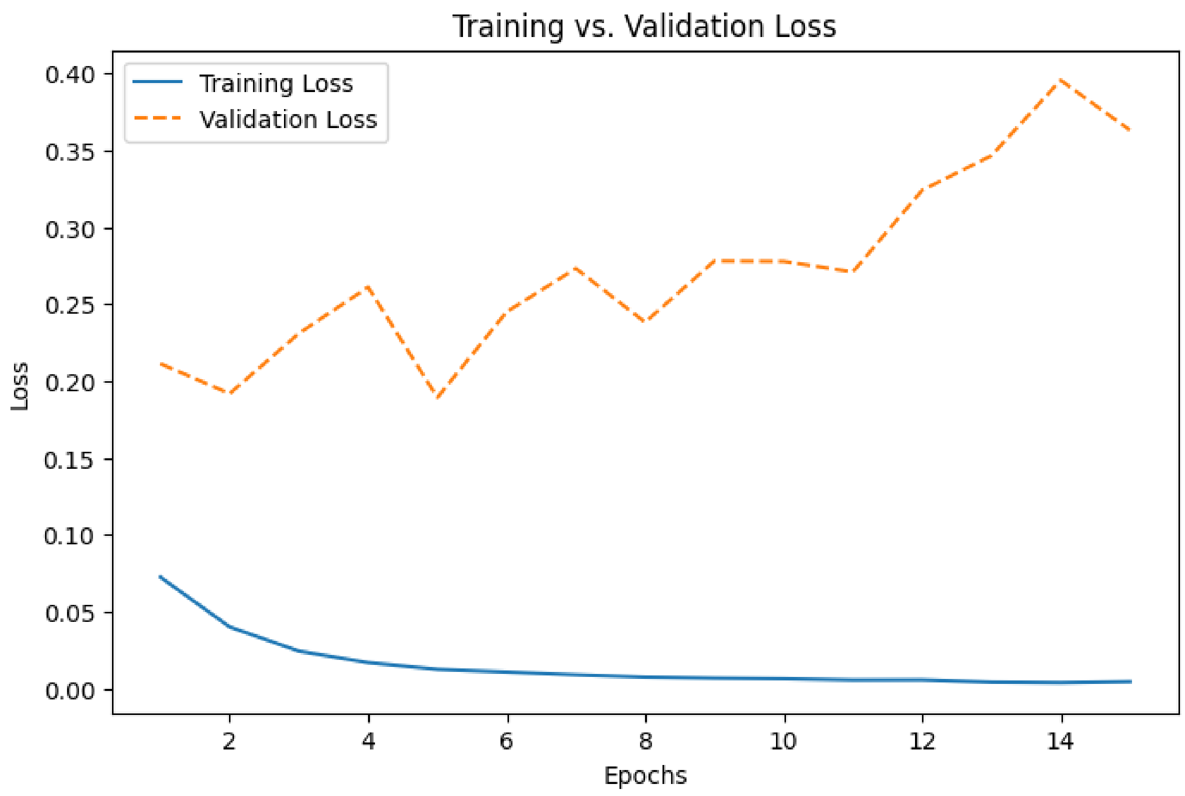

Figure 2: Training vs Validation Loss when NUM_EPOCHS=15 and LR=0.001

Hence, we try a few things to find the optimal setup.

First, we decrease the learning rate to 0.0001 and perform the same training for 15 epochs. As we can see in Figure 3, it looks slightly better in terms of learning behaviour in comparison to Figure 2. The initial training loss drops much more smoothly in comparison, and the validation loss remains lower for the first few epochs before starting to increase.

Next, we decrease the learning rate further to 0.00001 and perform the same training for 15 epochs. As we can see in Figure 4, the plot looks significantly better in comparison to Figure 2 and Figure 3. Both the training and validation loss drop significantly smoother than before, and brings us to the conclusion that we can train for a few more epochs, since both training and validation loss seem to be reducing at the 15th epoch. We then train with the same learning rate for 100 epochs, which is depicted in Figure 5.

Dhanya Srinivasan
951660903
dps6276@psu.edu

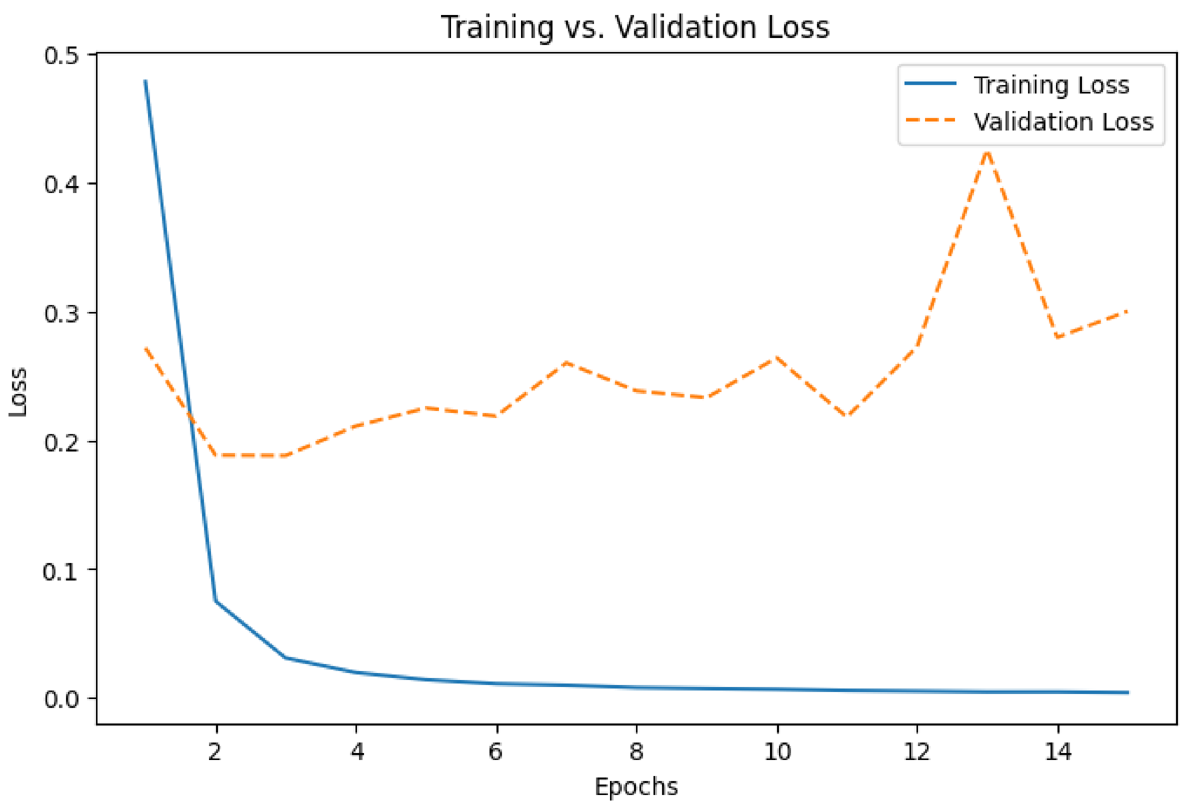

Figure 3: Training vs Validation Loss when NUM_EPOCHS=15 and LR=0.0001

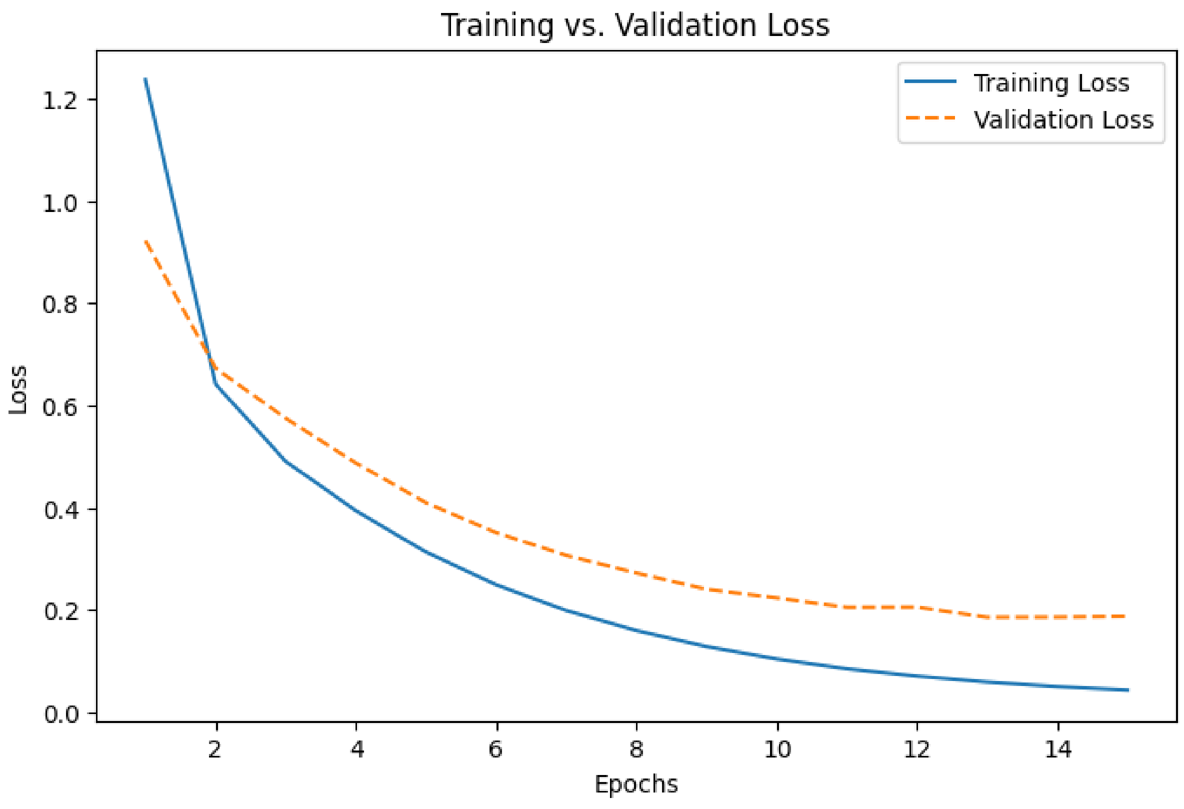

Figure 4: Training vs Validation Loss when NUM_EPOCHS=15 and LR=0.00001

Dhanya Srinivasan
951660903
dps6276@psu.edu

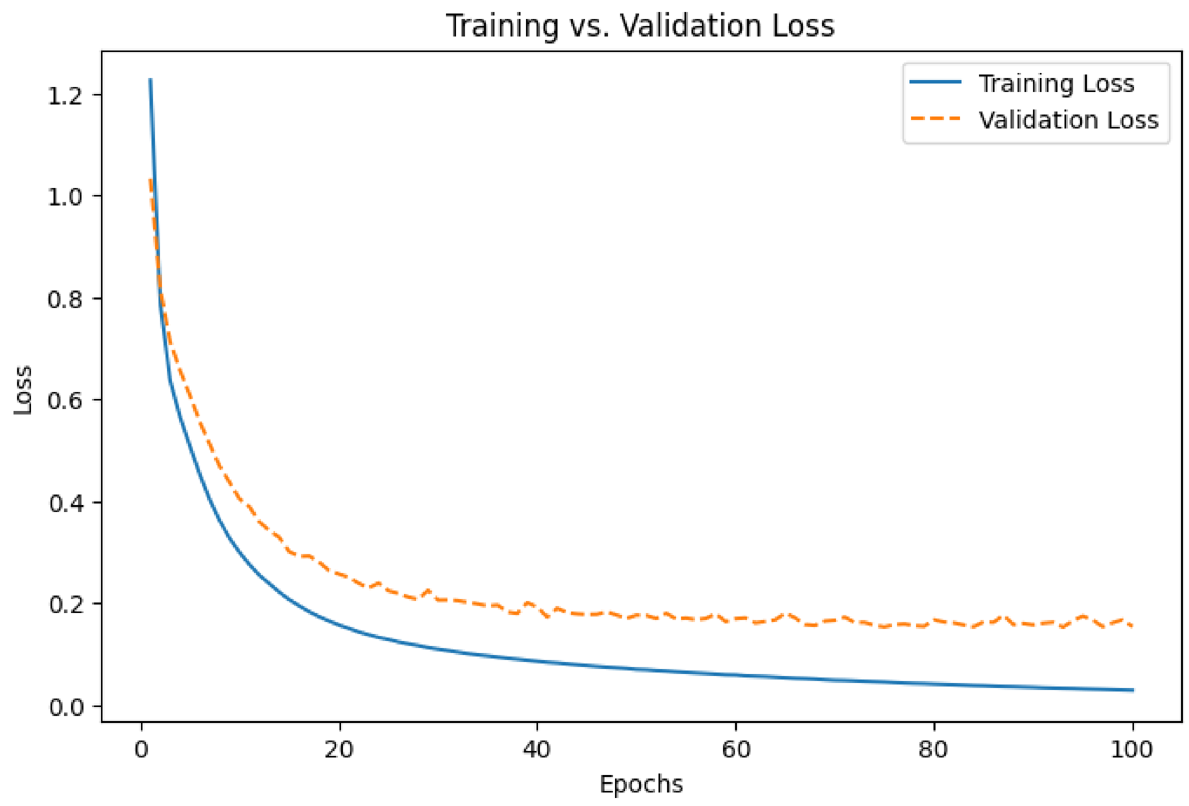

Figure 5: Training vs Validation Loss when NUM_EPOCHS=100 and LR=0.00001

Results:

We evaluate our model based on the following metrics:

1. Precision:

- Precision measures how many of the predicted entities were actually correct.
- Formula: Precision = TP / (TP + FP)
- High precision means fewer false positives.

2. Recall:

- Recall measures how many of the actual entities were correctly identified by the model.
- Formula: Recall = TP / (TP + FN)
- High recall means fewer false negatives.

3. F1-score:

- The F1-score is the harmonic mean of precision and recall, balancing both metrics.
- Formula: F1 = 2 * (Precision * Recall) / (Precision + Recall)
- A higher F1-score indicates a better balance between precision and recall.

Dhanya Srinivasan
951660903
dps6276@psu.edu

4. Accuracy:

- Accuracy measures the percentage of correctly classified tokens (including non-entity tokens).
- Formula: Accuracy = (Correct Predictions) / (Total Predictions)
- While accuracy can be useful, it may not fully reflect model performance in NER, as most words belong to the "O" class.

The results are presented in Table 1 below.

Classes	**Precision**	**Recall**	**F1-Score**
O	0.96	0.99	0.97
B-PER	0.92	0.67	0.78
I-PER	0.95	0.70	0.80
B-ORG	0.85	0.68	0.75
I-ORG	0.85	0.65	0.73
B-LOC	0.89	0.84	0.86
I-LOC	0.83	0.63	0.71
B-MISC	0.77	0.72	0.75
I-MISC	0.68	0.61	0.64
Accuracy			0.94

Table 1: Results of our model

As we can see in Table 1 and Figure 6, the model achieves an overall high accuracy of 0.94. However, accuracy alone is not a good metric to evaluate the NER task, as class O dominates all the other classes, whose performance is denoted by the F1 score of 0.97. The other entities, such as B-PER, I-PER, B-ORG, I-ORG, etc., however, have a lower score, suggesting that the model struggles a little bit with these classes.

For example, the B-PER class has a precision of 0.92 but a recall of 0.67, leading to an F1-score of 0.78. This suggests that the model is precise when predicting B-PER (few false positives) but misses many actual person entities (high false negatives).

The model performs better on B-LOC (F1 Score= 0.86) but struggles with I-LOC (F1 Score= 0.71), meaning it identifies locations well but struggles when they span multiple words.

Dhanya Srinivasan
951660903
dps6276@psu.edu

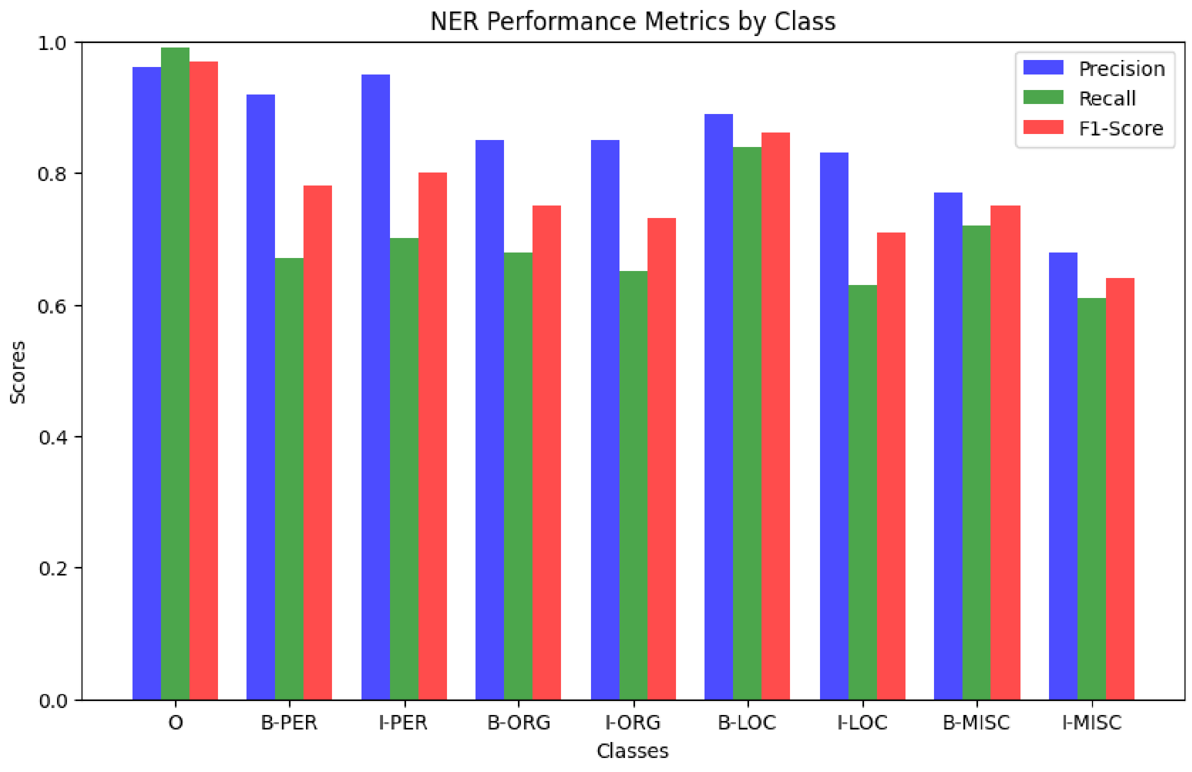

Figure 6: NER Performance Metrics by Class

Adding more convolutional layers to the CNN will help with improving in recognition of entities with a lower occurrence (entities apart from Objects (class O)). Moreover, training with a much lower learning rate and greater number of epochs will also help improve performance.

Lessons and Experience:

1. The importance of word embeddings in capturing meaningful word representations was one important lesson learned. We improved the model's recognition of named entities by utilizing pre-trained knowledge through the use of Word2Vec embeddings. Handling out-of-vocabulary (OOV) words is still difficult, nevertheless, because invisible words can result in incorrect classification.

2. CNNs are more frequently utilized in image processing, but this experiment showed that they are also capable of handling text-based tasks. The model effectively captured local context by applying numerous convolutional filters of varying sizes. CNNs might have trouble with long-range dependencies in contrast to more conventional sequence models like LSTMs.

3. Hyperparameter tuning affects model performance.

- High learning rates caused unstable training, leading to fluctuating loss values and difficulty in reaching an optimal solution.

Dhanya Srinivasan
951660903
dps6276@psu.edu

- Very low learning rates (e.g., 0.0001) resulted in slow convergence, requiring significantly more epochs to achieve reasonable performance.

4. Because there were a lot of non-entity (O) tokens in the sample, the model was more likely to predict the "O" class. For rare entity types (such as "I-MISC" and "I-LOC"), this resulted in decreased precision and recall. Class weighting or data augmentation are two potential solutions to lessen this problem.

References:

1. https://www.datacamp.com/blog/what-is-named-entity-recognition-ner

2. https://paperswithcode.com/dataset/conll-2003

3. https://huggingface.co/datasets/eriktks/conll2003

4. https://medium.com/@manansuri/a-dummys-guide-to-word2vec-456444f3c673