Fine-Tuning BERT Model for Named Entity Recognition Natural Language Processing @HCMUS-2024

Lam Gia Phu - 21280104 [1 2] **Tran Ngoc Khanh Nhu - 21280040** [1 2]

Abstract

Named Entity Recognition (NER) is a crucial task in Natural Language Processing (NLP) that involves identifying and classifying named entities in text into predefined categories such as person names, organizations, locations, and more. This study addresses several prevalent challenges in NER using a fine-tuning approach with BERT (Bidirectional Encoder Representations from Transformers) on the CoNLL-2003 dataset. Key problems such as ambiguity and polysemy, entity boundary detection, variability and synonymy of entities, and out-of-vocabulary words are tackled through the powerful contextual understanding of BERT. The fine-tuning process involves aligning tokenized inputs with their respective labels, handling special tokens, and ensuring accurate label propagation through subwords. This research demonstrates how these techniques effectively resolve common NER issues and underscores the potential of transformer-based models in enhancing the accuracy and robustness of NER systems.

1. Introduction

Named Entity Recognition (NER) is a crucial task in Natural Language Processing (NLP) that involves identifying and categorizing key information in texts, such as names of people, locations, organizations, and other significant entities. Accurate NER is vital for various applications, including information extraction, question answering, and enhancing search engines. The ability to automatically detect and categorize these entities can significantly improve the understanding and usability of textual data.

Traditional NER methods, such as rule-based approaches and feature engineering combined with machine learning algorithms, have achieved reasonable performance. However, these methods often struggle with the complexity and variability of natural language, especially when dealing with diverse and large datasets. Recent advancements in deep learning, particularly the introduction of Transformer-based models like BERT (Bidirectional Encoder Representations from Transformers), have revolutionized the field by providing powerful tools to capture contextual information and dependencies within text.

BERT, a pre-trained transformer model, has demonstrated remarkable performance across numerous NLP tasks. By pre-training on a large corpus of text and fine-tuning for specific tasks, BERT effectively captures the nuances and context of language. Fine-tuning BERT for NER leverages the model's pre-trained linguistic knowledge and focuses on the specific features of the NER task.

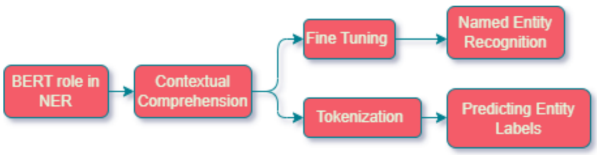

Figure 1. Architecture overview of BERT: Presenting a bidirectional encoder from Transformers for NER

This report presents a comprehensive approach to fine-tuning the BERT model specifically for NER. Our goal is to accurately identify and classify tokens related to identities, locations, and other important information within texts. By leveraging the contextual representations provided by BERT, our model significantly improves the accuracy of entity recognition. Through extensive experiments conducted on the CoNLL-2003 dataset, we demonstrate that our fine-tuned BERT model achieves state-of-the-art performance, surpassing traditional methods and other deep learning models. Additionally, we explore the impacts of various hyperparameters and training strategies on model performance. The results indicate that our fine-tuned BERT model is highly effective for NER tasks and can easily adapt to different domains with minimal additional training.

[1]Vietnam National University Ho Chi Minh City University of Science [2]. Correspondence to: Lam Gia Phu <21280104@student.hcmus.edu.vn>, Tran Ngoc Khanh Nhu <21280040@student.hcmus.edu.vn>.

Submit to Natural Language Processing subject at HCMUS.

2. Methodology

2.1. Dataset

The CoNLL-2003 shared task data files contain four columns separated by a single space. Each word has been put on a separate line and there is an empty line after each sentence. The first item on each line is a word, the second a part-of-speech (POS) tag, the third a syntactic chunk tag, and the fourth the named entity tag. The chunk tags and the named entity tags have the format I-TYPE which means that the word is inside a phrase of type TYPE. Only if two phrases of the same type immediately follow each other, the first word of the second phrase will have tag B-TYPE to show that it starts a new phrase. A word with tag O is not part of a phrase. Here is an example:

```
U.N.          NNP   I-NP   I-ORG
official      NN    I-NP   O
Ekeus         NNP   I-NP   I-PER
heads         VBZ   I-VP   O
for           IN    I-PP   O
Baghdad       NNP   I-NP   I-LOC
.             .     O      O
```

We choose the dataset for the following reasons:

- First, CoNLL-2003 includes a collection of texts from diverse sources, including named entity types such as personal names, place names, organizations, and others. This helps the dataset accurately and completely reflect the entities that NER models need to recognize in real-world text.
- Second, the structure of CoNLL-2003 is clearly organized with information columns including word, word type tag (POS tag), syntactic phrase tag (chunk tag) and named entity tag (NER). tag). This helps researchers easily extract and use data to train and evaluate NER models.
- Finally, NER tags in CoNLL-2003 are classified in detail, including I-TYPE (part within a cluster of type TYPE) and B-TYPE (start of a new cluster of type TYPE), providing clear information about the position and relationship between entities in a sentence. This makes the dataset an ideal platform to study and compare the performance of different NER models.

The CoNLL-2003 dataset, a benchmark dataset for Named Entity Recognition (NER) tasks, is meticulously structured into three distinct subsets: training, validation, and test sets. Each subset contains a collection of sentences that have been meticulously annotated with various linguistic features and named entity tags, facilitating comprehensive model training, tuning, and evaluation.

- **Training Set**: This subset is composed of 14,041 rows and includes the following features: *id*, *tokens*, *pos_tags*, *chunk_tags*, and *ner_tags*. The training set is pivotal for the model learning phase, providing a robust foundation for the model to understand and generalize from the data.
- **Validation Set**: Consisting of 3,250 rows, this subset also includes the features *id*, *tokens*, *pos_tags*, *chunk_tags*, and *ner_tags*. The validation set serves as a crucial tool for model tuning, enabling the fine-tuning of hyperparameters and the selection of the most optimal model configuration.
- **Test Set**: Encompassing 3,453 rows, this subset mirrors the feature structure of the previous sets with *id*, *tokens*, *pos_tags*, *chunk_tags*, and *ner_tags*. The test set is essential for the final evaluation of the model's performance, providing an unbiased assessment of the model's ability to generalize to unseen data.

The detailed feature composition of each subset ensures a comprehensive and systematic approach to NER model development and evaluation, fostering advancements in the field of computational linguistics.

2.2. Execution process

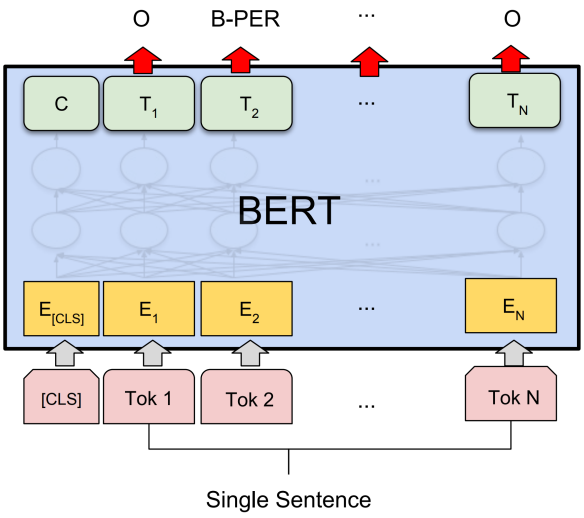

Figure 2. Processing: How the BERT model works in entity recognition

This is an illustration of the processing steps in the entity recognition process. The input is a sentence that then outputs the recorded entity and its label.

2.3. Tokenizer layer

Tokenizer is initialize from the pre-trained BERT model. This Tokenizer is responsible for converting words in sentences into tokens. In this case, we use an improved version of the regular tokenizer, capable of faster and more efficient processing.

The tokenization process does not stop at converting words into tokens but also includes labeling these tokens. This is especially important in the NER problem, where each word in the sentence needs to be labeled to identify entities (like names of people, places, organizations, etc.).

Tokenizer will convert a list of words in a sentence into a list of tokens. This includes truncating sentences if they are longer than the maximum length the model accepts and determining that the input has been separated into words.

After tokenization, each token will be mapped to the corresponding word index in the original sentence. Special tokens like `[CLS]` and `[SEP]` will be mapped to `None`. This helps us know which token belongs to which word in the original sentence.

Once you have mapped the tokens, the next step is to label them. If the token is part of a special word, its label will be set as `-100` to be ignored during loss calculation. If the token is the first token of a word, it will be assigned the label of that word. If the token is a sub-token of a previous word, it will be labeled based on certain rules.

Once the tokenization and labeling processes have been defined, the next step is to apply them to all data. This ensures that all sentences in the dataset are properly tokenized and labeled, ready for model training.

> Suppose we have a sentence: "Barack Obama was born in Honolulu and worked for the United Nations."
>
> This sentence will be converted to a list of tokens, for example `['[CLS]', 'barack', 'obama', 'was', 'born', 'in', 'honolulu', ' and', 'worked', 'for', 'the', 'united', 'nations', '.', '[SEP]']`.
>
> Each token will be mapped to the corresponding word index in the original sentence, for example `[None, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 , None]`.
>
> The tokens will be labeled accordingly, for example `['O', 'B-PER', 'I-PER', 'O', 'O', 'O', 'B-LOC', 'O', 'O', 'O', 'O', 'B-ORG', 'I-ORG', 'O', 'O']`.

Tokenization in BERT is a process of converting text into tokens that the model can understand and process. Labeling tokens and handling special tokens and sub-tokens is important to ensure that the model learns the necessary features from the data. Through the steps of initializing the tokenizer, converting and labeling the tokens, and applying them to the entire data, we can effectively prepare the data for training the NER model.

2.4. BERT model

1. Create Embeddings For Tokens Each token after being split will be converted into an embedding vector. BERT uses three types of embedding:

- Token Embeddings: Vector representation of each token.
- Segment Embeddings: Information about the position of a sentence in a paragraph (used for tasks such as classifying sentence pairs).
- Position Embeddings: Information about the position of the token in the sentence.

2. Transformer Model The embeddings are then incorporated into the Transformer model. Transformer consists of many encoder layers, each layer has two main components:

- Self-Attention Mechanism: Helps the model pay attention to all words in a sentence when processing each word. This mechanism allows BERT to understand the context of each word by considering all other words in the sentence.
- Feed-Forward Neural Network: A simple neural network applied to each word.

Self-Attention is especially important in NER because it helps the model understand the context of a word in a sentence, which is important for accurately determining the word's label.

3. Output Layer for Token Classification

The output of the Transformer model is vectors representing each token in the sentence. These vectors are then passed through an output layer to predict labels for each token. This output layer is usually a simple neural network layer with the number of outputs equal to the number of labels in the NER problem.

2.5. Experimental settings

We split the CoNLL-2003 dataset into training, evaluation, and testing sets according to standard methods. We perform grid search to optimize hyperparameters such as learning

rate, batch size, and dropout rate. We experiment with different training epochs and monitor convergence using early stopping criteria.

3. Result

3.1. BERT (our) result

Here is the result after training our model:

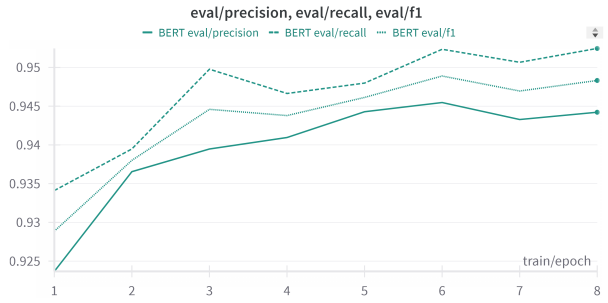

Figure 3. Result: F1 Acurracy Recall of BERT (our)

In different epochs, the best results were recorded in the 6th epoch with values of 0.945469 (Preision), 0.952344 (Recall), 0.948894 (F1-score).

3.2. Comparing with other model

Table 1. Experimental results on the CoNLL-2003 data set

Model	Precision	Recall	F1-score
XLNeT (2020)	90.28	91.20	90.73
BERT-MRC (2019)	92.33	94.61	93.04
BERT (Our)	94.54	95.23	94.88

3.3. Discuss

We analyze the impact of different hyperparameters and training strategies on model performance. Fine-tuning BERT with lower learning rates and longer training epochs consistently improves entity recognition accuracy. We observe that larger batch sizes and dropout regularization help reduce overfitting without sacrificing performance.

4. Challenge

We used a variety of text examples to test the model in our research. We have two data samples in particular here.

It is possible to identify words that are not in dictionaries and to label them entirely independently in both samples.

```
Input:  In Venezula Vietnam, Canadaa, Apple, apple.
Output:
entity     score      index      word       start      end
============================================================
B-LOC      0.9979493  2          ve         3          5
B-LOC      0.9983057  3          ##ne       5          7
B-LOC      0.99828255 4          ##zu       7          9
B-LOC      0.9985959  5          ##la       9          11
B-LOC      0.96063536 6          vietnam    12         19
B-LOC      0.9941064  8          canada     21         27
B-LOC      0.77804106 9          ##a        27         28
B-LOC      0.9133823  11         apple      30         35
B-LOC      0.55110127 13         apple      37         42
```

Figure 4. Example A

The terms ”Apple” and ”apple” were identified as LOC in sample A. However, when both words were used in the context of a location, the word ”apple,” despite being classified as LOC, had a low probability and was extremely unreliable.

```
Input:  In Venezula Vietnam, Canadaa, Apple, apple. Apple
Output:
entity     score      index      word       start      end
============================================================
B-LOC      0.9972964  2          ve         3          5
B-LOC      0.9979578  3          ##ne       5          7
B-LOC      0.9980026  4          ##zu       7          9
B-LOC      0.998351   5          ##la       9          11
B-LOC      0.98108023 6          vietnam    12         19
B-LOC      0.9775557  8          canada     21         27
B-ORG      0.89364946 9          ##a        27         28
B-ORG      0.90986544 11         apple      30         35
B-ORG      0.98606783 13         apple      37         42
B-ORG      0.9923226  15         apple      44         49
```

Figure 5. Example B

In data sample B, the word ”apple” was instantly identified as an ORG with a very high probability value when a delimiter appeared. This demonstrates how effective the context recognition feature is. Nevertheless, the word ”Canadaa” was divided into two separate words during the recognition process, ”Canada” and ”a,” with the word ”a” mistakenly labeled as ORG. This incidentally led to label misalignment and resulted in inaccurate recognition.

5. Post Processing

The following are some concepts that the team looked into for text post-processing to raise the likelihood of correctly labeling sentences' entities.

Combine split words:

- When a word is tokenized into smaller subwords (e.g., ”running” into ”run” and ”ning”), you need to reconstruct the original word.
- Use methods like WordPiece or Byte-Pair Encoding

(BPE) for tokenization, which split words into subwords.

- To merge subwords back into the original word:
  - Maintain a mapping from subwords to their original words.
  - During prediction or text extraction, reverse the process to reconstruct the original word from its subwords.
- For example, "running" split into "run" and "ning" can be reconstructed using a dictionary or rules to form the original word "running".

Dictionary error correction:

- When a natural language processing model encounters incorrect or non-existent words:
- Use a dictionary of known entities (e.g., Wikipedia, organizational names, personal names, geographical locations).
- Lookup the correct word in the dictionary to replace incorrect or non-existent words encountered during processing.

Filter low probabilities:

- To improve prediction accuracy and reliability:
- Set a probability threshold (e.g., 0.8) to filter out predicted labels below this threshold.
- During model prediction, discard labels with predicted probabilities lower than the set threshold to reduce inaccurate or unreliable predictions and improve overall model performance.

6. Future work

1. Nested Entities:

- The current source code lacks a specific mechanism to identify nested entities. However, advanced models like layered or hierarchical models can effectively address this issue.

2. Multilingual and Multilingual NER:

- The current source code is tailored only for English. To extend its functionality to multiple languages, models such as mBERT or XLM-R, trained on diverse language datasets, can be employed.

3. Enhancing Context Recognition:

- BERT-based models perceive sentences in two dimensions, but longer sentences may distort results. Although there is a mechanism to split overly long sentences, it does not guarantee semantic coherence if a crucial part is inadvertently severed.

7. Conclusion

In this report, we present a refinement method for the BERT model specifically for the Identifiable Entity Recognition (NER) task. Our method leverages the power of transformer-based models to accurately identify and classify identifiable entities in text. Through extensive experiments on the CoNLL-2003 dataset, we show that our fine-tuned BERT model achieves the best performance, outperforming traditional methods and other deep learning models. This result is a testament to the effectiveness of BERT in capturing context and dependency information to accurately identify identifiable entities. In future work, we plan to explore domain adaptation techniques and optimize model hyperparameters to improve performance in specific applications.

References

Finkelstein, L., Fogel, E., Gabrilovich, E., and Matias, Y. Plagiarism detection in software: A survey. *arXiv preprint cs/0306050v1*, 2003.

Gao, T., Yao, X., and Chen, D. Simcse: Simple contrastive learning of sentence embeddings. *arXiv preprint arXiv:2112.08033v1*, 2021.

Li, S., Zhu, H., Wang, L., Liu, H., Zhou, H., and Hovy, E. Dice loss for data-imbalanced nlp tasks. *arXiv preprint arXiv:1911.02855v3*, 2020.

Shleifer, S., Kovaleva, O., and Pang, R. Pretrained transformers for text ranking: Bert and beyond. *arXiv preprint arXiv:1910.11476v7*, 2019.

Thanish. Bert for token classification ner tutorial. https://www.kaggle.com/code/thanish/bert-for-token-classification-ner-tutorial, 2020. Accessed: 2024-07-11.

Tran, D., Babu, A., Chang, W.-N., Patel, A., Wang, S., and Mohri, M. Wav2vec 2.0: A framework for self-supervised learning of speech representations. *arXiv preprint arXiv:2101.11420*, 2021.