Title: Utilizing Transformers for Word Sense Disambiguation

Hesam Farjad

Abstract

This report focuses on the application of Transformers, specifically the Bert model, for Word Sense Disambiguation (WSD). The report discusses the process of installing the Transformers library and importing the necessary modules. It then explores the steps involved in reading and manipulating the data, including the utilization of nested dictionaries and constructing a mapper. The report also delves into tokenization and embedding of words using the Bert model. Furthermore, it explains the functions for obtaining word vectors and performing similarity calculations. The process of training a model for WSD using the encoded sentences and labels is described, along with the evaluation of model accuracy. The report concludes with the results obtained and discusses the potential applications of this approach.

1 Introduction

In this section, the report provides an overview of the purpose and scope of the study. It highlights the significance of Word Sense Disambiguation and introduces the Transformers library and the Bert model as key components of the proposed approach. The section concludes by outlining the structure of the report.

2 Installing / Importing Transformers:

This section focuses on the initial steps required to utilize the Transformers library. It discusses the process of installing the library and importing the necessary modules. The section emphasizes the importance of incorporating the Bert model for WSD.

3 Data Processing:

This section explains the steps involved in reading and manipulating the data. It introduces two functions, namely "read_map_data" and "read_files," which facilitate data extraction and transformation. The section highlights the use of nested dictionaries and the construction of a mapper to handle the dataset effectively.

4 Tokenization and Embedding:

This section delves into the tokenization process and the utilization of the Bert model for embedding words. It describes the conversion of words into tokens and the extraction of word vectors from the encoded sentences. The section emphasizes the importance of obtaining the desired word index and explores the output of the Bert model's hidden states.

5 Similarity Calculation:

This section explains the process of calculating similarity scores for word senses. It discusses the utilization of a similarity function and the comparison of output meanings. The section also introduces the SoftMax function for score normalization and the assignment of significance values to candidate words.

6 Model Training and Evaluation:

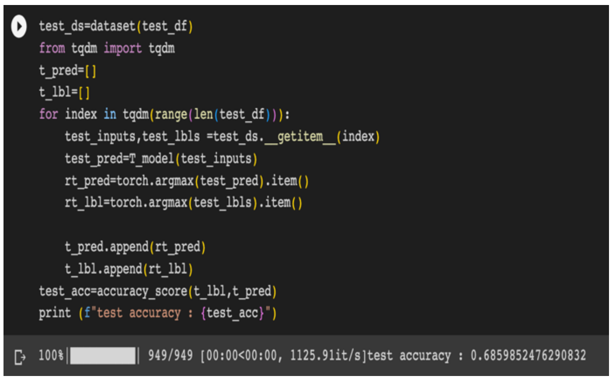
This section outlines the process of training a model for Word Sense Disambiguation using the encoded sentences and labels. It describes the creation of a dataset class and the implementation of a model based on the Torch library. The section discusses the training process, including the choice of optimizer, loss function, and learning rate. It also presents the evaluation of model accuracy using the test dataset.

7 Results and Discussion:

This section presents the results obtained from the model training and evaluation. It discusses the accuracy achieved and provides insights into the performance of the proposed approach. The section also highlights the potential applications of the developed model in real-world scenarios.

8 Conclusion:

The conclusion summarizes the key findings of the study and discusses the significance of using Transformers, specifically the Bert model, for Word Sense Disambiguation. It also mentions the limitations of the approach and suggests areas for future research and improvement.
Overall, this report provides a comprehensive overview of the utilization of Transformers for Word Sense Disambiguation. It covers the necessary steps involved in data processing, tokenization, embedding, similarity calculation, model training, and evaluation. The results obtained demonstrate the potential of this approach for improving the accuracy of word sense disambiguation tasks.



```
test_ds=dataset(test_df)
from tqdm import tqdm
t_pred=[]
t_lbl=[]
for index in tqdm(range(len(test_df))):
    test_inputs,test_lbls =test_ds.__getitem__(index)
    test_pred=T_model(test_inputs)
    rt_pred=torch.argmax(test_pred).item()
    rt_lbl=torch.argmax(test_lbls).item()

    t_pred.append(rt_pred)
    t_lbl.append(rt_lbl)
test_acc=accuracy_score(t_lbl,t_pred)
print (f"test accuracy : {test_acc}")
```

```
100%|██████████| 949/949 [00:00<00:00, 1125.91it/s]test accuracy : 0.6859852476290832
```