The first assignment of NLP

Marco Acerbis, Lesi Li, Xiaotian Fan
Master's Degree in Artificial Intelligence, University of Bologna
{ marco.acerbis, xiaotian.fan, lesi.li }@studio.unibo.it

Abstract

Parts of Speech (POS) Labeling as Sequence Tokens Using Recurrent Neural Architectures

1 Introduction

Code for POS tagging using GRU. Unlike LSTM, GRU requires only one gate to control the flow of information, thus reducing the number of parameters. As such, it is easier to train than LSTMs and performs reasonably well on many NLP tasks.

2 Background

In this Assignment, we were asked to perform POS tagging using neural architectures.

3 System description

The main steps include Download the corpora and split it in training and test sets, structuring a dataframe. Embed the words using GloVe embeddings Create a baseline model, using a simple neural architecture Experiment doing small modifications to the baseline model, choose hyperparameters using the validation set Evaluate two best model Analyze the errors of the model

4 Data

"dependency_treebank.zip" contains the Dependency Treebank dataset. The Dependency Treebank is a resource for natural language processing research and development that contains sentences and their dependency syntactic structures.

By using the "dependency_treebank.zip" file, we can obtain marked sentences and their dependency syntactic structure data for training, research and evaluation of natural language processing tasks.

5 Experimental setup and results

1.Download the dependency treebank corpus and Splits: documents 1-100 are the train set, 101-150 validation set, 151-199 test set.

2.A data frame (DataFrame) is created, and the divided training, validation and test data are stored in the data frame in the form of "Word" and "POS" columns.

3.Count and plot the distribution of the number of words in each file in a dependency syntax tree bank.

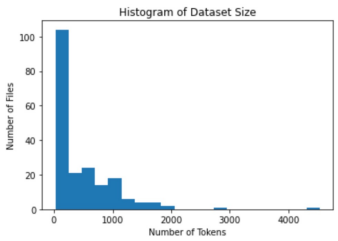

Figure 1: Number of Tokens

4.Draw a histogram of the sentence length distribution

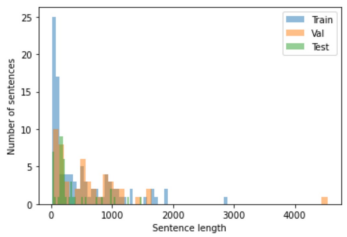

Figure 2: sentence length

5.Embedding Words Using GloVe Embeddings: Download GloVe Embeddings, Read GloVe Embedding Files

6.Prepare the vocabulary, add embeddings to the vocabulary, and finally get the vocabulary V4=V1+OOV1+OOV2+OOV3

7.Prepare the model: A baseline model is defined that uses bidirectional GRU layers and fully connected layers to process sequence data, using pretrained word embeddings as input features.

8.Based on the extension and improvement of the baseline model, by adding layers and parame-

```
GRUModel(
  (embedding): Embedding(12003, 50, padding_idx=0)
  (gru): GRU(50, 100, batch_first=True, bidirectional=True)
  (fc1): Linear(in_features=200, out_features=100, bias=True)
  (relu): ReLU()
  (fc2): Linear(in_features=100, out_features=10, bias=True)
)
```

Figure 3: GRU-model

ters, different model configurations can be tried to achieve better performance.

```
ModifiedModel(
  (embedding): Embedding(12003, 50)
  (gru): GRU(50, 100, num_layers=2, batch_first=True, bidirectional=True)
  (fc1): Linear(in_features=200, out_features=100, bias=True)
  (relu): ReLU()
  (fc2): Linear(in_features=100, out_features=10, bias=True)
)
```

Figure 4: MODIFIED-model

9.Define hyperparameters

10.Create three best models model1:GRU (two-way); model2: GRU with 2 layers (bidirectional); model3: LSTM with 2 layers (bidirectional)

11.Three different models are trained, namely Model1 ,Model2 and Model3, using a similar training loop structure. Each model has its own loss function, optimizer and training/validation loop.

12.Prepare test data

13.Three models that have been trained are loaded, and their accuracy and F1-Macro scores are calculated on the test data to evaluate their performance on real data.

14.Choose the Best Model

6 Discussion

1. When creating a DataFrame,store the processed training, validation, and test datasets as columns containing words (Word) and part-of-speech tags (POS). The extraction of words and part-of-speech tags occurs simultaneously with the creation of the dataframe, making the code clearer and more concise.

2. Use the requests module to download files, using a simpler download method. In comparison, before using urllib.request and zipfile modules to download and decompress files, using a more traditional download and decompression method may appear more cumbersome

3. Gradually expand the vocabulary and compute the embeddings of the terms in the split vocabulary in order to prepare more vocabulary for the model. I ignored these steps before, which caused the model to have great difficulties when dealing with new data, because it could not generate appropriate word vector representations to capture the semantic information of unknown vocabulary. Without these steps, the model will not be able to handle unknown words in the training and test data, resulting in a loss of semantic understanding of these words. These steps are to deal with the unknown vocabulary (Out-of-Vocabulary, OOV) that appears in the training and test data, and to generate suitable word vector representations for these unknown vocabulary. Therefore, these steps play a key role in dealing with unknown vocabulary.

4.Evaluate the best models:

```
Model1 - Test Accuracy: 0.8144, Test F1-Macro: 0.7823
Model2 - Test Accuracy: 0.8313, Test F1-Macro: 0.7625
Model3 - Test Accuracy: 0.8313, Test F1-Macro: 0.7575
```

Figure 5: Accuracy and F1-Macro scores

According to the final test results, Model1 has a slightly higher test F1-Macro score than other models, while Model2 has a slightly higher test accuracy. Given that the F1-Macro score can be more instructive in some cases, I choose Model1 as the best model

7 Conclusion

The purpose of this project is to perform a dependency parsing task based on the "dependency_treebank.zip" corpus and build a baseline model. The final conclusion is to choose the model with the best performance as the final model. Overall, this project aims to research and implement dependency parsing tasks, and through data processing, model building and evaluation, the best performing model can be obtained for final application. The final conclusion will be the best model selected and the performance evaluation results of this model on the test set.

Based on the above considerations, I would tend to choose the single-layer GRU model, because it performs best on the test F1-Macro score, and also has a good performance on test accuracy. Although the test accuracy of the two-layer GRU and two-layer LSTM is slightly higher than that of the single-layer GRU, they are slightly inferior in the F1-Macro score, and may also be more prone to overfitting due to the higher model complexity.