Spring 2019 COSI137: Programming Assignment #2
Zhiheng Wang

**Dependencies:**
Python 3.6.5
Flair (https://github.com/zalandoresearch/flair): used for NER tags
Segtok (https://github.com/fnl/segtok): used for tokenizing sentence
Mxnet Gluonnlp deep learning framework (https://github.com/dmlc/gluon-nlp).
Scikit Learn (https://scikit-learn.org/stable/install.html)

*Install via pip:*

```
pip3 install flair
pip3 install segtok
pip3 install mxnet
pip3 install gluonnlp
pip3 install -U scikit-learn
```

**Run Instructions:**
Step 1:
Run text_cleaner.py to clean the original data sets (both training data and test data). This will remove whitespace between the leading double quote marks and first token and then write into a new file (cleanTrain.tsv/cleanTest.tsv).

Step 2:
Run data_preprocessing.py to use NER tags to replace originally tokens if any NER tags found in the sentences. This will write the NER tagged sentences with their new indices for $e_1$ and $e_2$ into a new file (EntityTypeTrain.tsv/EntityTypeTest.tsv).

Step 3:
Run text_cleanertwo.py to clean the output file again. After this, the final training data and test data will be usable (train: cleaned_entity_type_train.tsv, test: clean_entity_type_test.tsv). (known issues: there seems to be some missing quote marks in the original training data and some \x01 and \ in the original test data, you may need to clean them manually since there are only few).

Step 4:
Run train.py to start training the model and predict the labels for the test data. The predicted label will be written into a txt file. The code for cross validation to select model and tune the parameter is also available.

*Note: Since I have uploaded the modified training data and test data, you can jump to step 4 to train the model.*

**Data Description:**
cleanTrain.tsv/cleanTest.tsv makes no change to original data, it only cleans the whitespace. You may use this to train the model.

cleaned_entity_type_train.tsv/clean_entity_type_test.tsv replaces the tokens by their entity type if detected. There are three types of entity: PER-span, ORG-span, and LOC-span.
For example, Stephanie Eitel is replaced by PER-span, East London is replaced by LOC-span, and InterSense, Inc. is replaced by ORG-span.

**Experiment Report**

**Objective:** The focus of this experiment is to develop a relation classifier to identify 19 different classes of two identified entities in the sentence.
**Dataset:** The dataset for this assignment is the SemEval 2010 Task 8 Dataset.
**Experiment Table**

| Model/Features | Architectures | Validation Accuracy |
|---|---|---|
| Baseline | Transformer Encoder + glove.6B.300d + adam | 0.492 |
| Model 1 | BiLSTM glove.6B.300d + adam | 0.487 |
| Model 2 | Transformer Encoder + glove.6B.100d + adam | 0.554 |
| Model 3 | Transformer Encoder + Binary Relation Encoder + glove.6B.100d + adam | 0.641 |
| Model 4 | *Chunk + Transformer Encoder + Binary Relation Encoder + glove.6B.100d + adam | 0.640 |
| Model 5 | *NER + Transformer Encoder + Binary Relation Encoder + glove.6B.100d + adam | 0.653 |
| Model 6 | *NER + Transformer Encoder + Binary Relation Encoder + FastText crawl-300d-2M + fine-tuned dropout rate + adam | 0.705 |
| Model 7 | *NER + Transformer Encoder + Binary Relation Encoder + glove.840B.300d + fine-tuned dropout rate + adam | 0.684 |
| Model 8 | *NER + Transformer Encoder + Binary Relation Encoder + FastText crawl-300d-2M + fine-tuned dropout rate + nadam | 0.718 |

*Note: *NER refers to apply flair model of 4-class Named Entity Recognition pre-trained on Conll-03 dataset. See details in flair's documentation.*
**Chunk refers to apply flair model of Syntactic Chunking pre-trained on Conll-2000 dataset. See details in flair's documentation.*

**Word Embedding**

In the above experiment table, I find that there is a 6% accuracy increase after switching pre-trained embedding from glove.6B.300d to glove.6B.100d in the naïve transformer encoder classifier, which is somehow anti-intuitive. Usually, models are beneficial from larger dimension of words embedding and this can be verified by the accuracy improvement from model 5 to model 6 and model 7. The reason why this happens might be that there are some rare words which do not appear in the pre-trained embedding. Then since I randomly initialize the unknown words, using 300d pre-trained embedding would make the unknown words easier to over fit the training dataset.

**Model Architectures**

Transformer encode runs faster than BiLSTM and CNN on my machine and based on the accuracy result, we can see that their ability to capture the features and patterns are equally good. Therefore, I decide to use transformer encoder on my input embedding layer. Binary Encoder which only uses two entities ($e_1$, $e_2$) as query would focus on the two entity and ideally this will remove the noises.

**Input features**

Intuitively, person's name, location's name, and organization's name have little or no impact on these 19 relationship between two entities. To reduce the noises, I decide to replace the tokens by their detected entity type. The experiment shows that there is an increase in validation accuracy when tokens are replaced by their detected entity type.

Besides, some words might be ambiguous and I therefore add chunking tags between every token. However, this does not change the accuracy result at all. Since it doubles the sentence length and make the computation time longer, I discard this approach.

**Fine-tune Model Parameter**

Some researchers like to use relu activation function in their dense layer, and therefore I also try it in my model. However, either replace current dense layer without activation function by dense layer with relu or add another dense layer with relu before the current dense layer, the validation accuracy decreases although relu is very fast. Also, I notice that dropout rate impact the validation accuracy significantly. Currently, I choose to have a dropout rate of 0.1 on the transformer encoder, 0 on binary relation encoder, and 0.5 on the output layer to achieve the best result among all experiments. A dropout rate of 0.1 on transformer encoder would decrease the noises in the model. I apply 0 dropout rate on the binary encoder is because the output of binary encoder is very important and I don't want to lose any information on the output. Finally, a 0.5 dropout rate in the output layer to avoid overfitting. I change the optimizer from adam to nadam because nadam seems to have better performance on finding the optimal result, but there is a tradeoff of speed. Researchers usually use sgd to get the state of the art result, but sgd is extremely slow on my machine.

**Result**

| | Accuracy | Macro Precision | Macro Recall | Macro F1 Score | Training Loss | Test Loss |
|---|---|---|---|---|---|---|
| *NER + Transformer Encoder + Binary Relation Encoder + FastText crawl-300d-2M + fine-tuned dropout rate + nadam | 0.71 | 0.67 | 0.66 | 0.65 | 398.52 | 156.18 |

| Class | Precision | Recall | F1 Score |
|---|---|---|---|
| Component-Whole | 0.84 | 0.64 | 0.72 |
| Component-Whole-Inv | 0.74 | 0.62 | 0.68 |
| Instrument-Agency | 0.5 | 0.21 | 0.30 |
| Instrument-Agency-Inv | 0.73 | 0.78 | 0.75 |
| Member-Collection | 0.36 | 0.29 | 0.32 |
| Member-Collection-Inv | 0.78 | 0.93 | 0.85 |
| Cause-Effect | 0.80 | 0.71 | 0.76 |
| Cause-Effect-Inv | 0.74 | 0.92 | 0.82 |
| Entity-Destination | 0.81 | 0.87 | 0.84 |
| Entity-Destination-Inv | 0 | 0 | 0 |
| Content-Container | 0.71 | 0.86 | 0.78 |
| Content-Container-Inv | 0.71 | 0.81 | 0.76 |
| Message-Topic | 0.72 | 0.86 | 0.78 |
| Message-Topic-Inv | 0.84 | 0.57 | 0.68 |
| Product-Producer | 0.70 | 0.79 | 0.74 |
| Product-Producer-Inv | 0.70 | 0.67 | 0.69 |
| Entity-Origin | 0.73 | 0.66 | 0.69 |
| Entity-Origin-Inv | 0.69 | 0.83 | 0.76 |
| Other | 0.53 | 0.45 | 0.49 |

The table above is a single iteration of a 5-fold cross-validation. The model gets its optimal performance on test data with a training loss of range from 280 to 400, and beyond that the model starts overfitting. Since I use the random batch approach, there might be an uneven class distribution and some class may have few or 0 training dataset and this may cause the class "Entity-Destination-Inv" to have 0 precision, recall, and F1 score. Also notice that the model struggles on learning Member-Collection while performing well on Member-Collection-Inv.

**Future Work**

For future work, one possible approach would be adding an attention layer which pays attention on only tokens between two entity ($e_1$, $e_2$), since these tokens are intuitively more significant

than others. Another approach is that add a BiLSTM layer with self-attention on the input embedding with a dependency parser added in the sentence. Then concatenate the results to the original words embedding and send it to the transformer encoder and then binary relation encoder. Also, there is a problem with the positions of two entities.

**Known Problems**

Some entities have multiple tokens but the position of those entities is the position of their first tokens and this would raise a problem of using binary encoder which only pays attention to those two positions.