Project Report

CS 6320: Natural Language Processing (Fall 2020)

Submitted by:

Team: Twisted Minds

Members:

- Shankaranarayanan Kallidaikuruchi Ramakrishnan (SXK190109)
- Vigneshwaran Paramasivan(VXP180052)

Problem Description

For this project, we have designed and implemented models for extracting relations between two named entities in a sentence.

Given two nominals embedded in a sentence,the task requires identifying which of the following nine semantic relations holds between the nominals:

- Cause-Effect,
- Instrument-Agency,
- Product-Producer,
- Content-Container,
- Entity-Origin,
- Entity-Destination,
- Component-Whole,
- Member-Collection,
- Message-Topic, or
- Other

For instance, the following sentence provides an example of the Entity-Origin relation.

Input sentence = "The male took over the entire care of the <e1>young</e1> that had left the <e2>nest</e2>"

Relation between entities = 'Entity-Origin(e1,e2)'

Proposed Solution

We plan to use deep learning techniques combined with powerful pre-trained Natural Language Processing models to obtain the semantic & syntactic information from sentences.

We created a deep NLP pipeline to extract the features such as Parts-of-Speech tags, dependency parse tree, NER tags, tokens, lemmas, synonyms, hypernyms, hyponyms, meronyms and holonyms from the statements.

We implemented a convolutional neural network based approach using word embeddings to determine the relation and its direction for the given pair of arguments.

Implementation Details

Programming tools:

- Python (*version*: 3.7.4)
- jupyter==1.0.0
- nltk==3.4.5
- spacy==2.3.2
- scikit-learn==0.21.3
- tensorflow==2.1.0
- matplotlib==3.1.1
- tqdm==4.51.0
- Keras==2.3.1

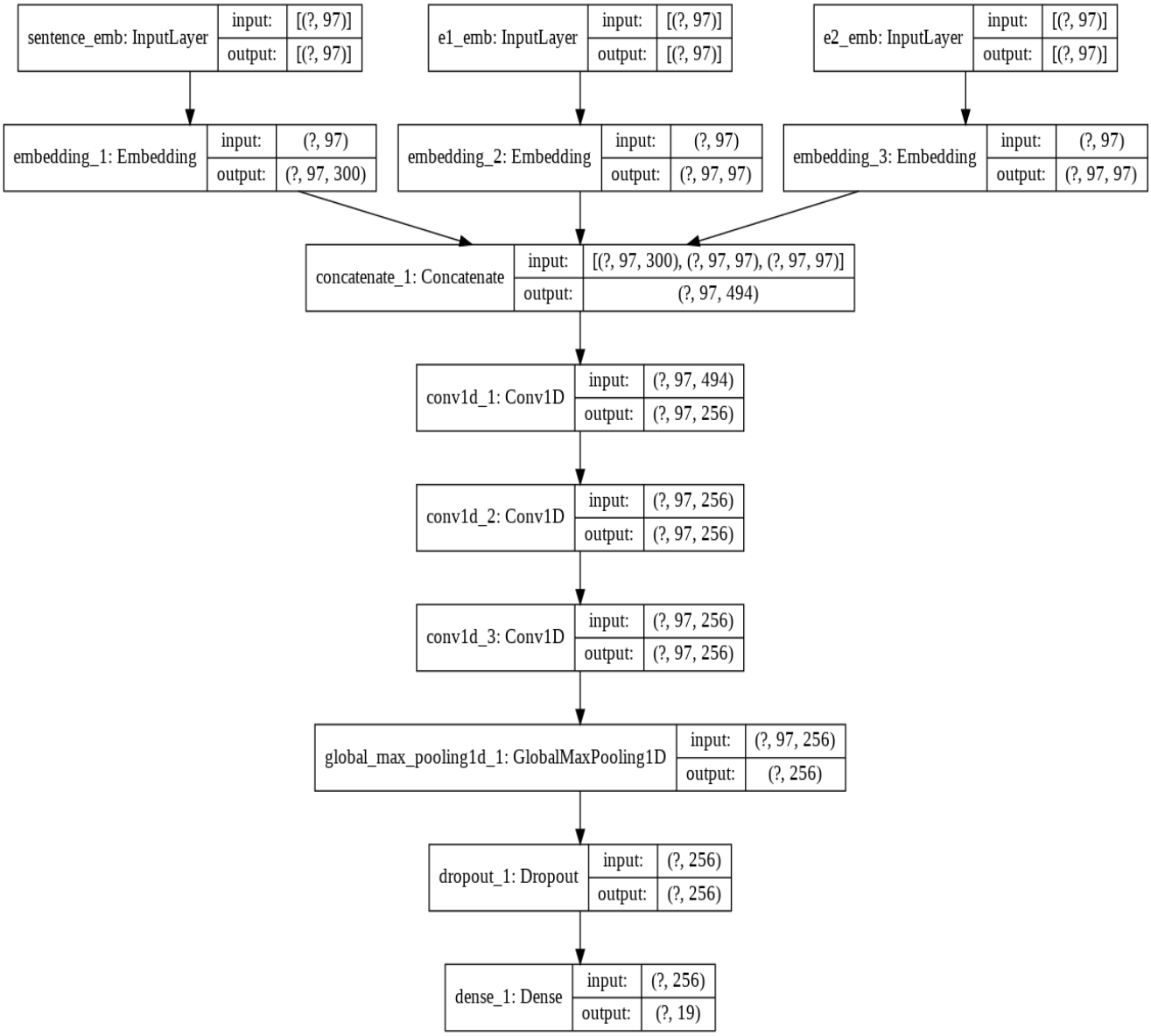
Architecture:

We explore the feasibility of performing relation classification without complex NLP feature extraction methods. In this project, we tackle the relation classification task using a convolutional neural network that performs classification by using word embeddings and position embeddings, similar to the implementation done by Zeng et al. (2014).

The raw sentences are extracted from the file using the CorpusReader class which takes care of cleaning the sentences and extracting the entities. The processed sentences are stored in ProcessedSentences class. Features like POS tags, Dependency parse tree and Wordnet hypernyms, holonyms and meronyms are extracted for each sentence in the corpus. LabelEncoder is used to encode the given classes to integer format for the model.

We are using GloVe word vectors pre-trained on Wikipedia data to create word embeddings for each token in the given dataset. Position embeddings denote the position of each nominal entity in the given sentence and the distance between the current entity and other tokens in the sentence. These are provided as inputs to the Convolutional layer. SoftMax activation is used to predict the model output probabilities. ArgMax of all probabilities is the predicted label of the model.

The model architecture is as follows:



Results and error analysis:

- On the test set, we evaluated our system and computed the
    - accuracy
    - macro precision
    - recall
    - F-scores

for the predictions made by our model under two settings

1. Assuming that the relation is classified correctly (but not the direction)

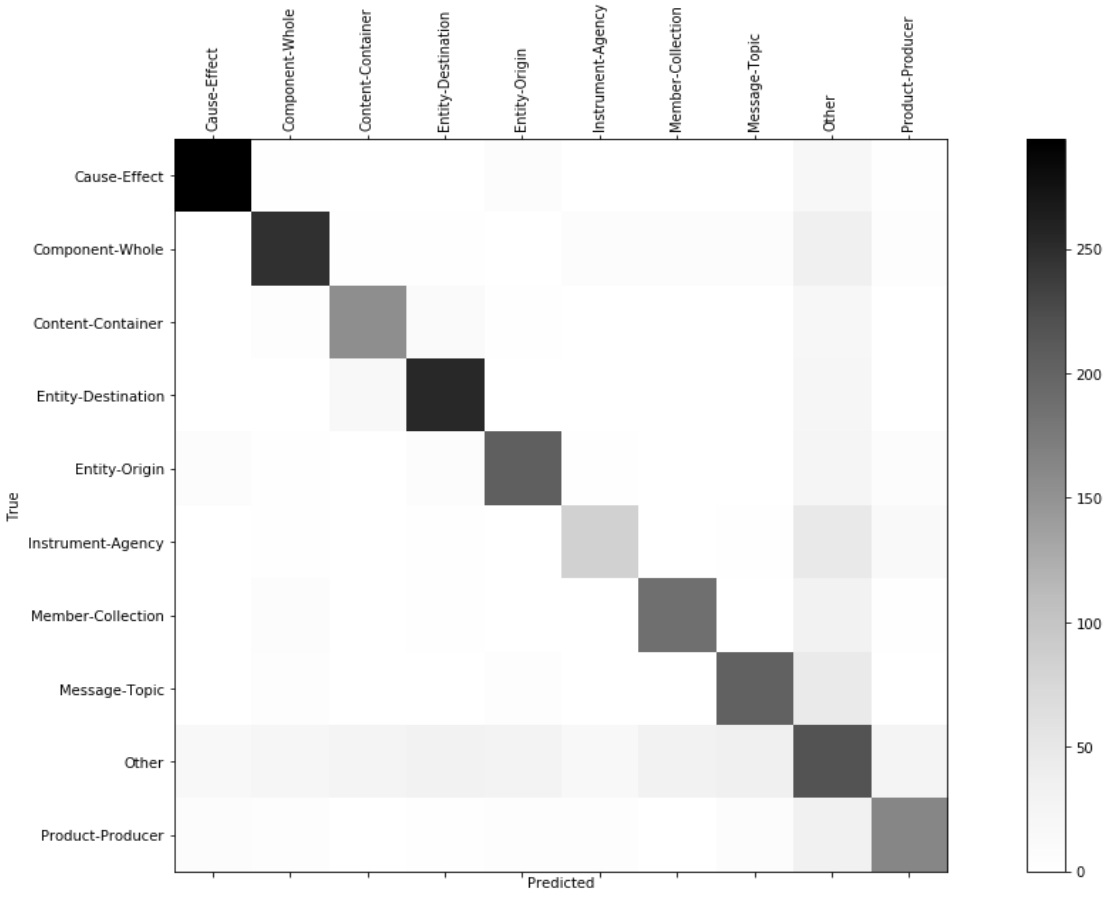
   Accuracy: 0.7401545822598454

   Macro precision 0.7617677770825443

   Recall 0.7463559697725137

   F1 0.752209532364631

   Confusion matrix:



2. Assuming both the relation and direction are classified correctly.

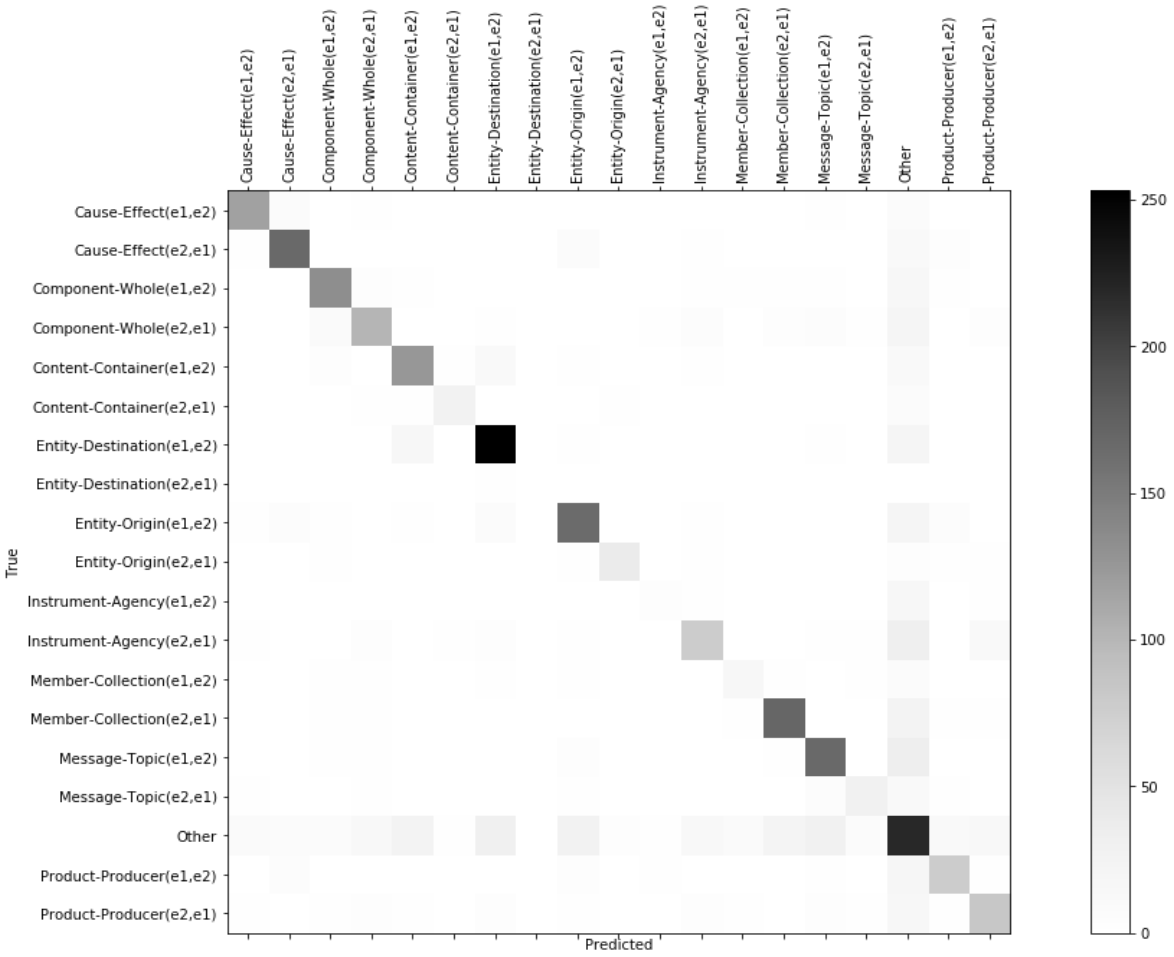
   Accuracy: 0.7261685682738315

   Macro precision 0.7503082642960909

Recall 0.6985477912725911

F1 0.7165436358532591

Confusion matrix:



Problems Encountered:

- Had difficulty in understanding each paper regarding this problem and finally arrived at the paper by Zeng et al. which uses the deep learning approach to solve this given below. (https://www.aclweb.org/anthology/C14-1220.pdf)
- Clear description of how to extract the features from the sentence was not given and later we found by reading many other similar papers by using spacy libraries
- Even though we replicated the paper ,we still had many practical difficulties in achieving the accuracy and F1 Score as mentioned and later we were able to achieve in par by having different hyper parameter configurations.

Pending Issues:

- *Other* class is very noisy since it groups many different infrequent relation types. We can see that by omitting the embedding of the class *Other* both precision and recall for the other classes may improve.
- Required an efficient way to embed the word vectors in future for better classification.
- Required to annotate data and use Deep Learning techniques to extract more accurate information.

Potential Improvements:

- By hyper parameter tuning the model can achieve a higher F1 Score on the cost of the training time.
- Deep learning models perform better with more training examples. Hence a larger dataset can be provided.
- Instead of using tanh activation function, we can use other activation functions to achieve a higher accuracy.