NNIA Project winter semester 2020/21 - BERT and NER

Sandra Gajic
s8sagaji@stud.uni-saarland.de

Houda Kaoukab
s8hokaou@stud.uni-saarland.de

1 Introduction

Named Entity Recognition (NER) is a NLP task where we detect most commonly nouns in a sentence or text which meanings represent certain entities that fall under different classes such as names, organizations, locations, etc. The motivation behind selecting this method was that it is an important and popular task in the field of information extraction and felt well suited for the implementation of a neural network model for the first time, working with this was helpful in getting familiar with the basic architecture of a neural network and working with different modules common and widely used in neural network applications. The objective was to create a model that can correctly identify an entity from an unseen sentence.

2 Methods

2.1 Contextualized Word Embeddings

Since computers are made for working on numerical data, it is common to embed every word or token into some numerical vector representation. The simplest method would be *one-hot encoding* every word in the vocabulary, which would result in a vector size equal to the size of available words and could not encapsulate any relationships between words. The next step would be to train those one-hot encoded on their neighboring words (embed the word on its context) with a single hidden layer, whose weights are the word embeddings for each word. Similar vectors now induce that there is also a semantic similarity between the words those vectors represent, we can distinguish between similar looking word with different senses (*bark* of a tree vs. a dog's *bark*) and the vectors size is not strictly bound to the size of the vocabulary anymore.

2.2 ***B**idirectional **E**ncoder **R**epresentations from **T**ransformers*

BERT is a large transformer masked language model, where one of the key strategies is that we remove some words in a sentence and then let the model fill in the blanks. BERT can be used by either using the contextualized word embeddings as inputs for another model or fine-tuning the BERT model itself. The BERT Base model uses instead of a single encoder as in the transformers architecture, a stack of 12 encoders (BERT Large uses 24 encoders).

2.3 Implementation

We use the BERT tokenizer and the pretrained, uncased BERT base model in our implementation of Entity Name Recognition. Both of them are specified in the *config.py* file, which contains the paths and hyper parameters of our model. The data we used, which was made available to us and is part of the OntoNotes dataset, was concatenated into one .conll file and then preprocessed by the code from the first part of the project ('datapreprocess.py') to create a .tsv file with three columns for position, word and POS-Tag. The data path can be set in the config.py file.
The architecture of my implementation consists so far of five python files (six when making my datascript loader work). I stored hyper-parameters and paths in one file to easily tune them, as well as defining my BERT model and tokenizer for easy reference within my code and better readability.
Another file processes the dataset and returns inputs that can be used by the BERT model. Two other files, namely the model itself, which returns the loss the POS-Tags, and a script for running the training and evaluation are implemented. They are combined in the final training file, which is also the file to be run in order to train the model. Please see

my README.md file for more information and difficulties I have: README.md