|  |
| --- |
| **Improving on** **DrQA biLSTM Model for NLP on SQuAD by Preprocessing Text with NER** |
|  |
| **Zachary Hardesty**  University of Texas, Department of Computer Science  zachhardesty@utexas.edu |
|  |
|  |
|  |
|  |

Abstract

The DrQA by Chen et al., 2017 scores, at best, 49.39/61.72 on SQuAD Dev EM/F1. Some of the best models score in the high 90s, presenting clear room for improvement. The DrQA model runs a recursive neural net over the entire reference passage. I present the opportunity for improvement in both training time and accuracy by preprocessing passages and questions with named entity recognition (NER). Unfortunately, despite the theoretical small accuracy improvements, EM: & F1: , the time to preprocess each sample significantly hampers it’s effectiveness. Caching this preprocessing can be done to mitigate it.

Introduction

In order to train a QA model effectively, one needs a large and complex pipeline. Named entity recognition can be incorporated into this pipeline to better focus a passage before passing it to a RNN. Using the Python library, spaCy, it’s easy to load a pretrained model to find and label these named entities.

Named entities include various people, places, and things that are proper nouns. The full list of entities can be found in spaCy’s documentation at <https://spacy.io/api/annotation#named-entities>.

Methodology

Everything is handled in the data.py file when samples are generated. I first check for a bin, cached representation of samples. These files are called Pickle files and have been a part of Python for a long time. These allow you to store arbitrary data structures by simply making one function call.

If a Pickle file is not found, I first modify the default pipeline in spaCy by removing the `tagger` and the `parser` and add in the `sentencizer`. This leaves just the `ner` module and serves as a method to save time when processing the data.

The first step is to send in all of the paragraphs to spaCy and let it generate its internal data structure: the document. Documents store several useful pieces of information about each of the paragraphs, including a list of found entities and a list with the paragraph split into sentences.

For each paragraph, I want to generate a sample for each of the questions relating to it. These questions are also processed by spaCy to generate named entities for comparison.

In order to generate a more targeted paragraph, we check and see if the question at hand has any entities of the same type as each sentence in the paragraph. For each sentence, if there is no overlap with the question, it is discarded for that question.

The most difficult challenge to overcome using this method is aligning the answers to the new passage. The simplest way to do this is count all of the words that were removed before the index of where the answer can be found. This occasionally causes issues when the sentence that contains the answer is removed from the passage. In these cases, I simply ignore the results and use the originally generated sample.

I also came across and issue with the PyTorch library and cuda-enabled tensors. Padding the tensors requires cpu-mode but there is no way to do it. See the following issue that’s yet to be fixed. <https://github.com/pytorch/pytorch/issues/16542>

Results & Conclusion

Unfortunately, results beyond a loss of negative infinity were never achieved. Despite this shortcoming, full integration with spaCy appears to fully allow additional avenues of text processing. spaCy, as a library on its own, enables a vast array of functionality but essentially requires reimplementing many of the features already present in the original DrQA model. This creates significant overhead to both training time and development time to integrate the two.

Creating a custom pipeline in spaCy should theoretically allow integrating PyTorch in any or all the stages. What it does not easily enable is implementing spaCy within PyTorch. spaCy uses its own internal representations of models that require duplicating functionality of the DrQA model.

Even without creating a custom trained model in spaCy, entities were recognized very well. Additionally, most passages could be shortened to fit possible domains that a question could be asking about. Manual analysis of individual results suggested that the preprocessing should provide noticeable speed increases.

Future research on better analyzing this dataset should probably start with spaCy and integrate pyTorch for certain elements when beneficial. Most operations could be completed in spaCy alone, taking advantage of the optimizations already present in the library.

Additionally, research from Facebook showed promising results by conducting NLP over entire sentences rather than just words. The last reference has more information on this subject.

Acknowledgments

Thank you to the various references for the information they published and for our NLP Professor for providing the original files. All discussed files can be found on my forked Github repo here: https://github.com/zachhardesty7/cs378nlp-sp20-fp#using-pre-trained-models

References

https://arxiv.org/pdf/1606.05250.pdf

https://arxiv.org/pdf/1806.03822.pdf

https://rajpurkar.github.io/SQuAD-explorer/

https://github.com/explosion/spaCy

https://spacy.io/usage/training#ner

https://towardsdatascience.com/building-a-question-answering-system-part-1-9388aadff507