In this project, you will be using recurrent neural networks and transformers for text classification. You are welcome to work in either or both of Tensorflow and Pytorch. As we've done before, you are encouraged to create Jupyter notebooks for your scratchwork, and then you will eventually submit a write-up notebook with a polished presentation of your results.

Partners and Groups

You are required to work with your assigned partner on this project. You are responsible for ensuring that you and your partner both fully understand all aspects of your submission; you are expected to work collaboratively and take responsibility for each other's learning.

You will also be assigned a code-review partner after the deadline. To prepare for that code review, make sure you understand all of your code and that it is cleaned up to be readable by people outside your group. See the code review guidelines for more information.

Objectives

This project has three primary goals; it will give give you experience with:

1. pre-processing and training on text data,
2. fine-tuning recurrent and transformer architectures,
3. experimentally comparing the performance of different NLP models.

As with the last project, you will be learning to use many existing tools that won't be explicitly introduced in lecture. You are once again encouraged to look for tutorials and examples that use these tools for both inspiration and guidance. Once again, the data sets come from Kaggle, and you are encouraged to explore what other people have done with these and similar data sets.

Data Sets

Both of the data sets we will be working with are located in the folder `/home/DAVIDSON/brwiedenbeck/public/NLP`. The first data set contains questions posted to Quora that we'll be classifying as sincere or insincere. The second data set contains statements rated by

Politifact that we'll be classifying according to their truthfulness scale. As usual, you should begin by creating a symbolic link to the data folder with the `ln -s` command.

Both of these data sets will require you to read in a text file and do some basic processing to extract the text and the labels. It may be a good idea to save a small subset of each data set into a file that's in an easier-to-process format so that you can test out basic operations without having to deal with all the data at once. If your training runs into memory pressure, you may also want to explore splitting the data sets into smaller files that can be loaded individually.

Data Pre-Processing

Both data sets will require some pre-processing to get the data ready. First, you'll need to extract the relevant text and labels, and then create a train/validate/test split (and probably also debugging set with more like 1% of the data). You should not add any of these data files to your Git repository; if you add the filenames to the `.gitignore` file, this will help ensure you don't commit them by mistake. As usual, you are encouraged to use existing tools rather than rolling your own, so look into how other people have processed the data, and what tools pandas and scikit-learn have for processing the data.

You will also need to prepare the input data in a way that's appropriate for each of the models you will be training. This will require extracting tokens from the questions and encoding those tokens. The different types of models we will be using—recurrent networks and transformers—require different types of tokenization and encoding. You should look into documentation and examples for the specific models you choose and ensure that you are preparing appropriate inputs, but the Keras and Hugging Face tokenizers/encoders are likely to be helpful.

Training and Transfer Learning

Once you have processed the data, your first goal is to train some sort of recurrent neural network (for example an LSTM) to classify the data sets. The Quora dataset has lots of example notebooks, but note that since that competition is several years old, many of the interfaces have changed, so you may want to seek out other examples. An important note is that it will pay dividends to test things out on a small scale before trying to train on large data sets: make yourself a debug-set and use it to help getting set up! Remember to keep a record as you go of the things that you've tried and how well they worked (I recommend a

spreadsheet), since you will need to explain your choices in the writeup.

In addition to training a recurrent network, you should also try using some sort of transformer model (such as BERT or GPT). However, note that the common transformer models are extremely large, and require enormous amounts of time and memory even if we're just doing fine-tuning for transfer learning. You should therefore look for examples (in the standard model zoos ore elsewhere on the web) of smaller versions of these models that can plausibly be trained with the time and resources we have available.

For any models you choose to use, you should read about where they come from and what data they were trained on. In your writeup you are expected to cite your sources, explain your results, and justify the model(s) that you tried. Note that different models may have different input requirements which may require modification to your pre-processing of the data. Note also that pre-trained models vary substantially in size and architecture, which can have significant impacts on training time, memory use, and performance.

Experiments

In total, you will be trying out several different approaches to architecture, training, and data preprocessing, that in the aggregate amount to quite different approaches to solving the problem of flagging troll questions. You should therefore run some actual experiments comparing performance between the best versions of each model. Specifically, you are required to run and report on an experiment comparing an LSTM-based approach to a transformer-based approach. You should also provide experimental evidence that you have thoroughly optimized each model. Discuss the alternative parameters and approaches you considered and show why the ones you settled on are best.

Also think about what metrics make sense for comparing different models. For example, the Quora dataset has very lopsided label categories, so simple accuracy may not be appropriate. In your write-up you may want to comment on precision/recall tradeoffs between different model variants.

Writeup

Once again, you will submit a `writeup.ipynb` notebook with a clear explanation of what you've implemented and what you found in your experiments. This should show comparisons on both problems between the recurrent and transformer models you trained, along with discussion of other approaches you tried. Explain what went into training each model variant,

how it performed, and what you learned. You should use markdown features including headings, paragraphs, lists, and so on to make your write-up read like a blog post. Think of this as a mini-version of the sort of write-up you will be doing for the implementation project at the end of the semester.

As a part of your write-up, you should include citations to all resources you used in coming up with your solutions. The expected format for citations is a numbered list of web-links, along with a one-sentence description of what you learned from each resource, and in-text references to those citation numbers.

Resources

Once again, our data sets come from Kaggle. *You are encouraged* to read other people's Kaggle notebooks and learn from them; don't try to figure everything out on your own! You may also want to look around the web for other examples of notebooks for *related* NLP problems, since the Quora dataset is a few years old and from a competition with weird rules, so you may find that the notebooks have different objectives and old library versions, while the Politifact dataset doesn't have an associated competition, and therefore has far fewer notebooks. The same ground-rules from last project still apply. You may:

- view any notebooks and other examples you wish, and
- take notes on any functions they use and how you think they might be useful.

But you may not:

- have both someone else's notebook and your own open at the same time, or
- directly copy any amount of code from someone else.

The key idea here is that you will use existing notebooks as a source of ideas for the sorts of tools you might want to use, but you will then read the documentation on those tools to figure out how to use them yourself. If you're uncertain about what's allowed, don't hesitate to ask in class, in office hours, or on Slack. Make sure you keep track of links to any notebooks or other resources you find useful so that you can cite them in your write-up.