Capstone Project 2 Milestone Report

The Problem

Simply put, the problem/challenge that this project will attempt to 'solve' is that of *Question Answering*, an NLP discipline that is concerned with building systems that automatically answer questions posed by humans in a natural language. The goal of the project is to produce a closed-domain question answering system that (details on the dataset later), in an ideal case, will correctly answer answerable questions and not attempt to answer unanswerable questions. Furthermore, the project will serve as a personal challenge to develop in areas of Deep Learning and NLP.

Data Acquisition/Wrangling

The dataset is the '*Stanford Question Answering Dataset*' (SQuAD), it's a reading comprehension dataset, consisting of questions posed by crowdworkers on a set of Wikipedia articles, where the answer to every question is a segment of text, or span, from the corresponding reading passage, or the question might be unanswerable. It consists of 150,000 question-answer pairs, 100,000 of those questions are 'answerable' and 50,000 are 'unanswerable'. The aim of the dataset is to encourage further research and development in the field of Question Answering and NLP. More can be found at: https://rajpurkar.github.io/SQuAD-explorer/.

The data is in a JSON file so the acquisition process is as simple as downloading the file and reading in the file via pandas. The goal of the wrangling step is to transform the data from the JSON format into a pandas dataframe where each row is a question, and the columns are: the question, answer, context of the question, if the question is answerable etc.

After analysing the JSON it was clear a simple import was not possible, hence the need for nested for loops. We looped over the Wikipedia articles present in the dataset, and then over the 'contexts' (paragraphs in the Wikipedia article), extracting the questions/answers (along with other fields) for each context into a dataframe and appending the dataframes to a 'master dataframe'. We also add the 'context' and 'title' of the Wikipedia articles as columns. The 'answer_data' function was defined in response to the differences in JSON format for the answerable and unanswerable questions to extract the 'start_index' and 'text' fields for the answer. As mentioned above, simply put, the data wrangling steps taken above can be best summarised by going over each theme and each context in turn and extracting the relevant information to create rows of data in a dataframe, the details of the implementation are a result of the structure of the JSON file.

Word Vectorization

I make an assumption at this point that the reader knows about word vectorization to some level. 'One hot encoding' as a word vectorization method, while simple, is not very 'useful' for many NLP tasks, as it does not encode any information about the relationships of words with one another etc. In the context of Question Answering this is especially important as the system relies on some kind of 'comprehension' of the questions asked of it to be able to provide an answer. To that end, after doing a moderate amount of research 2 main word vectorization 'methods' stood out, word2vec and GloVe. As also discussed in the notebook, given the slight edge in performance in most NLP tasks, we chose GloVe's pre-trained word vectors over word2vec's skip-gram word vectors.

GloVe has pre-trained vectors for various corpora including *Wikipedia 2014*, *'Common Crawl'* (a crawl/scrape of the World Wide Web) and *Twitter*. Given the differences in the style of language, amongst other aspects, it makes sense to use word vectorizations pre-trained on a corpus that is most alike the dataset we are using for this project (SQuAD). Given that the dataset is a bunch of Wikipedia articles along with questions and answers, it makes most sense to use GloVe's *'Wikipedia 2014 + Gigaword 5'* pre-trained word vectors.

We will not re-train the word vectors with either method, as the file/corpus size (40MB) is not large enough to guarantee an increase in performance over the pre-trained variants. Furthermore, given the simplicity in changing out the word vectors to those of higher/lower dimension (the size of the vector representing each word), we will try various word vector sizes (dimensions) as this effectively a hyperparameter of the model. Similarly, we will be able to do this with word2vec's word vectorizations too.

In conclusion, GloVe is the initial word vectorization method of choice, we will use pre-trained word vectors, and in time, we will try out various word vector dimensionalities from GloVe and word2vec providing it is time efficient to do so.