Report on Phase 1

A Joint Model for Multi-Task Question Answering

Mentor: Man Luo(mluo26@asu.edu)
Authors: Maitri Dodiya(mdodiya@asu.edu)
Matthew Huff(mwhuff@asu.edu)

Abstract:

This report provides details of the data obtained, methods employed for data processing, models created and future work for our final project- "A joint model for Multi-Task question Answering".

Introduction

HotpotQA is a diverse explainable multi-hop question answering dataset introduced because existing datasets failed to train QA systems to perform complex reasoning and provide answers for questions. This dataset contains 113k question-answer pairs extracted from Wikipedia with four key features:

1) The questions require finding and reasoning over multiple supporting documents to answer.

2) The questions are diverse and not constrained to any pre-existing knowledge bases or knowledge schemas.

3) Sentence-level supporting facts required for reasoning are provided, allowing QA systems to reason with strong supervision and explain the predictions.

4) New types of factoid comparison questions are included in the dataset to test QA systems' ability to extract relevant facts and perform necessary comparison.

Our goal is to create a natural language processing model trained on HotpotQA which can:

a) Select relevant documents from a set of documents
b) Answer a question
c) Identify supporting facts which reach the answer

A joint model architecture capable of selecting relevant passages and identifying supporting facts but not the span has been provided by our mentor Man Luo. Our task was to

update the data processing element of the code by adding extra features which will be needed to solve this problem and change the model such that it contains these extra features and a loss function for them. We are extending the current model with an answering head and evaluating the resulting model on HotpotQA.

Method

To do so, we first referenced papers on HotpotQA and BERT-model to better understand the model and how it works. We took some time to look through the code provided such as process_hotpotQA.py, extract_answer_label.ipynb, utils.py, and glue_dataset.py as these all contained critical functionality for making the model work. Once we had some idea of the overall base code set, we began extending the data processing given in the base code given for this project. The base code took the raw data and processed it into chunks, each chunk representing a set of documents and labels for their relevance in the question asked. We added a label for the model to determine whether the answer to the question asked resided in that set of documents. Our answer label works by checking for relevant terms in the title of each set of documents and for the answer keywords within the paragraph itself. We then added the start and end positions of the tokenized text by searching for the answer keyword within the paragraph text. We were then able to reprocess the raw HotpotQA data to fit the format for our new model.

We then worked on extending the JointQA model to complete this project. We had to update the functionality of the JointQA model by adding a start_position parameter and end_position parameter to the forward function of the network. To obtain logits for the start and end position, we added a question_answer layer with two outputs, the start and end position logits. We updated the loss function to include loss from the start and end positions using BertForQuestionAnswering as a guide. We averaged the Cross Entropy Loss from the start_position, start_logits and the end_position, end_logits and added this to the overall loss.

We also had to make some slight changes to the utils.py and glue_dataset.py files to get them to work with our altered dataset by including the start_position and end_position in the feature set for the DataLoader for Torch. Because of some emergency medical issues, we paired down the training examples to get the model to finish training quicker. We used roughly 35% of the training examples. We will attempt this model against the full set once this is submitted to see how it performs, but our losses decreased to reasonable amounts in that amount of training time.

Individualized Contributions

Matthew Huff:

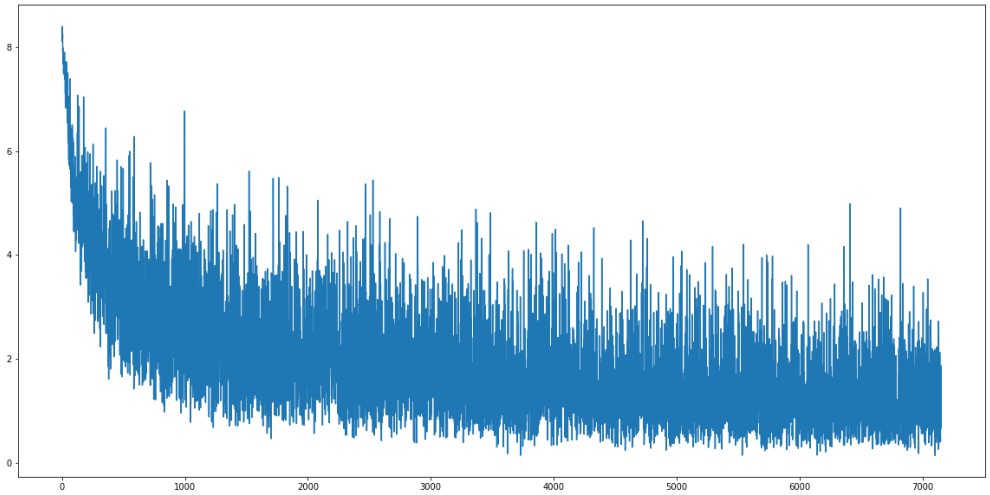
Matthew worked on changing the data processing code to include the start and end position and changing the extract_answer_label code to reprocess the data into a format that would work for this project. Matthew then worked on modifying some of the functions in utils.py and glue_dataset.py to make everything fit together, and also helped with updating the model.

Maitri Dodiya:

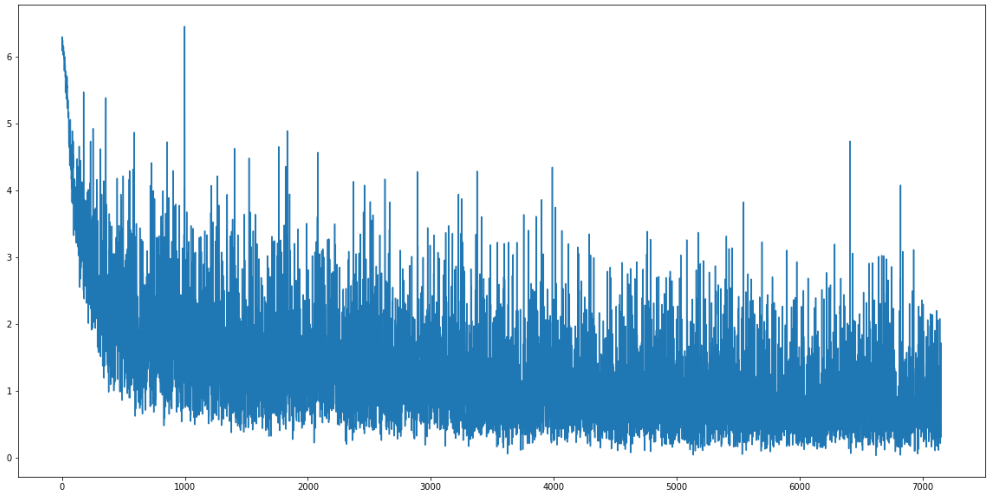
Maitri worked primarily on the model portion of the project, testing the necessary changes to the forward function and setting up the model in an online environment. Maitri also identified portions of the utils.py and glue_dataset.py that had to be updated or changed in order to get the code to run smoothly.

Results:

Our total loss decreased significantly to, by the end of training, hovering between 0.5 to 1.75.



The loss of the start and end positions (loss3) decreased from ~6 to a value between 0.35-1.75, depending on the training instance.



A text file containing a list of total losses (first array) and loss3 losses (second array) will be provided with the code and other materials.

1) HotpotQA
2) Man Luo's code