Project Milestone:
Cloud Infrastructure for Portable Chatbot

Kennedy Fairey, Nathan Dolbir, Anirudh Oruganti

Introduction and Problem Statement

Large-scale neural network (NN) models have emerged as innovative solutions for an infinite amount of predictive and interpretive tasks, but they require an extreme amount of computational power in order to yield fast and precise results. The devices most portable and actively used, such as smartphones or smartwatches, would be excellent application areas of large NNs, but they can't run strictly on their own hardware. The solution lies in cloud computing. Highly developed machine learning systems such as Apple's Siri and Amazon's Alexa are able to deliver intelligent responses by offloading computing power to the cloud, but these systems have an unbelievable amount of people, resources, and finances pumped into having these systems work without flaw. In order to spring forward innovation in all areas of ML, cloud infrastructure must be made possible for many developers with different interests, not just the tech giants. Our goal is to create a full-stack cloud solution to deploying large-scale NNs on portable devices, and which is understandable and reproducible. Our infrastructure's implementation area is the general-usage chatbot, as the actual purpose of the chatbot is not the focus of the project, but may be defined later.

Technical Approach

From front to backend, we start with the user interface, which is written in React, an interface framework for JavaScript. This application reads and replies to messages like a chatbot. The app is accessible on any web browser, and is formatted correctly on any type of device. Being deployed and hosted on Google Firebase, the app stores and displays data directly from the Firestore message database. When a user sends a message on the app, the message is stored in this database, and the bot's response process begins. A Google cloud function keeps watch of the Firestore database, and when a new message is added, the text of the message is taken into the cloud function's script, and the script reaches out to the appropriate neural network stored in the cloud to run and produce a reply to the user's message. After the response is made, it is stored back into the message's document within the Firestore database. While this is happening, the UI awaits the answer within the message's

document while displaying a “I am thinking” message to the user. When the reply is stored, the answer is displayed, and the process repeats for each message. As the chatbot becomes more complex, the base of the infrastructure remains the same. For a chatbot that must have a classifier describe what the intent of the message is (information retrieval vs conversation), the classifier would run the infrastructure loop first and then store the intent in the database document, followed by the appropriate model being activated to run through the message and create a response. The system boils down to interprocess communication, with cloud functions being the main mode of activation.

As far as the model itself goes, so far we have been developing the chatbot based on TF-IDF and Sequence2Sequence principles model, but we have arrived at a crossroads with this type of design. After recent discussion, we decided that it would be better to implement a transformer architecture for this project since transformers are the core of the large-scale neural networks recently, and we want to build this infrastructure to work for present and future models. We are using Google Cloud Services and Colab to train, test, and run custom-created models for interpreting text.

The TF-IDF model we have developed processes and replies based on input. The model uses a “Bag of Words” approach for its vectors. This model we have developed takes in an input sentence and outputs an intent of the sentence. The model is built using three Dense layers with keras integrated api in tensorflow and tools from Natural Language Toolkit and SKLearn. We expect to use the same tools for making the transformer model.

Intermediate/Preliminary Results

Our project is far from complete, but our tribulations and curiosity have led us to a promising spot to build on with a clear course of action. We have spent ample time learning about the blocks we need to create the project, and have started piecing them together. As a team, we split responsibilities while keeping collaboration key to the project. One member is mainly in charge of the cloud aspect of the project: learning about the GCP, how to create a Firebase application, and data synchronization across processes and the pipeline. Another is responsible for minor bug fixes, learning how our model will be able to process data, compiling results, and compiling relative project documents.

The results from the TF-IDF model we have built could be seen in Figures 1-3. We tested the model using a dataset

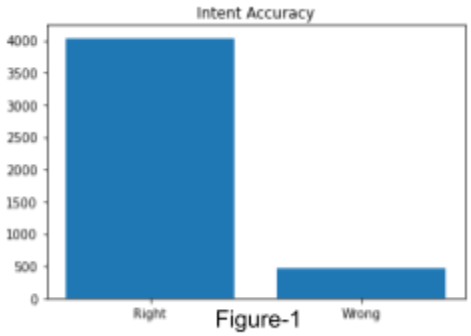

Figure-1

that consists of 4500 sentences with several different intents. At first we were looking to plot the results using a confusion matrix but shortly after we realized that the dataset has over a hundred different intents, instead we plotted using a bar chart of the number of correct and wrong guesses it made as seen in Figure 1. Figures 2 and 3 show training and validation accuracy and loss. The validation accuracy lines up with testing results as it converges at 89%.

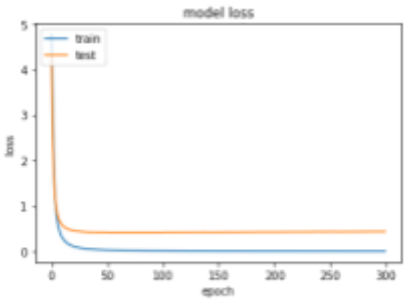

Figure-2

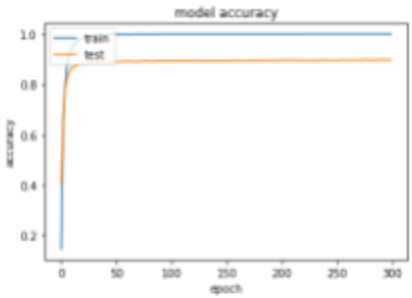

Figure-3

The infrastructure has not been implemented completely yet, but all the pieces are there. We have deployed the application UI, and the chatbot works correctly, but we have not yet developed the necessary cloud function to run the user message through the model. We have a 4GB Google T5 model which runs correctly through Colab script and also the chatbot app itself, we just need to connect the two.

Try the project out for yourself, Dr. Jamshidi! The “naming” part is a small glitch we’re working to fix, just reload the page after you name yourself and it will work again.

https://oval-tuner-326314.web.app/

Going forward, our main priorities are to get the cloud functions running the T5 model correctly and to build our own custom transformer-based mode.

Here is an example of the T5 model we created and plan to implement in the infrastructure

```
for question in ["trivia question: where is the google headquarters?",
                 "trivia question: what is the most populous country in the world?",
                 "trivia question: how is dr. jamshidi's day going so far?",
                 "trivia question: who is the worst basketball player of all time?"]:
    print(answer(question))
```

```
theresa may house (also known as bose plaza) in woodlawn, maryland, united states
china
good
draymond green
```