Eric Liu, Joan Zhu, Jun Zhang, Rick Yang, Shun Lin
Group 4 • Project Paradigm Chatbot

# Final Report

## Client Introduction

Our client is Paradigm, a chatting software for traders in the cryptocurrency trading market. Paradigm provides various features such as counter party discovery, text-to-trade recognition, and OTC cryptocurrency trading.

## Project Overview

The core challenge of our project is how to improve the experience of the conversational marketplace for cryptocurrency traders that are using Paradigm. We wanted to add a feature to the Paradigm app that increases its functionality and ultimately makes the product better. After conducting some basic industry research and meeting with Paradigm members, we decided that our main focus was on performing price prediction. By using machine learning models to analyze news articles and their impacts on day to day prices, we can show Paradigm users the most impactful news articles of the day. The side effect of this is we can perform different news analytics operations, potentially detecting fake or misleading cryptocurrency news and either filtering them out if we deem them unnecessary or marking them if we think that it could lead to a price spike or dip. All in all, we could fold these features into a chatbot that can recommend news articles to Paradigm users, integrating price prediction and news analytics into one unified application.

## Our Solution

- **The Baseline Model**

We started with constructing a k-means clustering algorithm for classification, as this is the industry standard as a first step in classifying large datasets. The k-means clustering algorithms we explored were used to both predict the sentiment of an article as well as its impact on bitcoin prices. This was intended to aid us in classifying articles as well as explore how well a relatively simple model could predict

cryptocurrency prices. Unfortunately, our k-means results did not produce satisfactory results. In fact, our models predicted that every test article would lead to an increase in bitcoin price. As a result, we determined that this was not a good path towards the fulfillment of our goal and decided to pursue different models.

- **The Language Model**

We built a language model to better analyze headlines and content, which is especially useful for detecting fake news articles. We found that this was much more successful than our k-means algorithm. Utilizing Google Colab, we generated a notebook that used RNN/LSTM and tensorflow for our language modeling. Although according to our graphs of validation vs training loss our model was overfitting, we still generated relatively good results. We generated much better predictions and could better support our additional features. For example, our model showed that an article titled “Bitcoin price crashes” is a much more likely headline than “Bitcoin is legal in all countries”. The success of this model on predicting whether or not an article is unusual leads us to start the User Interface of our project, the chatbot.

In terms of technical architecture, we preprocess the data we have and turn sentences into a list of numbers. Then we generate the target for our input, which is a shifted version of the input. Then we pass the input and the target through the embedding layer, then through a basic LSTM, which we will modify such as adding Dropout or Multi-RNN layer to achieve a better result. Lastly, we use softmax loss to see how different our prediction is against the target sequence.

- **The Chatbot**

Because Paradigm currently does not yet support external application to build on their platform, we built a chatbot on Facebook Messenger to demo our model and our vision of the project. We have built our chatbot in a portable way such that it will be easy for a future team to port the chatbot to Paradigm's

platform. Our chatbot is the User Interface of our project, where user can get useful information and analysis of cryptocurrency-related articles in real-time.

- **Why it's useful**

Having such a chatbot can help Paradigm as it can provide the traders with a quick analysis of the articles they should read during the day of the market. Our chatbot can also help traders to identify articles that are just usual and stop spending time reading cryptocurrency articles that do not bring much information. Similarly, our chatbot can help traders identify which cryptocurrency article is the most unusual and help them reduce the time required to find those unusual articles. The language model we trained this semester is also capable of performing other tasks such as news headline generation and when further tuned is also able to detect fake news, which future team that works with Paradigm can easily pick up and complete. Furthermore, we have built a language model and preprocessing code that is easy to train on contents of articles instead of just the headline, which the Paradigm can utilize to analyze the contents of the articles instead of just the headline. Lastly, our chatbot is portable and it would be easy for Paradigm to use the code we have wrote for Facebook Messenger Chatbot to build a chatbot on their own platform for cryptocurrency traders on their own platform.

- **How it works**

When the user gives a command to our chatbot, our chatbot will first read and analyze the command through the Facebook Messenger API, which has its server hosted on Glitch. Our chatbot will prompt an error message if the command is not understood, else it will ask users on what type of cryptocurrency they would like to learn more about. After receiving the name of a cryptocurrency the user want to know more about, our chatbot will call the NewsAPI to retrieve a list of top articles about the particular cryptocurrency that the user requested and are published in the last 24 hours. After receiving those articles, our chatbot will send requests to our machine learning model (language model) in our backend and rank and analyze them based on the unusual score. What we call the "unusual score" is the negative

log loss recorded when we run a request through our model. The higher the loss, the less likely it is to see this sort of item in our model. After receiving the sorted list of articles back from our model, our chatbot will then process the list and put it into the format that Facebook Messenger API requested and send the analysis and links back to the user. Another functionality of our chatbot we have developed is the "unusual score" command. This command takes in an article title or an url from the user. If the chatbot receives an url, the chatbot will scrap the title of the article from the url using BeautifulSoup4. If the url is invalid or if it is not a link to a news article, our chatbot will detect it and prompt the user to give a valid url. Similar to how we process the other command, our chatbot will send the title to our language model and send the analysis and unusual score back to the user.

**Journey Over the Semester**

- **Critical decisions**

After we were selected for an in-class demo of our MVP and moved towards completing our chatbot, we looked for better free resources to host our model. Instead of using google colab, we found a free server website called Glitch, which provided a free tier that had 512MB of RAM and 200MB of disk space. While it was likely possible to get slightly better resources elsewhere, we decided to use Glitch for its ease of integration. We found that there existed an example that outlined the integration between Glitch and Facebook Messenger.

For convenience, we decided that the chatbot should be hosted on Glitch following the example. We set some default responses to common phrases and queries, and used the News API to pull the most recent, most popular news articles. Popularity here is defined by the News API, which only allows us to fetch a specified number of articles in order of most popular. Another decision we had to make was not integrating the chatbot with our machine learning algorithms. We felt it was too difficult to integrate everything before our MVP demo, so we presented the MVP as having a chatbot side and a machine learning side. In the end, we had a successful presentation and our chatbot worked flawlessly. From this

point onward, our goals were to integrate the machine learning model with the chatbot backend and tune the model as we see fit.

This is where we hit another major challenge. Integrating our machine learning backend with our chatbot backend was not as simple as we thought it would be. Glitch's Facebook messenger chatbot example used Javascript as the server backend, while our machine learning algorithms were all written in Python, making it difficult to put together. In the end, we determined that we could either connect Javascript to Python via sockets or reimplement our model in Javascript. We decided that it would be easier to learn socket programming than more Javascript, so we spent countless hours over several days looking into libraries and experimenting to see which one worked best. Finally, we eventually were able to perform simple socket communication using Flask for Python and Najax for Javascript.

Luckily, everything else worked out mostly well for us. We had to keep our model simple to keep it under the 512MB RAM limit and we had to make sure none of the files we saved to disk would exceed its 200MB threshold. As a result, we were limited in the number of modifications to our algorithm. Notably, we limited ourselves to displaying five news articles at once, to decrease the processing time and RAM usage.