Microsoft

Microsoft Summer Internship

Austin Huang

Summer 2019

Agenda

1

Introduction

General overview of what I did during my time here.

2

Approaches

Discussion of the various methods used to solve the problem.

3

Final Product

Description and Demonstration of my final application.

4

What I Learned

Rundown of the overall skillset gained through this experience.

5

Future Work

How can I extend this project for the future?

Introduction

Can I use public sentiment to predict the price fluctuations of cryptocurrencies?

Cryptocurrencies are **highly volatile** because:

- No institutional regulations
- Questionable long-term staying power
- Herd Mentality / easily influenced by emotions

Proposition

Because crypto is so easily influenced by public sentiment, it poses as the perfect candidate for sentiment analysis.

By creating a web app to analyze the public sentiment surrounding a keyword, for example Bitcoin (BTC), Ethereum (XRP), or Litecoin (LTC), I could hypothetically gain some insight into the seemingly erratic price fluctuations of such cryptocurrencies.

Methodology & Tools

Goal: Create a program to gather and process commentary surrounding a certain key term

What:

- Twitter API
- Tweepy – Python Library to Access Twitter API
- Azure Cognitive Services – Sentiment Analyzer
- Azure Web App Hosting
- (Power BI, MySQL database)

Why:

- I used Twitter because of its extensive docs and proprietary Tweepy library
- Azure Cognitive Services assigns a sentiment score (0-1)
- Initially, I stored data via database. Later on, used cache to store data.

Approaches

Approach 1

Procedure

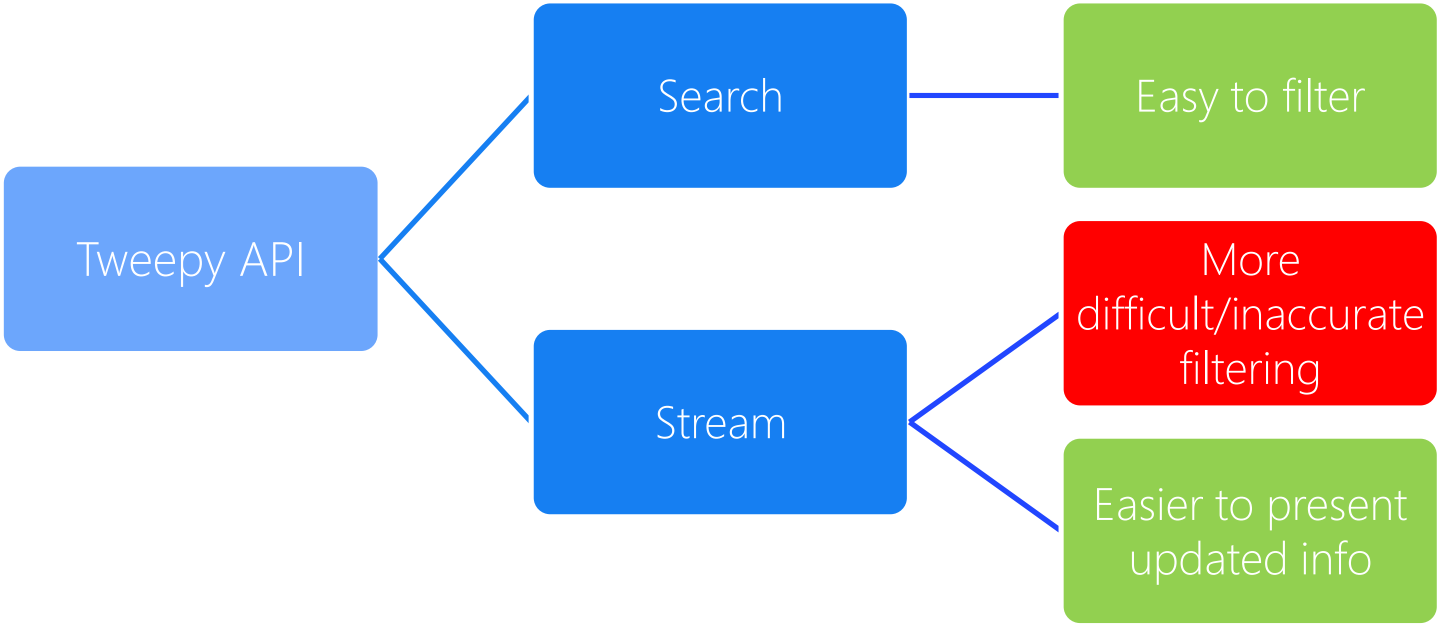
Twitter API

→ Process with Tweepy

→ Save Data

→ Analyze with Azure Sentiment Analysis API

Approach 1: Process with Tweepy



Tweepy API

Search

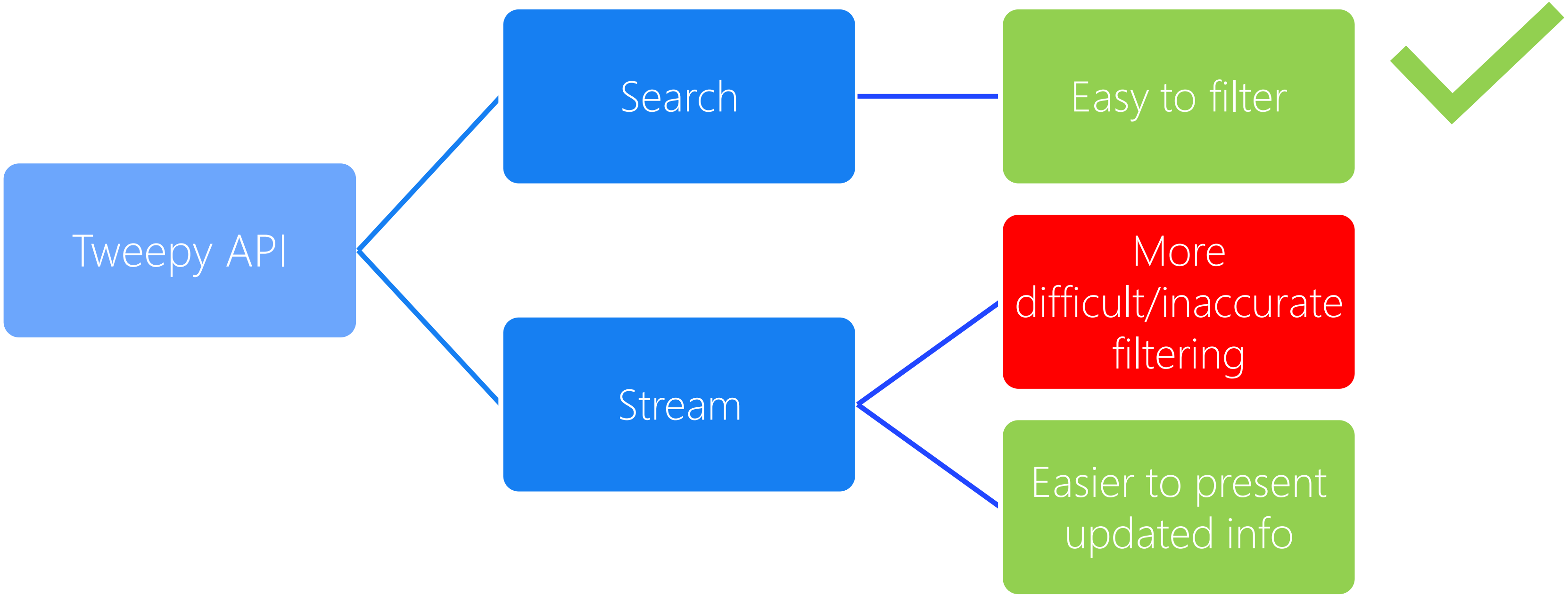
Stream

Easy to filter

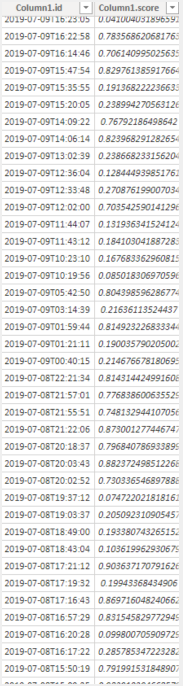
More difficult/inaccurate filtering

Easier to present updated info

Approach 1: Process with Tweepy

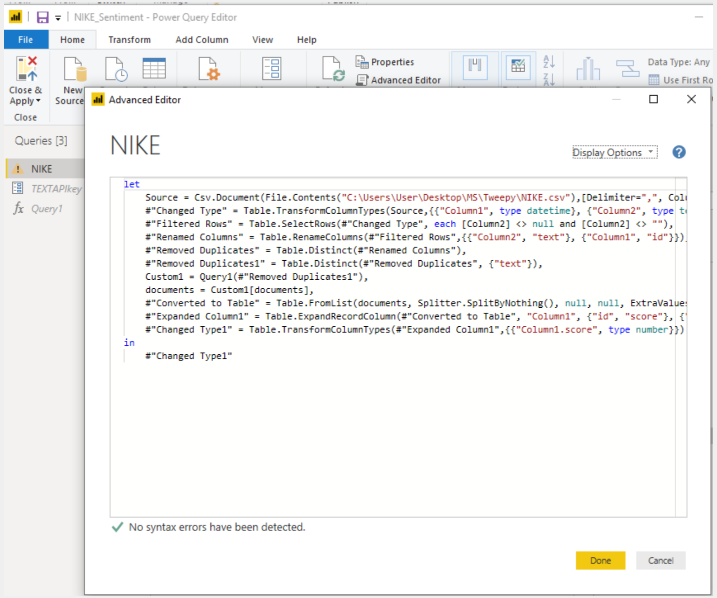

Tweepy API
Search
Stream
Easy to filter
✓
More difficult/inaccurate filtering
Easier to present updated info

Approach 1: Save Data



Column1.id	Column1.score
2019-07-09T16:23:05	0.04100403189659
2019-07-09T16:22:58	0.78356862068176
2019-07-09T16:14:46	0.70614099502563
2019-07-09T15:47:54	0.82976138591766
2019-07-09T15:35:55	0.19136822223663
2019-07-09T15:20:05	0.23899427056312
2019-07-09T14:09:22	0.76792186498642
2019-07-09T14:06:14	0.82396829128265
2019-07-09T13:02:39	0.23866823315620
2019-07-09T12:36:04	0.12844493985176
2019-07-09T12:33:48	0.27087619900703
2019-07-09T12:02:00	0.70354259014129
2019-07-09T11:44:07	0.13193634152412
2019-07-09T11:43:12	0.18410304188728
2019-07-09T10:23:10	0.16768336296081
2019-07-09T10:19:56	0.08501830697059
2019-07-09T05:42:50	0.80439859628677
2019-07-09T03:14:39	0.21636113524437
2019-07-09T01:59:44	0.81492322683334
2019-07-09T01:21:11	0.19003579020500
2019-07-09T00:40:15	0.21467667818069
2019-07-08T22:21:34	0.81431442499160
2019-07-08T21:57:01	0.77683860063552
2019-07-08T21:55:51	0.74813294410705
2019-07-08T21:22:06	0.87300127744674
2019-07-08T20:18:37	0.79684078693389
2019-07-08T20:03:43	0.88237249851226
2019-07-08T20:02:52	0.73033654689788
2019-07-08T19:37:12	0.07472202181816
2019-07-08T19:03:37	0.20509231090545
2019-07-08T18:49:00	0.19338074326515
2019-07-08T18:43:04	0.10361996293067
2019-07-08T17:21:12	0.90363717079162
2019-07-08T17:19:32	0.19943368434906
2019-07-08T17:16:43	0.86971604824066
2019-07-08T16:57:29	0.83154582977294
2019-07-08T16:20:28	0.09980070590972
2019-07-08T16:17:22	0.28578534722328
2019-07-08T15:50:19	0.79199153184890

Raw Data

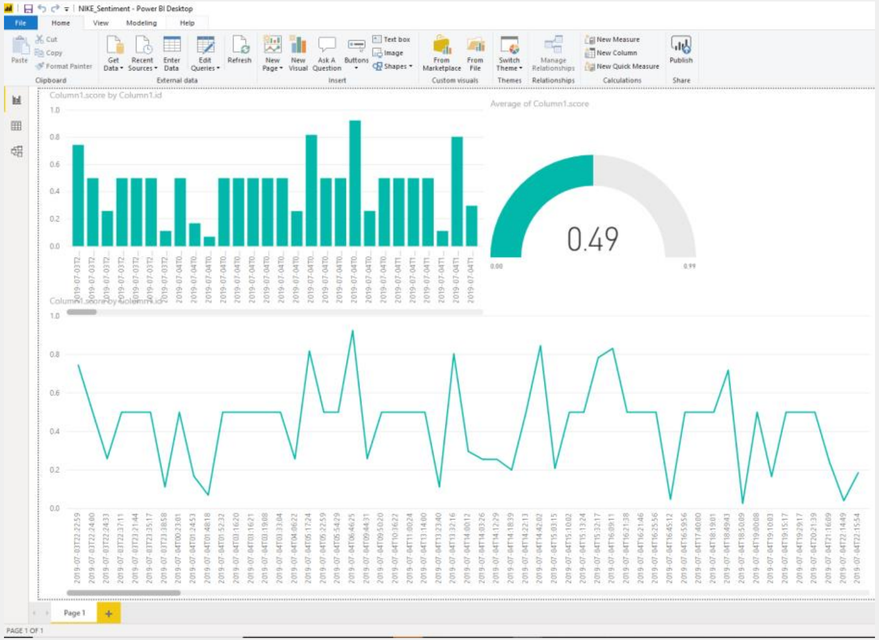


NIKE

```
let
    Source = Csv.Document(File.Contents("C:\Users\User\Desktop\MS\Tweepy\NIKE.csv"),[Delimiter=",", Colu
    #"Changed Type" = Table.TransformColumnTypes(Source,{{"Column1", type datetime}, {"Column2", type te
    #"Filtered Rows" = Table.SelectRows(#"Changed Type", each [Column2] <> null and [Column2] <> ""),
    #"Renamed Columns" = Table.RenameColumns(#"Filtered Rows",{{"Column2", "text"}, {"Column1", "id"}}),
    #"Removed Duplicates" = Table.Distinct(#"Renamed Columns"),
    #"Removed Duplicates1" = Table.Distinct(#"Removed Duplicates", {"text"}),
    Custom1 = Query1(#"Removed Duplicates1"),
    documents = Custom1[documents],
    #"Converted to Table" = Table.FromList(documents, Splitter.SplitByNothing(), null, null, ExtraValues
    #"Expanded Column1" = Table.ExpandRecordColumn(#"Converted to Table", "Column1", {"id", "score"}, {
    #"Changed Type1" = Table.TransformColumnTypes(#"Expanded Column1",{{"Column1.score", type number}})
in
    #"Changed Type1"
```

✓ No syntax errors have been detected.

PowerBI Advanced Query Editor



Processed Sentiment Scores

Approach 1: Save Data

MySQL Database

Saved data to MySQL database using MySQL API

Used MySQL Workbench to manage data

Why I ditched Approach 1

Initial approach familiarized me with the playing field.

Not ideal because:

1. Too many intermediate steps
2. Not user-friendly
3. Poor performance

Microsoft

Approach 2

Goal: No command line arguments, no unnecessary intermediate steps, faster performance, working web app.

What:

- Flask – Python Web Framework
- Tweepy
- TextBlob – NLP python library
- Azure Web App Hosting

Why:

- Flask is lightweight and scalable: doesn't require many tools & libraries to get started
- TextBlob – no need for more API keys and internet comm. (performance improvement). Also more in-depth docs than Azure Sentiment

Final Product

Goal: No command line arguments, no unnecessary intermediate steps, faster performance, working web app.

What:

- Flask – Python Web Framework
- Tweepy
- TextBlob – NLP python library
- Azure Web App Hosting

Why:

- Flask is lightweight and scalable: doesn't require many tools & libraries to get started
- TextBlob – no need for more API keys and internet comm. (performance improvement)

Final Product

Final Approach for Data Collection:

- Used Tweepy Streaming API – collects data faster based on personal testing. Also collects most recent data.

Final Approach for Data Processing:

- TextBlob library – easy to use, clear documentation, locally run.
- Matplotlib – plotting library for Python (not ideal because requires a lot (unnecessary) coding to implement.

  More on this in "*Future Work*" *Section*

Final Approach for Web Application

- Flask – popular, lightweight, scalable, quick start.
- Azure Web Hosting

Version Issues

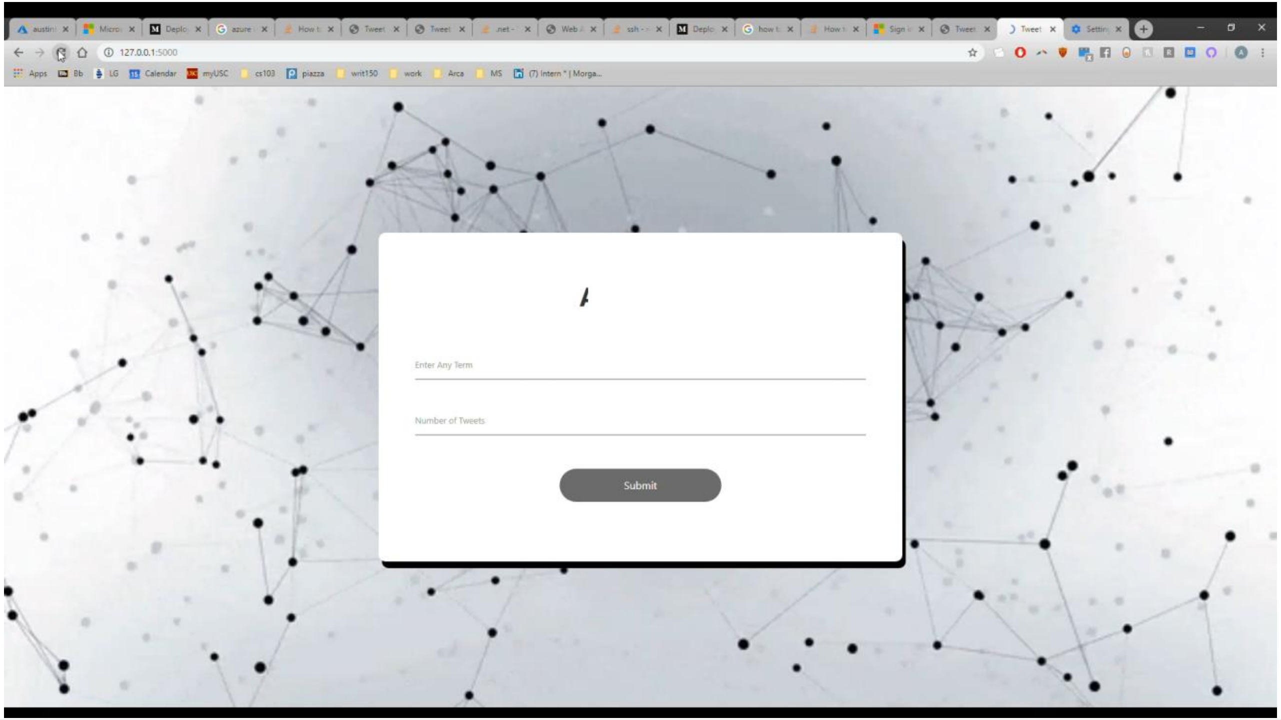
Flask-Cache Caching Error:

- Flask-Cache causes plots from previous searches to be saved within the app. I fixed this using a simple `app.config["CACHE_TYPE"] = "null”` command.

Unable to Deploy to Azure/Heroku:

- After numerous attempts to deploy to Azure Web App Services or Heroku, I was ultimately unsuccessful.
- I was able to create, deploy, and commit my local git to the cloud, but was unable to see the app run. Future troubleshooting with an Azure account with admin rights will prove more practical.

Web App Demonstration



Video Demonstration of App Running on Local Host

What I Learned

During my internship, I learned:

1. Cloud computing concepts, including IaaS, PaaS, and SaaS, how to create storage accounts, setting up cloud VMs, etc.
2. Python in the context of app development, including frameworks such as Django and Flask.
3. Using and leveraging APIs and libraries to make code more concise and more efficient.
4. Azure Cognitive Services, creating a chat bot using LUIS, Azure web apps.
5. App development workflow and timeline (from setting up to writing to publishing). Learned to prioritize certain functions during workflow.

Future Work

How I hope to extend my project

1. Create a more comprehensive dashboard to display and analyze collected data. Consider integrating PowerBI to generate more in-depth graphs.
2. Improve performance by speeding up tweet collection.
3. Providing real-time analysis of keyword tweet.
4. Showing feed of tweets to compare accuracy of sentiment analysis.
5. Testing various sentiment analysis algorithms for accuracy.

Thank You!

Austin Huang
Summer 2019