Distance Arbitrage aka “Distarb”

Pobody’s Nerfect!

By Diego, MSD 2022

Table of Contents

Introduction

This project is the realisation of an idea I had years ago: what if you make stock picks based off of user input mistakes, such as fat fingering or misreading the stock which they are trying to purchase. To do this, I would need to find a way to

1. Keep a list of stocks/companies available on the NYSE.

2. Find the edit distance of those companies of those companies based on both substitution/deletion, as well as accounting for the physical distance of those keys on the keyboard.

3. And finally as a fun aside, look at the stocks which members of the senate are buying and selling and visualising the value of those companies after executing the buy/sell order.

Background: What's wrong with existing solutions?

The current problem with existing solutions is that there is an issue of cost, as well as an overabundance of features. By their nature, they exist for the purpose of making the user money. The economy is driven to make more wealth, and the money poured into technology to make this “better” is in the trillions of dollars. A single Bloomberg Terminal, for example, is considered the gold standard of real-time stock tracking technology can have a per user cost of US$20,000 per year (Martel, 2022). Koyfin, a free alternative to the Bloomberg Terminal does not offer any sort of charting ability to the user. Which anyone who has looked at a stock’s performance will tell you that is a major deficiency. In addition, current programs offered by companies for stock tracking may provide tons of tools a user may want, but also provide many that a user might not need/want/care about. What sets my program apart even more is that I have yet to see stock analyser that uses a recommendation metric for stocks that is betting on humans being, well, human and making a mistake. Trying to figure out what ways a human may err is an interesting problem to me.

Design Choices and Libraries

For the sake of my sanity, I decided to use as much python 3 as I could. The fact that it has so many libraries, made it an obvious choice as a place to start. As Ben keeps telling me, he warned me about packaging. I do not deny there were issues that came from my choice to use python. I do not deny that packaging was the biggest issue. But, I stand by my decision. It was the right call to go with python.

While this is not an exhaustive list of libraries that I used, these are the ones that were of the most use to me in the development of Distance Arbitrage:

- PyQt5 – (/ˈpaɪ kjuːt faɪv/) a fast GUI library built on top of C++. It is free to use with an open source license. If you choose to monetise your application, the usage fees are substantial. More on this later.
- Yfinance – making API calls to Yahoo Finance to get stock data. It is very handy when you need to make general information inquiries about stocks, as it returns a lot of good information with each call. I used it for getting the price at day's open for my graphs.
- Clavier – to get the keyboard aware edit distances. This saved me a lot of time and allowed me to develop a more polished looking app. It is a small, lightweight library requiring python 3.8+ (mentioned because there were some issues. I know, Ben. I know.) and offers customisation for keyboard type e.g. dvorak and qwerty. As well as keyboard row offset specification, edit distance, and Euclidean distance calculator.
- Pendulum – for date manipulation. Pendulum is useful because it inherits from datetime. It has all the datetime methods, but also a lot of Pendulum specific

methods that are easier to use (in my opinion). It is also time zone aware. Which admittedly is not useful for my program. Time zones are a regular hassle in my daily life. I really appreciate this capability and wanted to give it recognition for that.

These are notable libraries that were useful in the creation of Distarb, but are common enough not to merit in-depth explanation:

- BeautifulSoup – for web scraping.
- Pandas – for data manipulation.
- Urllib – for web requests.

Organisation and Design

Class diagram

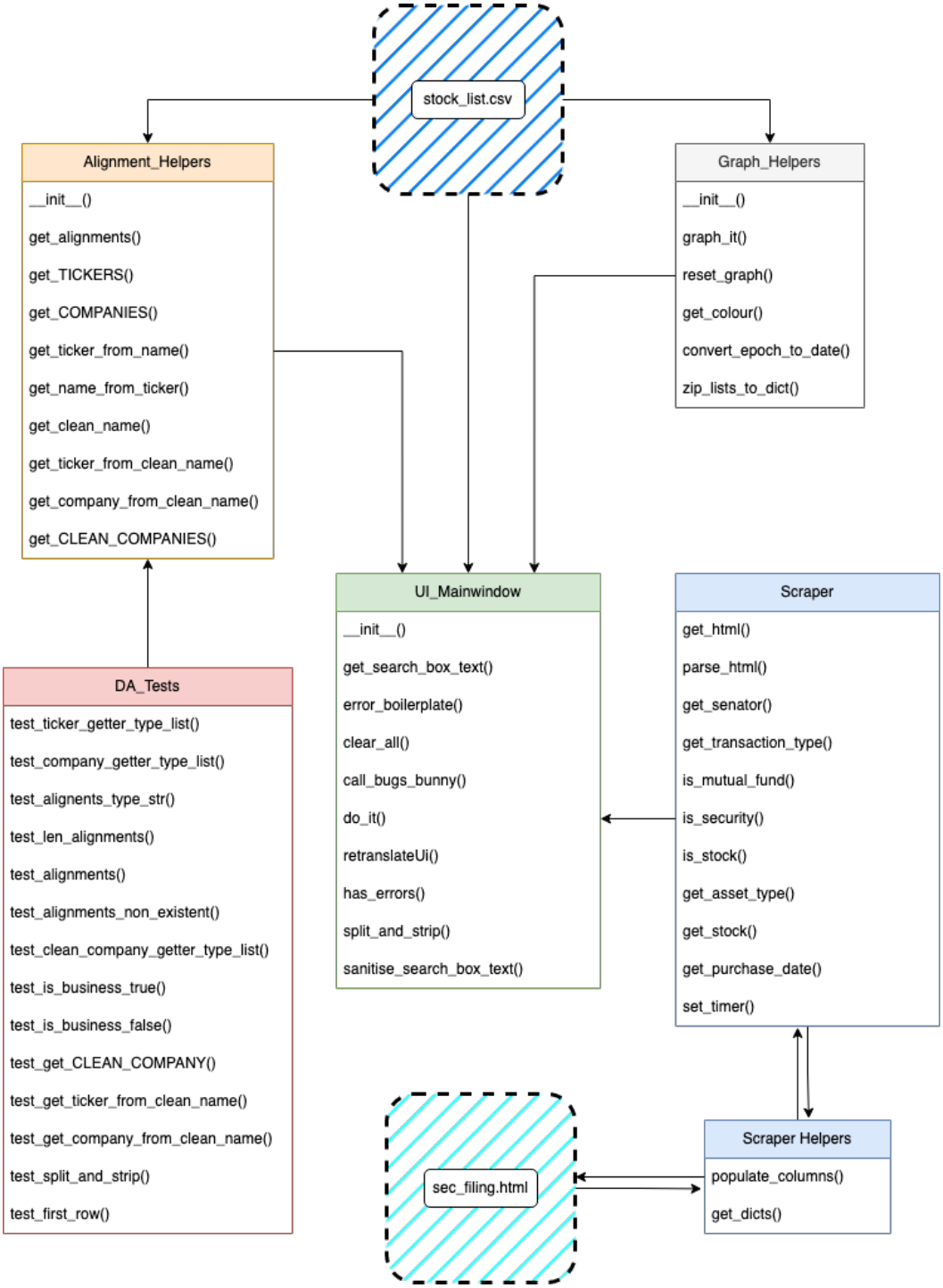

Figure 1 Internal organisation of Distarb

Results

Landing Page of the app

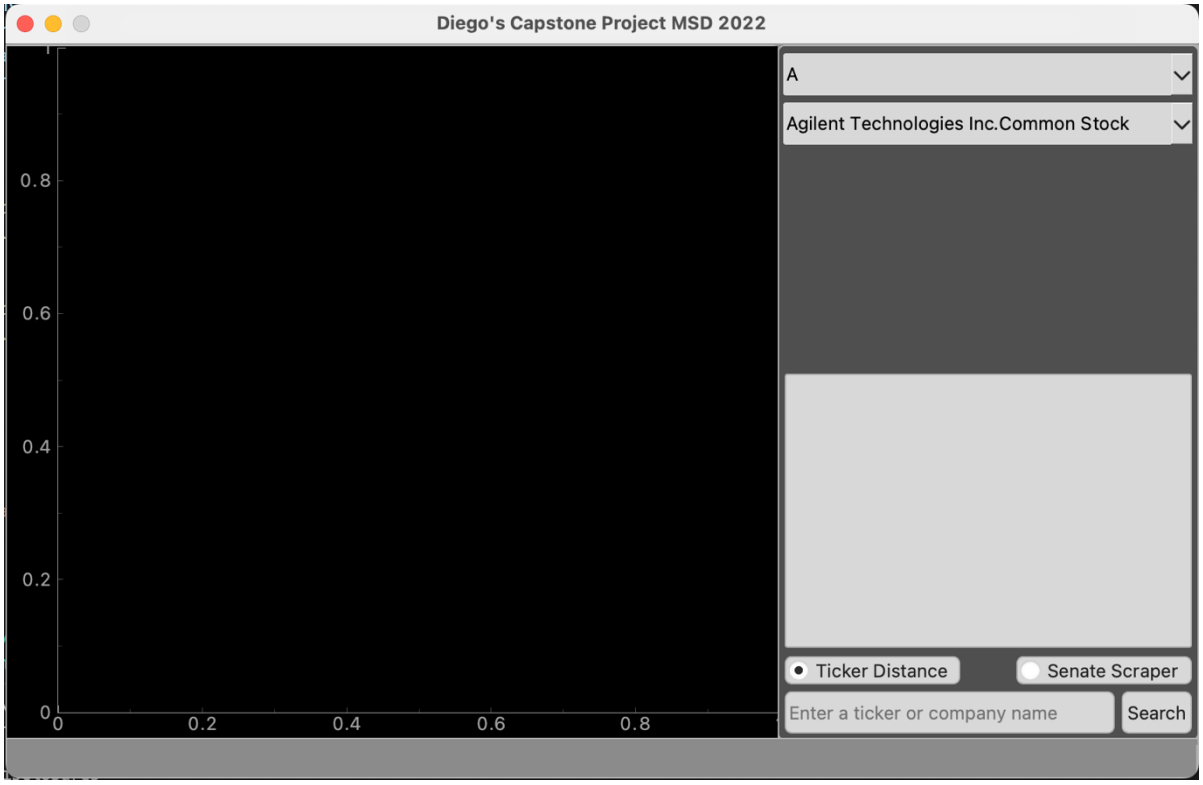

Figure 2 The program runs with a simple interface allowing the user to quickly and easily select the functionality they desire. The default is the ticker edit distance because that is as I see it, the primary use for the app.

User input

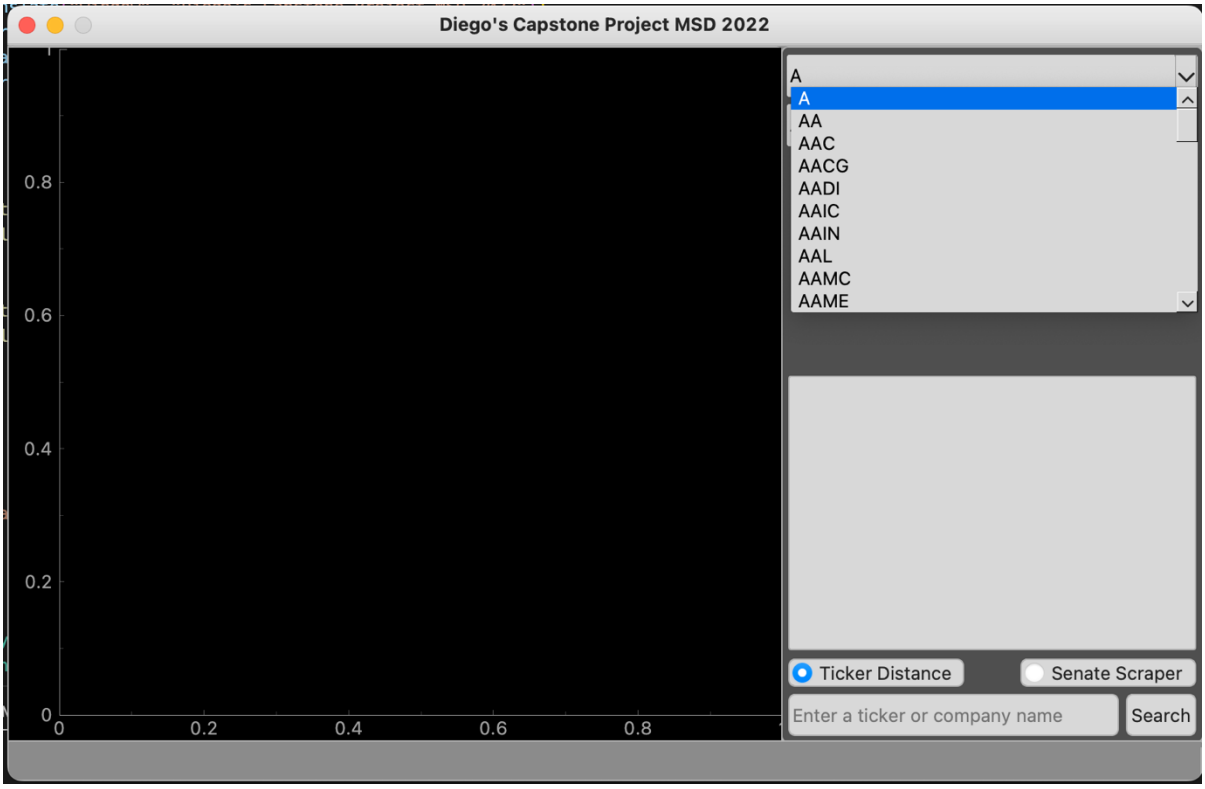

Figure 3 User can use a drop down to select a stock ticker

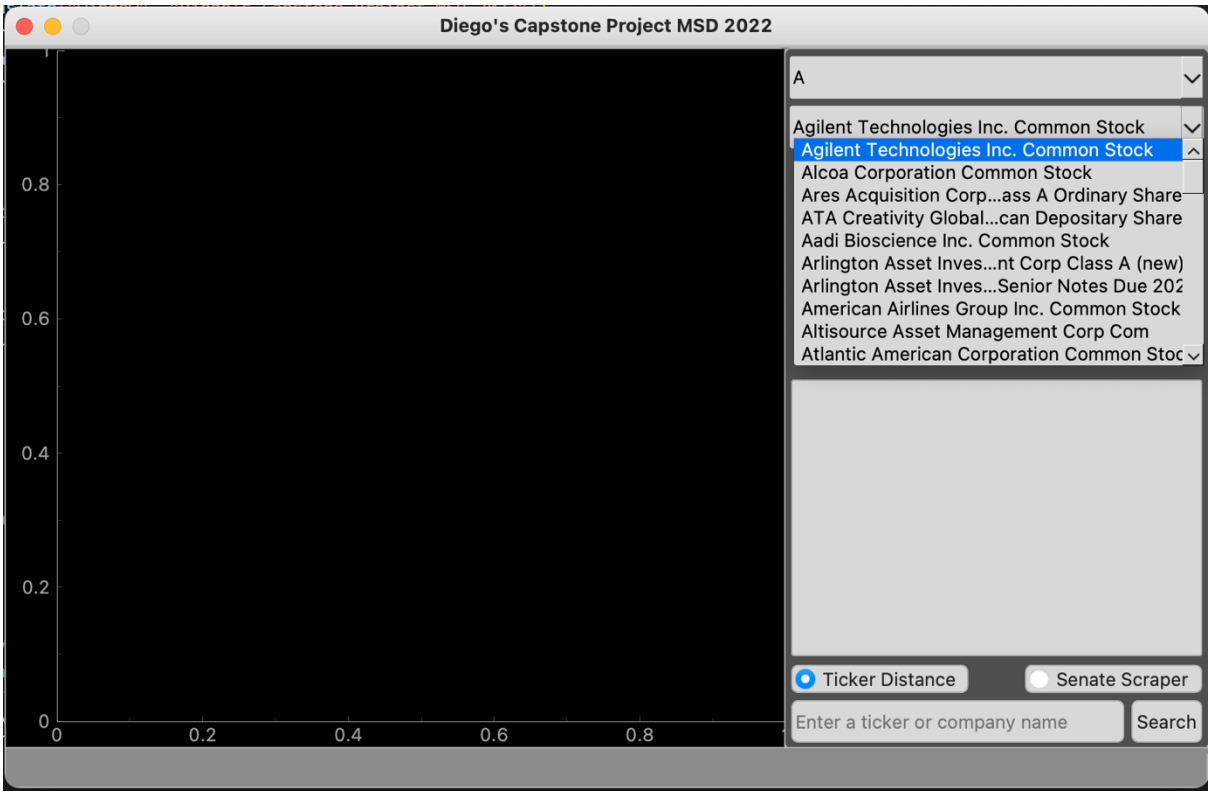

Figure 4 User can use a drop down to select the listed company name

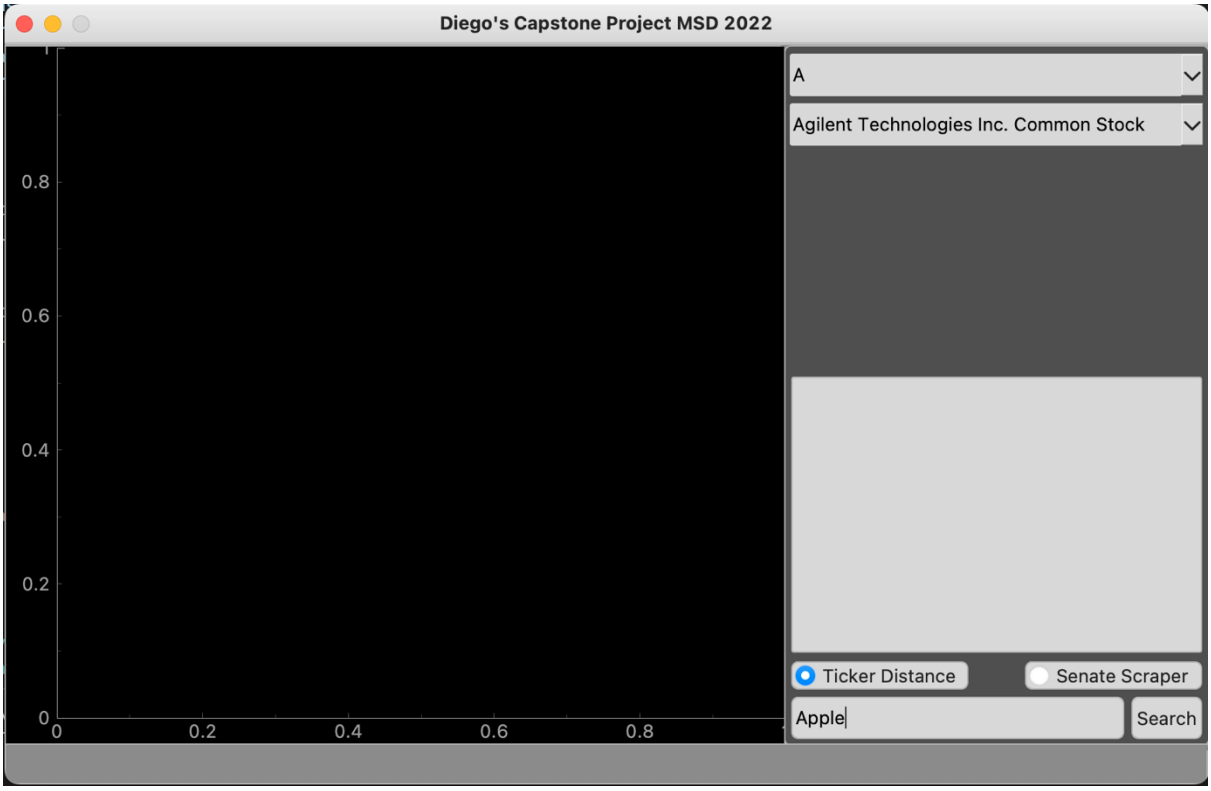

Figure 5 User can enter a custom search

The user is able to enter in their searches in two ways:

1. The user can use the dropdown menus at the top right of the application, and scroll through a list of stock tickers on the NYSE, or the registered company names. I wanted this option because as is the core of this program, mistakes can be made when user input is concerned, and removing typing from the equation when possible is a reasonable solution to the problem.
2. The user is able to manually enter a ticker/company name. This is so the user can search for companies that are not yet traded on the exchange that are about to IPO, as well as for companies that have a legal name vs colloquial name. Before Facebook became Meta, the stock ticker was FB, legally they were TheFacebook, Inc., then Facebook Inc., and colloquially we just call them Facebook. Giving the user the ability to search for what they feel is the most common name of a company adds more flexibility.

Edit Distance

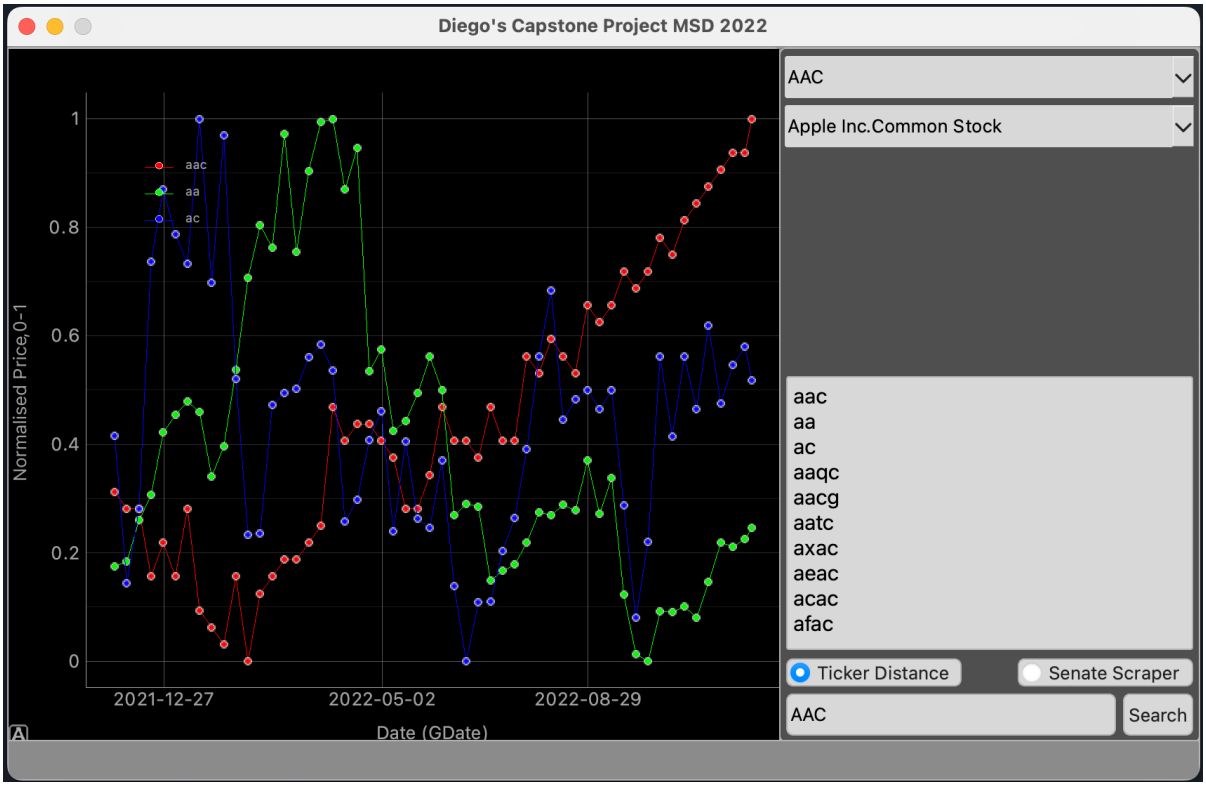

Figure 6 edit distance of tickers

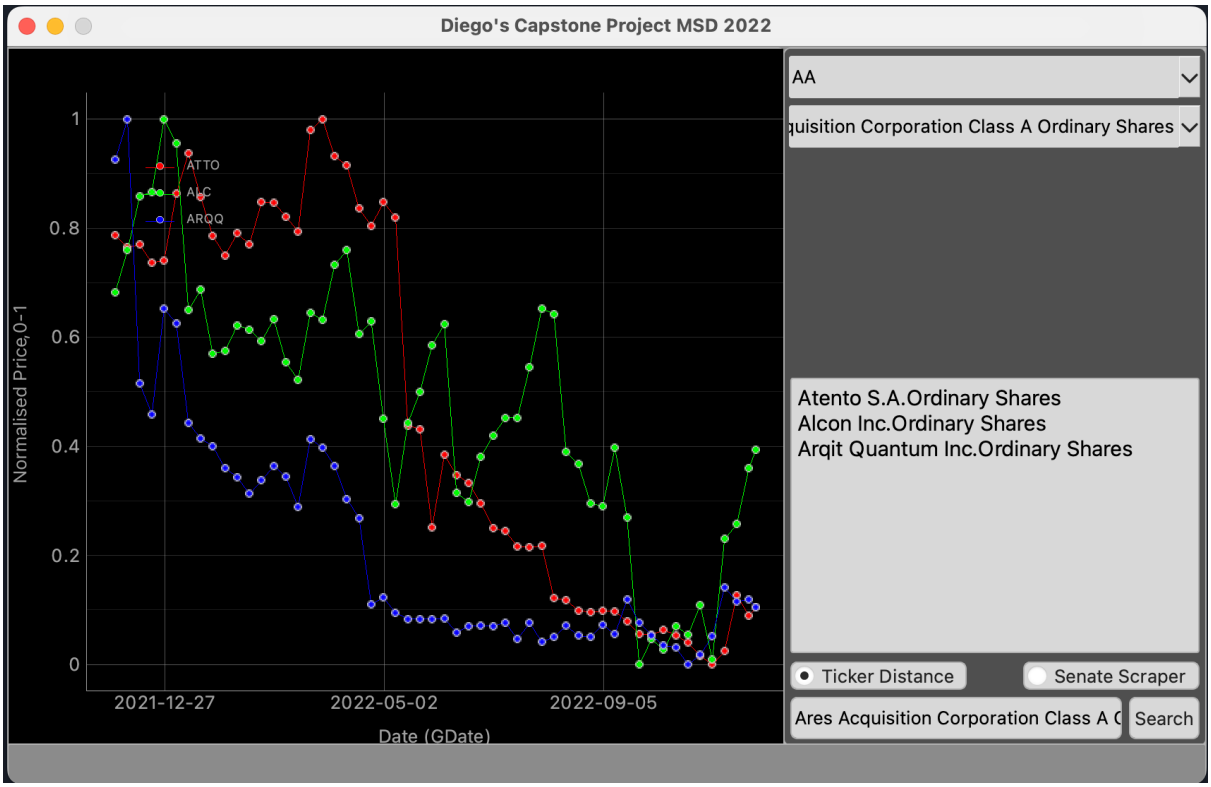

Figure 7 edit distance of listed companies

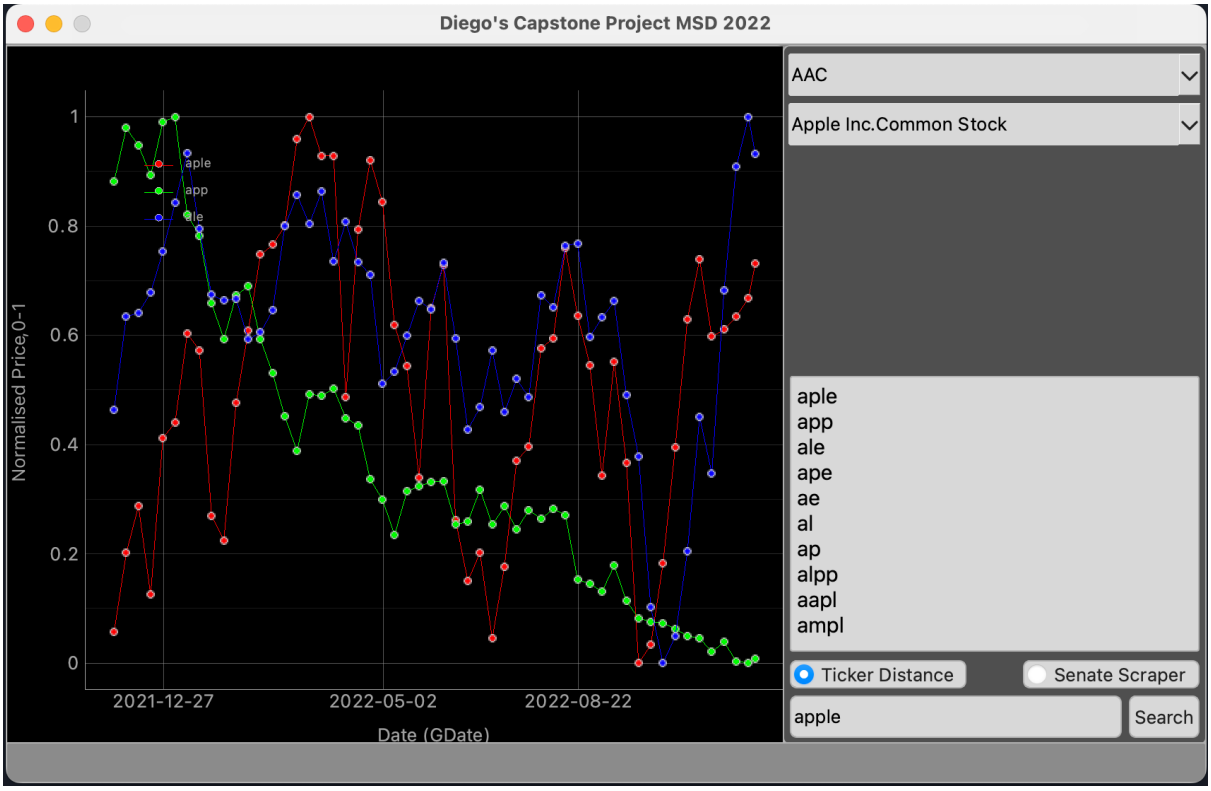

Figure 8 edit distance of custom search returning tickers

When the user enters their search term, the application tells the user the best matches based on a combination of edit and Euclidean distance of the stock and keyboard keys respectively. Distarb then graphs the normalised price data for the stock they searched for as well as the top two results. I chose to normalise the data because a detail I forgot to consider at first is the possibility that stocks may be wildly different prices. A silly oversite, but one that needed to be solved to make the app more user friendly and clear. With normalised prices, all the stock prices can be displayed cleanly in the window.

Senate Scraper

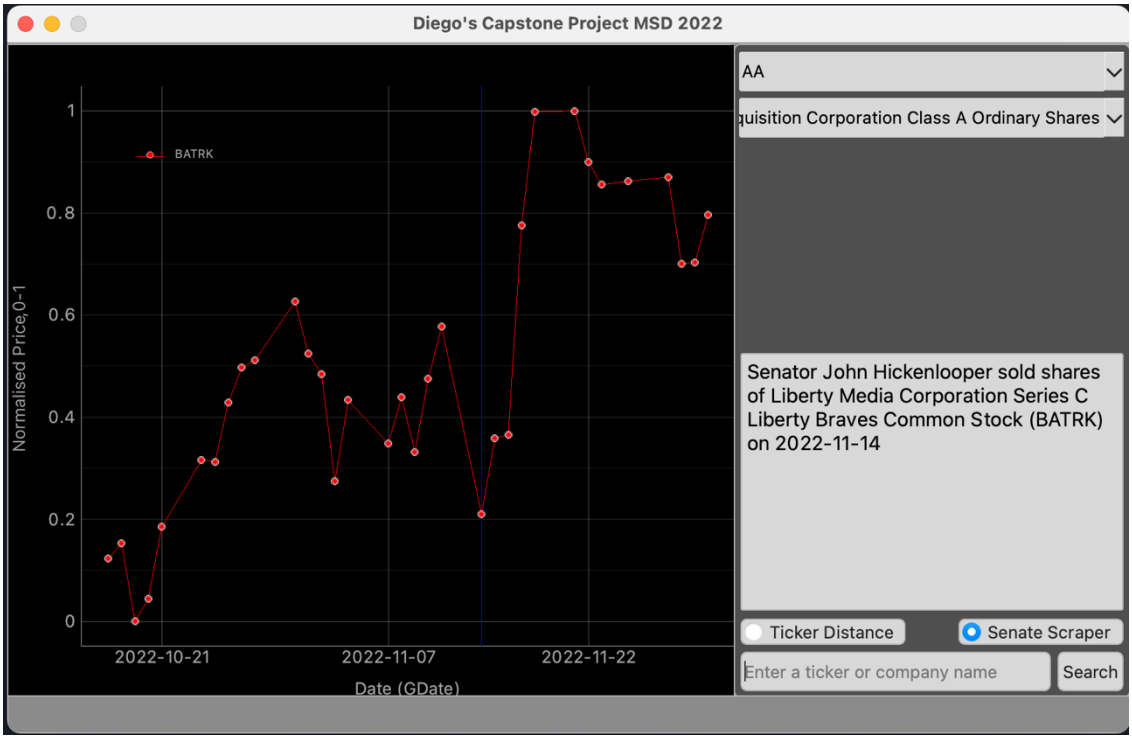

Figure 9 User has selected the scraper radio button

The program has a web scraper that scrapes the SEC's website. Senators are required by law to disclose what financial transactions they make. Ideally this would help deter them from capitalising on their ability to control the market. This app searches for the most recent stock transaction, and reports to the user the "who", "what", and "when" of the transaction. The program stores the entirety of the data from the SEC website, allowing for future versions to perform more data analysis. More of this in the conclusion. Distarb then charts the price data of the stock from four weeks before the purchase (marked by the blue line) up until the date the scraper was used.

Error Checking

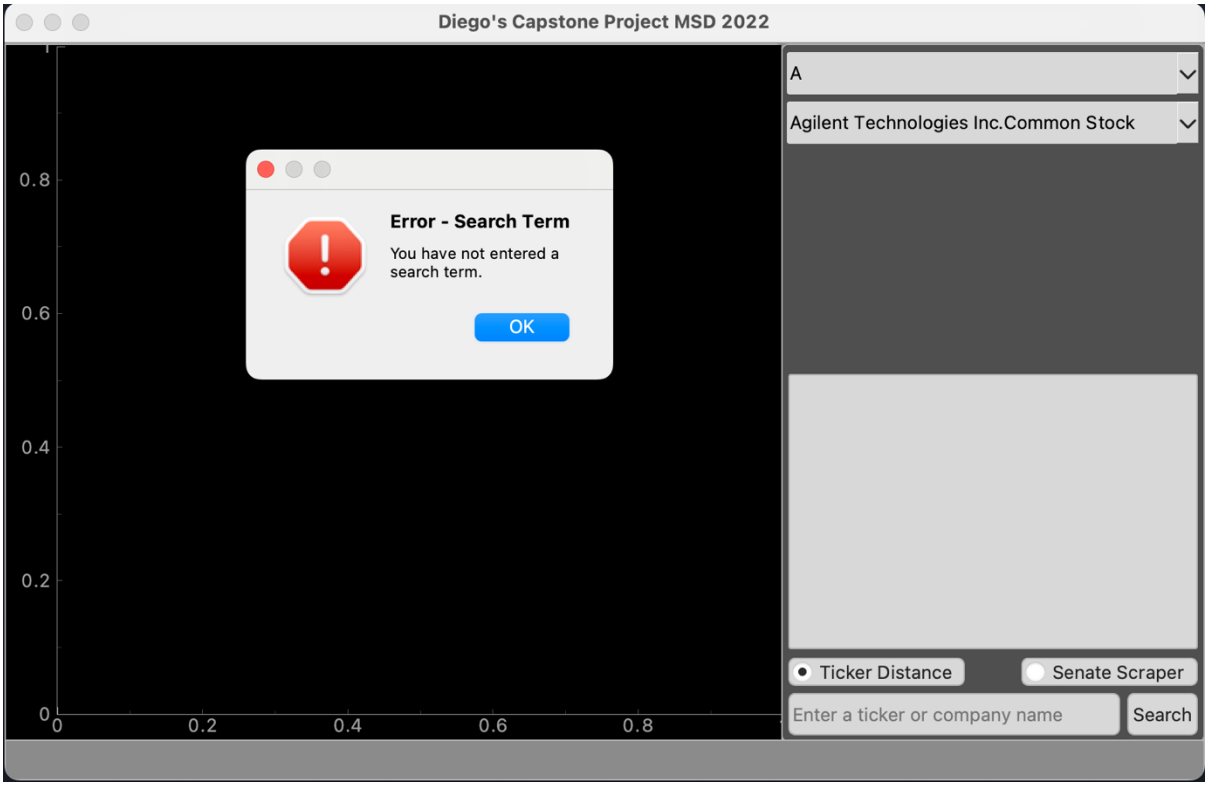

Figure 10 Empty search box error

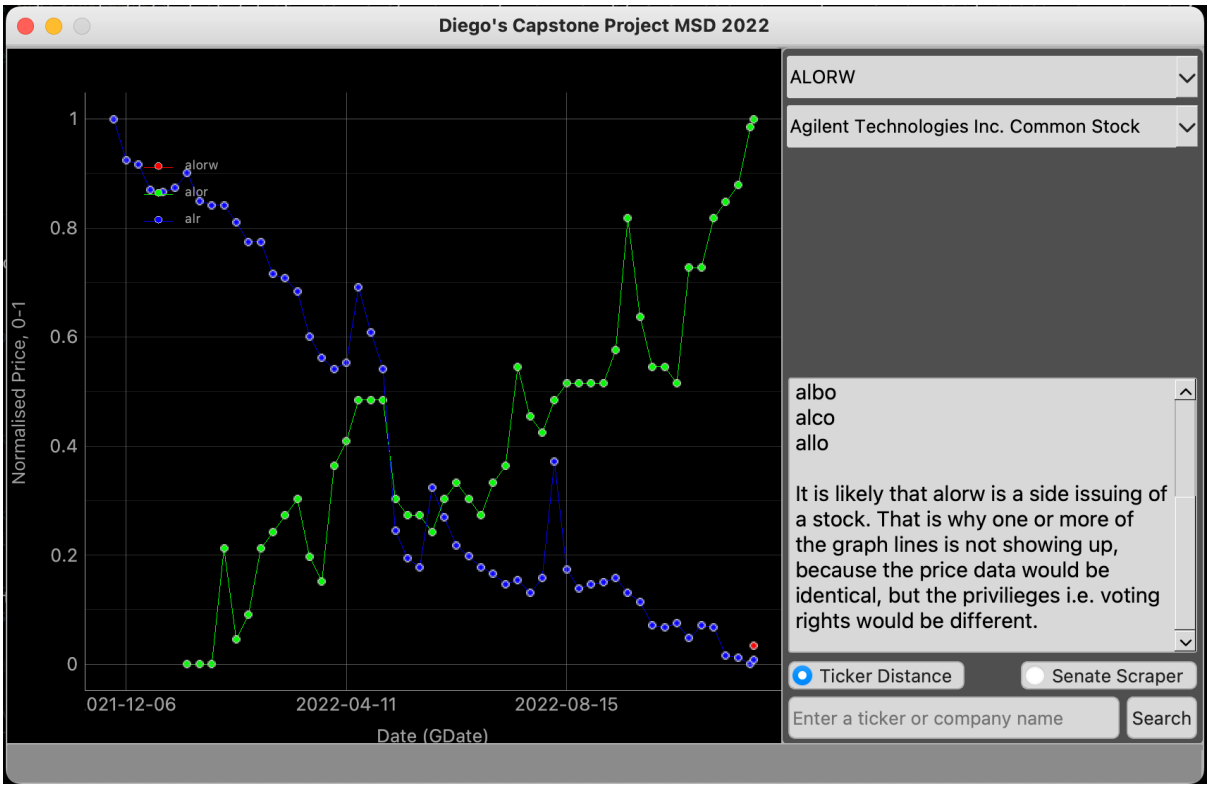

Figure 11 Side issuing of stock error

A concerted effort was made to make errors as clear as possible. For the error in figure 11, rather than take the time to sanitise my ticker list to look for every possible instance of multiple stock issuances, the program just explains why there are not as many price graphs the user might expect in the case where price data is identical between any number of stocks being checked. This is reasonable because there are many ways the same stock can be described. Warrant, series a, series b common – the list goes on. Rather than try to get them all, inevitably miss, and later have to start all over should I update my list, I chose to just let the program reasonably give an explanation that covered each case as they arose.

Successes

I think a lot of things went really well in this project. Python helped in making the code self-documenting, PyQt5 made for a fast GUI because of the underlying

code written in C++. I think the app having a simple front end makes it very user friendly too. For the SEC scraper, I was concerned for a time that the SEC website would eventually stop me from scraping. But, because the user likely will only be scraping once, maybe twice, there is enough time between scrapes, that this is not an issue. For further development, the scraper downloads and stores everything from the webpage. So further analysis that may be done on the data can work locally without the need for repeat visits to the webpage.

Failures

Something I am not happy with is how the stock legend displays. The legend is in the top left, and depending on the stock activity, it gets muddled in with the price graph. I was unable to find a way to reliably move it away from the lines to be readable. Having the list of edit distances to the side helps somewhat with understanding. But, it is less than ideal. I acknowledge that this issue may be operator error and PyQt5 has functionality for this that I just was not able to find.

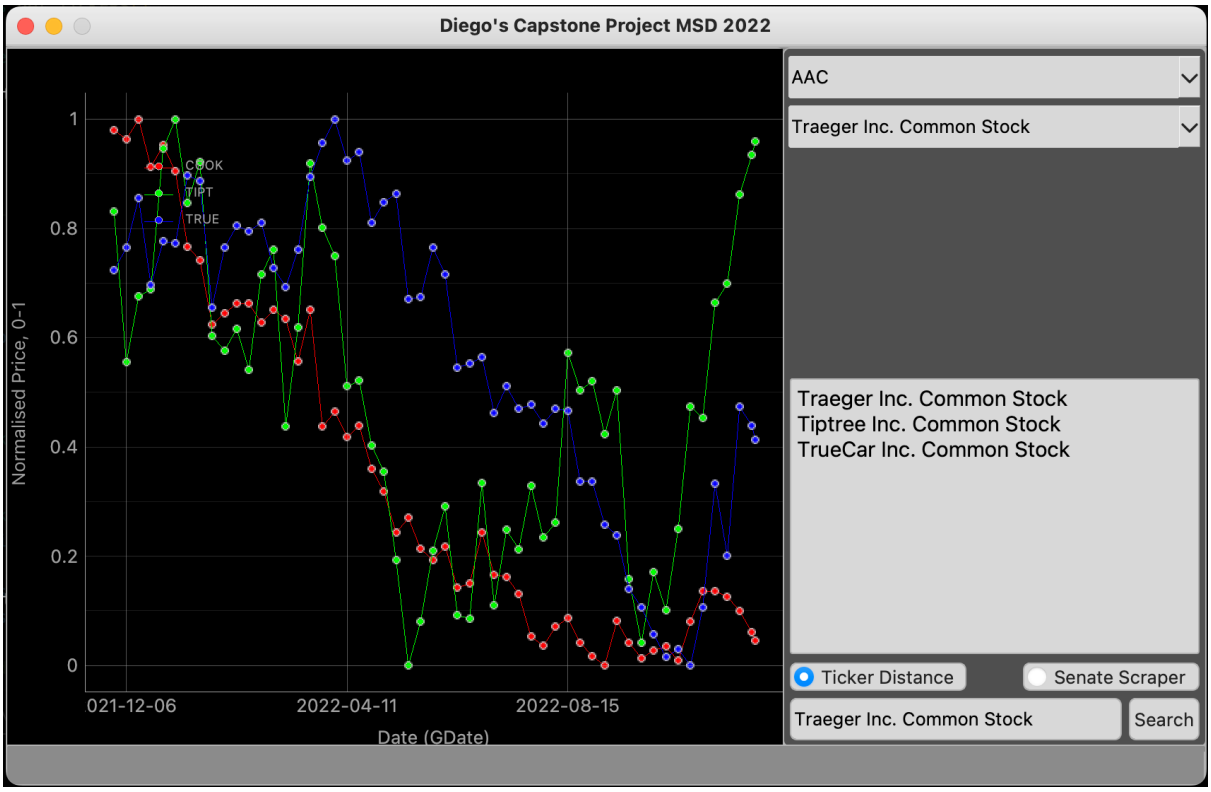

Figure 12 Graph legend is covered by unpredictable data

Something else I am not entirely satisfied with is that I was unable to set network permissions of the app for the packaged version to run off the bat from the GitHub release executable. I have a workaround using bash and the command line for running the application. The details of how to use it are laid out in the README's section "Why doesn't this work?". But, the crux of the solution is that the terminal can circumnavigate permissions.

This is less than ideal and rather disappointing. I have also provided a clear walkthrough and links to everything a user might need in the README in the event that all else fails. I am sure that with more time I could create a more elegant i.e. user friendly solution. For now, I believe I have made a reasonable, good faith effort to provide the user with tools to remedy a potential issue.

Conclusions

I have made the right design choices. An issue I may have later that might change my mind is maintaining the app in the event that the HTML changes on the site that it scrapes. There is not much that I can do to mitigate the amount of work that would need to be done to keep the HTML in the same format, as that is decided on by whoever is developing the website. However, something that I think might be interesting and more user friendly is to make the application a web app. I think that for future versions of this app, I will adapt it to a web app using flask or maybe django. This will allow me to continue using python libraries and a lot of the code I have written, without worrying as much about packaging the end product.

As mentioned above in *design choices* there are some potential issues regarding PyQt5 in the event that I try and make my app anything other than open source (Although as of now, it is licensed as open source under an MIT license). If I want to monetise my app, I would need to pay a rather steep fee to continue using PyQt5. Given that one of the industry problems I discussed is the cost of such applications, I would be a hypocrite if I tried to charge people a large amount to use it. Generally speaking, I am uncomfortable with the idea of charging any amount for a financial service that is built with helping people in mind. That said, I do have to put food on the table. Reworking Distarb as a webapp would be a good compromise to keep costs down for both the user and myself, and still provide a useful service.

Eventually, I would love to implement machine learning and run millions of tests on thousands of past IPOs against the entirety of the stock tickers list, weighing each key distance and each addition/deletion, and each price point (e.g. whether we

are more prone to mistakes on a stock with a higher or lower price point) to performances of similar tickers until my program had determined in what way we humans are most prone to error and could return stocks that fit that profile. The part of this that would be easy is that the code is built to easily adjust weights of adds/deletes in edit distance. However, I do not believe I will be able to make that many API calls using Yfinance in a narrow window that machine learning can make calls while adjusting the weights. That is a problem I can try and solve another time. What I like about this project is that there are heaps of opportunities to try something cool and interesting that I have learned in MSD.

Works Cited

(n.d.). Retrieved from pypi.org: https://pypi.org/project/yfinance/

Halford, M. (n.d.). *clavier*. Retrieved from github.com: https://github.com/MaxHalford/clavier

KindPNG. (n.d.). Retrieved from https://www.kindpng.com/downpng/iRJiiJh_stonks-meme-transparent-hd-png-download/

Martel, J. (2022, July 05). *investopedia*. Retrieved from https://www.investopedia.com/best-bloomberg-terminal-alternatives-5114517

SAMUELSSON, A. (2017). *Weighting Edit Distance to Improve Spelling Correction in Music Entity Search.* KTH Royal Institute of Technology , Computer Science, Stockholm.

Securities and Exchance Commision. (n.d.). *Latest Senate Stock Disclosures*. Retrieved from https://sec.report/Senate-Stock-Disclosures

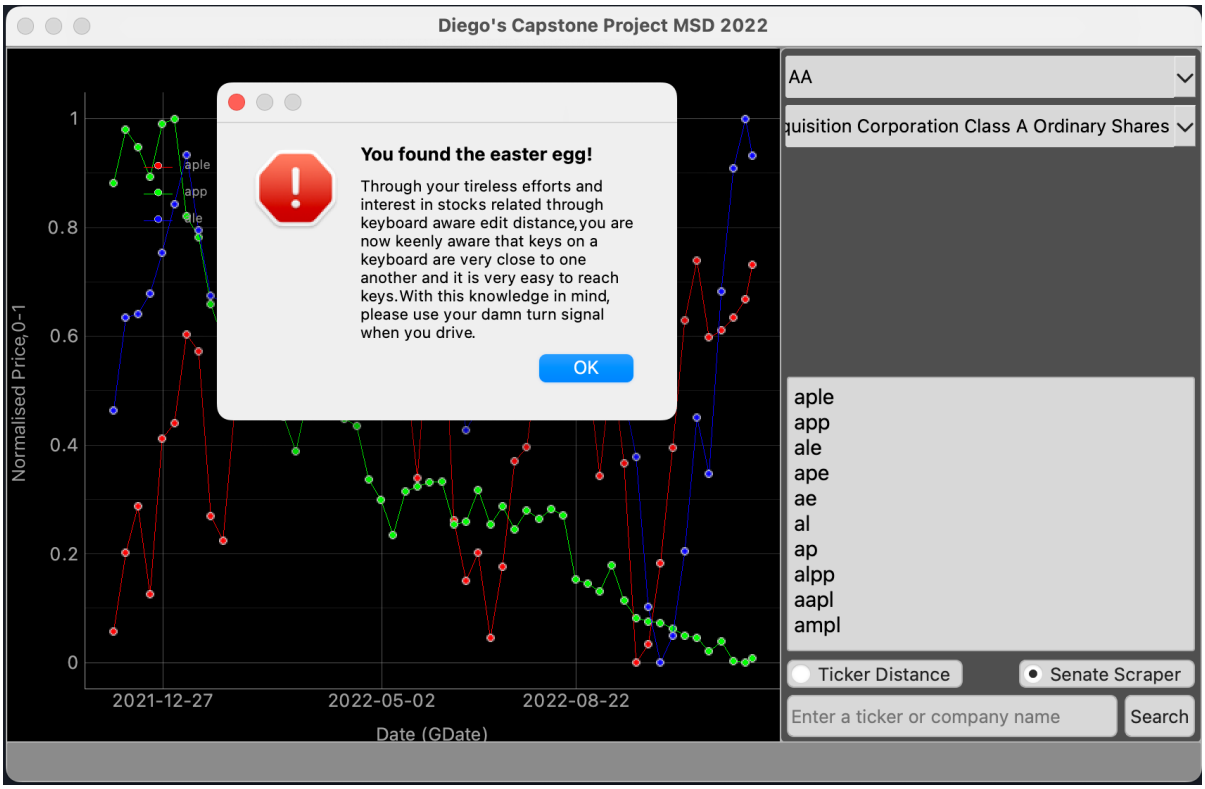

Figure 13 Just do it.