CPSC 537
CPSC 437 / 537

Chris Harshaw, Daniel Keller, Felipe Pires

December 3, 2016

1 Motivation

Motivation of paper-trading.

Describe main contributions of our system at a high level. Compare to existing systems.

2 Functionality and Walkthrough

3 Recommender Algorithms

4 Database Design

What we are storing.

How we are getting it.

ER Diagram with discussion.

5 Front-End Design

Discuss design discussions in front-end design.

Describe which tools were used.

6 Main Issues

Data fidelity was the main issue that we faced - both missing data and multiple inconsistent data points. For instance, sometimes the Tiingo API would be not have stock prices for certain days. There were also instances where Tiingo API returned several prices for a stock on a given day. This is likely a result of the aggregation of the sources that they pull from, such as Quotemedia and Quandl. The issues of missing and multiple data points pose a serious threat to the success of our recommendation algorithms.

To overcome these issues, we implemented data cleaning methods at several levels. First, when multiple stock prices were returned for a single day and stock, we removed all but the most recently pulled price. In this way, we cleaned our database as inconsistencies were found. Moreoften, it was the case that we were missing data. For instance, only a few stocks reported prices on Thanksgiving Day 2016. Our data cleaning method for dealing with missing data was handled on the front end and involved either (i) interpolating data (ii) throwing away entire days / stocks in computations. Essentially, if only few data points were missing, then the data was interpolated. On the other hand, if a sufficiently large amount of data was missing, then we completely removed a single day or a single stock in our computations. This front-end data cleaning method resulted in clean and well-founded computations.

Our system has several limitations. First, we only supported *static* portfolios. That is, our current framework assumes that stocks are added into a portfolio and kept there indefinitely. We do not support

the ability to buy and sell stocks within a portfolio. Fortunately, our framework can be generalized to support dynamic portfolios. without too much trouble. The second system limitation is the sacalability of our forecasting models. We currently use Vector Autoregression Models (VAR), whose parameters grow quadratically in the number of stocks used. For this reason, we only recommend portfolios from the most popular stocks. A more scalable algorithm would allow us to recommend portfolios from a larger range of stocks.

7 Division of Labor

Chris Harshaw did the initial domain research to understand relevant models, metrics, and methods. Chris proposed the high level view of the app, developed and implemented the recommender algorithms, developed back-end interfaces, wrote data-cleaning methods, and helped to debug back-end issues. Daniel Keller carried out the initial research to determine which API to use to gather stock prices. Daniel designed the database, wrote the database code, debugged back-end issues, and wrote initial front-end code. Felipe Pires designed and implemented the finalized front-end user interface in addition to writing front-end to back-end interface.