CORRELATION ANALYSIS:
STOCK PRICES AND GOOGLE TRENDS SEARCH VOLUMES

Naomi Nakagawa, Alvin Kuo & Vladimir Zinkovski - June 2023 [Notebook][Data]

MOTIVATION

This project aims to explore the correlation between stock prices and Google Trends search query volumes[1]. Investors are constantly in pursuit of information through which they can gain an edge and today's age of big data brings about many such potential new sources. Google Trends records search query volumes from its entire ecosystem of services, including from its leading internet search engine - Google Search.

We hypothesize that search behavior may be a leading indicator with regards to stock price movement. Specifically, we might expect an increase in search query volume to precede a price move in either direction [4]. Search query volume may thus be either positively or negatively correlated with a particular stock price changes, depending on their implied meaning, as well as their relation to the stock and the broader sector in which the stock operates. Conversely, declining search query volumes may manifest as relatively flat or slowly trending markets.

1.Google also uses alternate language to describe search query volumes and may apply the terms: search query index or search query popularity. Additionally, search query, search term and search keyword are all used interchangeably.

Our first step is to identify any potentially significant strong correlations between stock and search query pairs. The next step is to make use of visual techniques in order to inspect whether changes in search query volume precede changes in stock price. Being able to visually identify such leading patterns may help reduce the likelihood of reverse-causality, although a statistical test must be performed to reach a robust conclusion - which is outside the scope of this work.

Once we identify both a strong correlation and a clear leading pattern, we cannot yet assume causality. **Our project is an initial exploration of the topic and should we find visual evidence to support our hypothesis,** then it could serve as the basis for subsequent work which will need to apply additional statistical tests for establishing causality between two time series, such as the Granger causality test [3].

Our project can thus be incorporated into a larger pipeline to identify correlations and visually checks leading indicator potential, the next phase tests for causality, eventually this is fed into a comprehensive trading strategy [2]. Ultimately, creating a profitable trading strategy is often an end goal for both retail and professional investors and this project was undertaken to serve as a exploratory first step in that process.

SCOPE

In order to narrow our stock universe to a manageable size, **we select stocks listed on the S&P 500 and consider three sectors: finance, healthcare and technology**. Choosing the S&P 500 only means we are considering US listed stocks only. It is better with where Google is the preferred search engine [5]. With three sectors, although not representative of the entire economy, serves as a launching point for our investigation. **We restrict our scope to a 10-year date range**. It reduces the spurious short-term correlations and provide a robust exploration of our hypothesis. We also consider how correlation robustness is affected by time, sector & company size.

LIMITATIONS

Google Trends does not accurately reflect neither the retail nor professional investors as they may use additional data sources. Different countries prefer different search engines and even within the same country people may use different search engines.

DISCLAIMER

This project will not: perform statistical tests to establish causality, build a predictive model, formulate a trading strategy nor backtest its profitability.

DATA SOURCE: Alpha Vantage API
(Stock price data)

Alpha Vantage provides enterprise-grade financial market data through a set of free APIs. It covers real-time and historical market data from traditional asset classes, economic indicators, foreign exchange rates to commodities. Alpha Vantage is an open source database founded by Harvard Business School graduates. Unlike Yahoo finance, Alpha Vantage explicitly permits personal use in its terms of use, which satisfies ethical standards and ensures repeatability/reproducibility. The OHLC (Open, High, Low, Close) price and volume data are available through the API. We used daily adjusted closing price data, which takes into account dividends, stock splits, and new stock issuances.

DATA SOURCE: Pytrends API
(Search term data)

Pytrends API is an unofficial API for Google Trends. It's a free API. It allows simple interface for automating downloading of reports from Google Trends. Google Trends data is a public, time-series data of the Google search keywords popularity. We pull a specific 10-year of the US data for 3 categories: finance, healthcare and technology for search terms popularity index. *The Google Search Index (GSI) is a relative figure relative to the maximum value normalized between 0 to 100*. We use the web search and all categories without any specific category. It follows the definition of the search from Google trend.

API Name	Alpha Vantage API	Pytrends API for Google Trends
Company Info	[Alpha Vantage]	[Pypl][Google]
API weblink	[Alpha Vantage API Doc]	[Pytrends API Doc][Google Trends]
Returned format	**csv**	**csv**
Variable	Adjusted closing prices (daily)	terms (Search Terms Popularity)
Data Type	**float** between 0 to infinity (<5,000)	**float** between 0 to 100
Data Description	**Actual data**	**Sample Data**[1]
Version	**PyPi** (alpha-vantage 2.3.1)	**Pytrends** API 4.9.2 (Latest release: April 13th, 2023)
Records quantity to be used	Around **80,000** data points	Around **110,000** data points
Time period covered	**1999** ~ today	**2004** ~ today
Time period to be used	**2013-06-15 ~ 2023-06-15**	**2013-06-15 ~ 2023-06-15**

1. Note that Pytrends API returns slightly different data due to the sampling methodology: Impacts on correlations are limited.[More]

DATA MANIPULATION METHODS: ALPHA VANTAGE

Description of source code workflow

We instantiate a TimeSeries object by calling the TimeSeries() method of the alpha_vantage library. Inside this method we also set parameteroutput_format='pandas' which will allow us to subsequently return stock data in tabular format. By default this will also set the index as a pandas DatetimeIndex.

We instantiate an empty pandas DataFrame which will be built up iteratively through fetching data for each individual ticker (i.e. stock) separately through a for loop.

The API rate limit is 5 calls per minute, i.e. we can fetch data for 5 stocks per minute. In order to handle this programmatically, we embed a try-except block.

The try block will call the .get_daily_adjusted() method of the TimeSeries object and we specify that we want outputsize='full' which will return the full-length time series of 20+ years of stock data starting from June 2000. The returned DataFrame contains columns for open, high, low, close, adjusted close - all of which are price data. It also includes columns for traded volume, and dividend info.

We are only interested in the adjusted close column which is the same as closing price but accounts for any potential historical stock splits and other such corporate actions. The column contains approx. 252 timestamps per annum as this is the number of trading days after accounting for weekends and US public holidays.

After fetching data for 5 stocks, the except block is invoked because the try block will return an error due to having exceeded our API rate limit. We use .time.sleep(60) to pause our code execution for 1 minute in order to refresh our API limit.

The adjusted close column is appended to our empty DataFrame for each ticker. The concatenation happens along the column axis resulting in a DataFrame where each column is the adjusted closing price data for one specific ticker. Rows are easily matched up on the basis of having corresponding DatetimeIndex values.

The last step is to optionally specify the date range by slicing with the .loc[] method, where the row parameter takes either 'YYYY-MM-DD' or 'YYYY-MM' or 'YYYY' as the index value.

All of the above steps are wrapped into a function which takes three parameters: a list of tickers, start date, and end date. The ticker list is iterated over inside the function, the start and end dates are optionally passed to .loc[] slicing.

Handling missing and erroneous data

Fortunately, this API is well maintained and contains neither missing, nor erroneous data.

We can verify this on an ad-hoc basis through calling the .describe() method on the adjusted closing price column for a few randomly selected stocks which helps check for any outliers by comparing the min and max values against the mean and its standard deviation. Similarly, missingness can be inspected by calling .isna().sum() on the adjusted price column to check for the presence of NaN values. Spot-checking with this approach confirms the tidiness of this data source.

Another - entirely visual - option to verify both whether we have any erroneous and/or missing data is through plotting the price time series as a line chart. In which case erroneous values would show as spikes/throughs on the graph and missing values can be displayed as discontinuities.

DATA MANIPULATION METHODS: PYTRENDS

Description of source code workflow

We instantiate a TrendReq object by calling the TrendReq() method of the pytrends library. There are a number of default parameters in this method which we can leave as-is, such a hl='en-US' which sets the host language to US-English.

We instantiate an empty pandas DataFrame which will be built up iteratively by fetching data for each individual search term separately through a for loop. Note that a search term may be single-word or multi-word and our project includes instances of both cases. In order to perform exact matching on multi-word search terms we need to put the full term inside quotes. However, we opt not to take this approach as we make an assumption that most users of internet searches do not perform exact matching when, for example, searching for generic terms such as interest rate, economic growth, mental health or cloud computing.

Using a for loop we iterate over each separate search term. Inside the for loop, we call the TrendReq object .build_payload() method and pass it three arguments. First, the search term (note: can be either single or multi-word). Second, we set geo='US', which restricts the geographical location to USA only which means that only searches performed from USA based IP-addresses are considered when computing the search popularity. We make a key assumption here in that US-based individuals, both retail and professional investors, are more likely to invest in US-listed stocks. If we had set the geographical location to world (geo=''), then we introduce a form of skew where users from, for example, the UK or Australia search for English-language terms but may end up investing in similar respective domestic stocks. Third, we pass the the desired time range as timeframe='YYYY-MM-DD YYYY-MM-DD' with a start and end date.

Depending on the length of the timeframe, the frequency of timestamps returned differs in the following way: less than 9 months is daily, between 9 months and 5 years is weekly, more than 5 years is monthly. Our analysis ranges over periods both less than 5 years and more than 5 years and as such we account for these frequency mismatches - which is explained further on.

After we have built our payload, we call the .interest_over_time() method on the TrendReq object which returns a pandas DataFrame containing the search term popularity score in the first column, which is the sole column we need. The default index is a DatetimeIndex.

The search term popularity score column is appended to our empty DataFrame for each search term. The concatenation happens along the column axis resulting in a DataFrame where each column is the popularity score for one specific search term. Rows are easily matched up on the basis of having corresponding DatetimeIndex values.

All of the above steps are wrapped into a function which takes three parameters: a list of search terms, start date, and end date. The search term list is iterated over inside the function, the start and end dates are passed as the timeframe argument inside to the .build_payload() method.

Handling missing and erroneous data

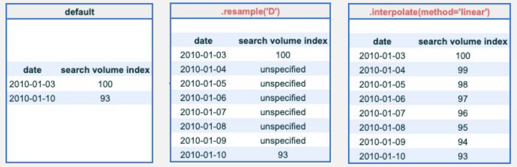
Now we address how we solve the frequency mismatches mentioned earlier.

This is achieved via a combination of a) the pandas .resample() method, which allows us to both upsample or downsample time series data to different frequencies, and b) the pandas .interpolate() method which allows us to fill missing values by interpolating from the nearest adjacent non-missing values from both ends.

In our case, we want to upsample our time frequencies to daily - by using the 'D' time offset alias for both scenarios where our original data comprises weekly or monthly timestamps. The .resample() method returns a DatetimeIndexResampler object. Although this object cannot be strictly shown in tabular form, nevertheless a helpful way to think of it is illustrated in the 'resample & interpolate' table. The idea is that an operation needs to be performed on the newly created rows in order to fill their values. For example, if we had weekly timestamps at '2010-01-03' and '2010-01-10', then the result is that we now also have timestamps (i.e. new rows) for each day in between with as yet unspecified values. Just to be clear, these are not strictly NaN values.

In order to fill these values, first an operation has to be performed which computes these values. Our computation is performed via .interpolate() where the default approach is method='linear'. This treats the rows with empty/unspecified values as equally spaced and fills these as shown in the 'resample and interpolate' table.

Table: Resample & Interpolate (toy data)



default

date	search volume index
2010-01-03	100
2010-01-10	93

.resample('D')

date	search volume index
2010-01-03	100
2010-01-04	unspecified
2010-01-05	unspecified
2010-01-06	unspecified
2010-01-07	unspecified
2010-01-08	unspecified
2010-01-09	unspecified
2010-01-10	93

.interpolate(method='linear')

date	search volume index
2010-01-03	100
2010-01-04	99
2010-01-05	98
2010-01-06	97
2010-01-07	96
2010-01-08	95
2010-01-09	94
2010-01-10	93

Using this approach makes a strong assumption that values for those days in between our weekly or monthly timestamps were indeed more or less linear.

This assumption is almost certainly not true, however it does not invalidate our analysis as it still captures the overall trend in search term volumes - and thus allows us to test our hypothesis whether changes in this trend can in fact be a leading indicator with respect to changes in stock price.

Google Trends data is clean and natively accounts for missing values in the following way: “a score of 0 means there was not enough data for this term”. We have no objective way to test whether values for certain observations are erroneous as values are capped in the 0-100 range, but they would likely appear as excessive spikes or throughs respective to neighbouring values if inspected visually through a line chart.

DATA MANIPULATION METHODS

JOINING THE DATASETS

As a reminder, our primary dataset is the stock prices dataset and our secondary dataset is the Google Trends search term volume index dataset. This is important as we are interested in understanding how a change in search volumes results in a change in stock prices - not the other way around. Thus, we would like to avoid any downsampling or upsampling of the primary dataset as both operations distort our original information.

Downsampling would reduce the count of observations and therefore potentially reduce the power of statistical tests in any subsequent work testing for causality. Upsampling produces synthetic data, which is not reflective of the real world, such as having closing prices on weekends or public holidays.

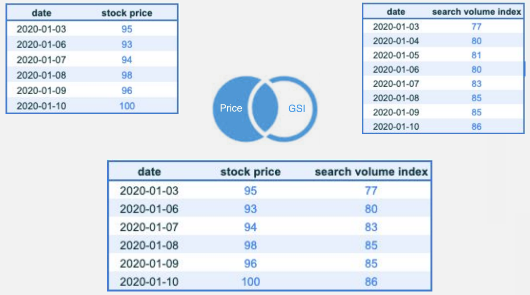
Therefore, the intention is to exactly match observations from our secondary dataset to those in our primary dataset.

In order to achieve this we perform a database-style join[1] through the pandas .merge() method. Our primary dataset is passed as the left dataset, the secondary dataset as the right dataset. Given both datasets have a DatetimeIndex, we can join on their indexes. The type of merge is a left-join where non-matching observations from the right table are discarded. The resulting merged dataframe contains all of the original data from the stock prices dataset and all observations from the search volumes dataset which were matched on date.

1. We use join and merge interchangeably.

The merge operation does not result in any missing values in the combined DataFrame. However, from a data hygiene perspective, we may consider removing all rows with missing values nevertheless through the pandas .dropna() method. Possibly eliminating some rows in this manner will not meaningfully impact our correlation analysis, and in either case pandas .corr() would ignore any pairwise correlation that has NaN values in one of the observations.

Table: Left Join[2]



date	stock price
2020-01-03	95
2020-01-06	93
2020-01-07	94
2020-01-08	98
2020-01-09	96
2020-01-10	100

date	search volume index
2020-01-03	77
2020-01-04	80
2020-01-05	81
2020-01-06	80
2020-01-07	83
2020-01-08	85
2020-01-09	85
2020-01-10	86

date	stock price	search volume index
2020-01-03	95	77
2020-01-06	93	80
2020-01-07	94	83
2020-01-08	98	85
2020-01-09	96	85
2020-01-10	100	86

2. Cell values are for illustrative purposes only

ANALYSIS & VISUALIZATION - OVERVIEW

SUMMARY

We analyzed correlations between stock prices and the Google Search Index (GSI) for several keywords for 3 sectors. In particular, we found that keywords related to corporate earnings and risk had consistently strong positive or negative correlations with stock prices regardless of time period. We also identified patterns between some stock prices and GSIs, which answered our initial question that search query volume may be positively or negatively correlated with changes in specific stock prices.

METHODS

We selected 10 stocks based on the following selection criteria: (1) 5 large- and 5 small-cap stocks, (2) relatively well-known stocks, and (3) stocks with relatively long trading records. For each sector, we selected 10 search terms including (1) industry terms, (2) trending terms, (3) earnings related terms, and (4) risk related terms. Lastly, correlations were tested over different time periods (3yr, 5yr, and 10yr) to examine the persistence of the relationship between keywords and stock prices. In this presentation, a 10-year period was used to illustrate the long-term trend.

ANALYSIS

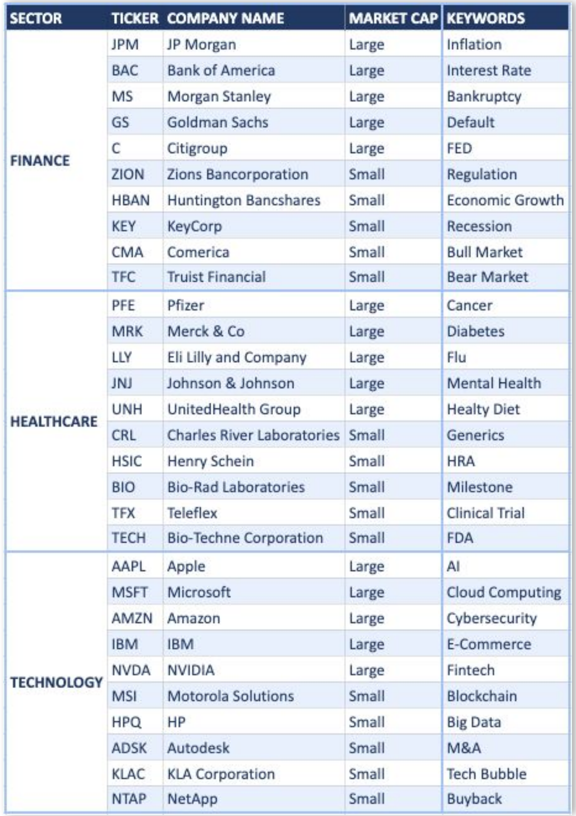
We analyzed the spearman correlation between stock prices and the GSIs because we focused on the direction and monotonicity (increasing or decreasing) of the relationship. Not all randomly selected keywords are statistically significant because there are many confounding factors, such as earnings and momentum, and there may not be a direct causal relationship between stock prices and GSIs. However, more than 95% of the keywords presented had a p-value less than 0.05.

VISUALIZATION

Three visualizations were used for the analysis: Barplots were used to analyze the magnitude of pairwise correlations between stocks and GSIs; Clustermaps were used to examine the similarity between stocks and GSIs; and, dual-axis Lineplots were used to investigate the patterns of movement of stocks prices and GSIs.

FURTHER RESEARCH

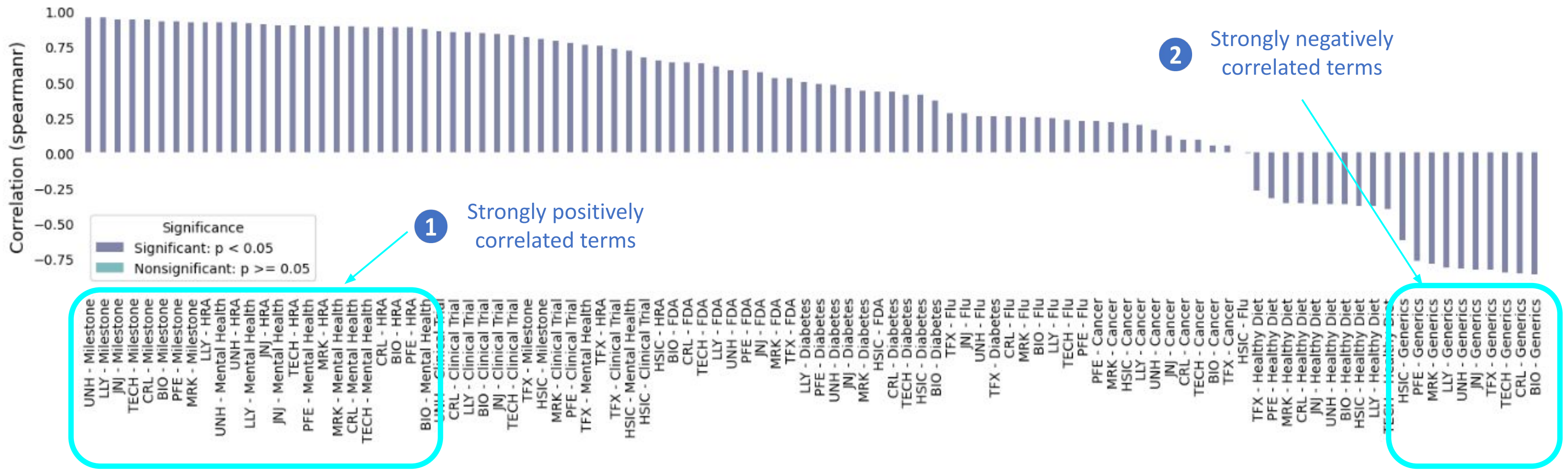
Further research could include (1) causality testing, (2) return and trading volume predictions with machine learning models, and (3) NLP / sentiment analysis using positive/negative search terms.



SECTOR	TICKER	COMPANY NAME	MARKET CAP	KEYWORDS
FINANCE	JPM	JP Morgan	Large	Inflation
	BAC	Bank of America	Large	Interest Rate
	MS	Morgan Stanley	Large	Bankruptcy
	GS	Goldman Sachs	Large	Default
	C	Citigroup	Large	FED
	ZION	Zions Bancorporation	Small	Regulation
	HBAN	Huntington Bancshares	Small	Economic Growth
	KEY	KeyCorp	Small	Recession
	CMA	Comerica	Small	Bull Market
	TFC	Truist Financial	Small	Bear Market
HEALTHCARE	PFE	Pfizer	Large	Cancer
	MRK	Merck & Co	Large	Diabetes
	LLY	Eli Lilly and Company	Large	Flu
	JNJ	Johnson & Johnson	Large	Mental Health
	UNH	UnitedHealth Group	Large	Healty Diet
	CRL	Charles River Laboratories	Small	Generics
	HSIC	Henry Schein	Small	HRA
	BIO	Bio-Rad Laboratories	Small	Milestone
	TFX	Teleflex	Small	Clinical Trial
	TECH	Bio-Techne Corporation	Small	FDA
TECHNOLOGY	AAPL	Apple	Large	AI
	MSFT	Microsoft	Large	Cloud Computing
	AMZN	Amazon	Large	Cybersecurity
	IBM	IBM	Large	E-Commerce
	NVDA	NVIDIA	Large	Fintech
	MSI	Motorola Solutions	Small	Blockchain
	HPQ	HP	Small	Big Data
	ADSK	Autodesk	Small	M&A
	KLAC	KLA Corporation	Small	Tech Bubble
	NTAP	NetApp	Small	Buyback

HEALTHCARE - Correlation Magnitude

We analyzed stock-keyword pairwise correlations using scipy.stats. Terms associated with increased earnings (Milestones[1], HRA[2]) have strong positive correlations with stock prices, while terms associated with decreased earnings (Generics[3]) have strong negative correlations with stock prices. Strong positive and strong negative correlations are similar across different market caps.



Data Source: Pytrends | Alpha Vantage
Note: The x-label is aligned vertically to display all search terms without overlapping
1, 2, 3: Milestone boost revenues. HRA (Health Reimbursement Arrangements) increases demand for healthcare services and prescription drugs. Generics leads to lower profitability.

TECHNOLOGY - Similarity & Dissimilarity

We used Seaborn's Clustermap to explore clustering patterns in stock-keyword combinations and identify underlying structures. The length and arrangement of the branches indicate the similarity and dissimilarity between the corresponding rows or columns. The height at which two branches merge represents the distance or similarity level.

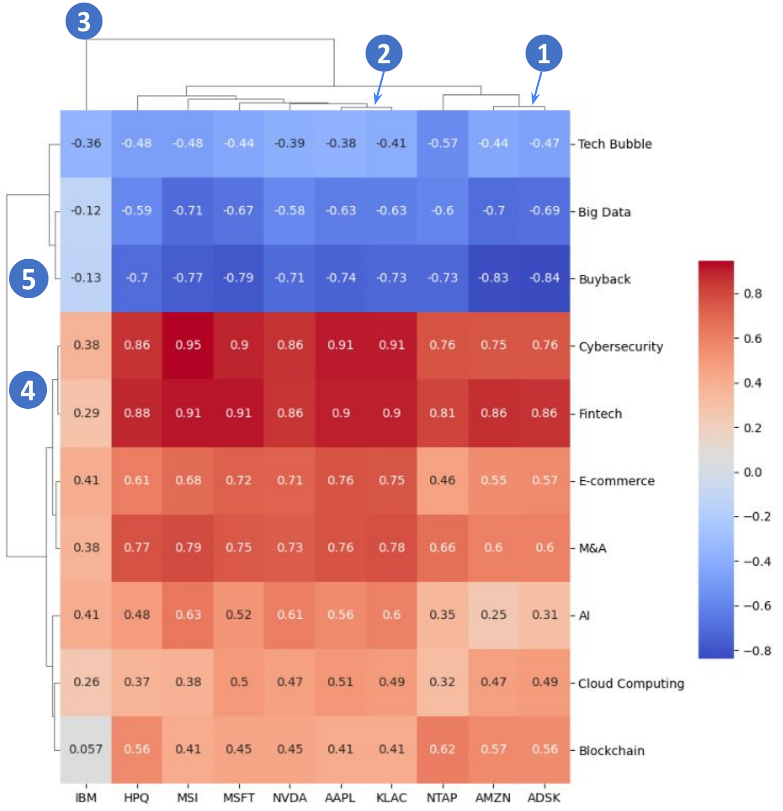
Stock Similarity

- AMZN and ADSK are similar in terms of keyword and stock price correlations
- Apple and KLAC are similar, although their market capitalizations are very different (1 2)
- **IBM, The Lone Wolf -** IBM reacts distinctly differently to keywords than other tech stocks (3)

Keyword Similarity

- Cybersecurity and Fintech are positive, while Buyback, Big Data, and Tech Bubble are negative (4 5)
- **Buybacks are negative** for Technology stocks - This is because buybacks signify that management has admitted that there is no more room for growth to invest cash in

Data Source: Pytrends | Alpha Vantage

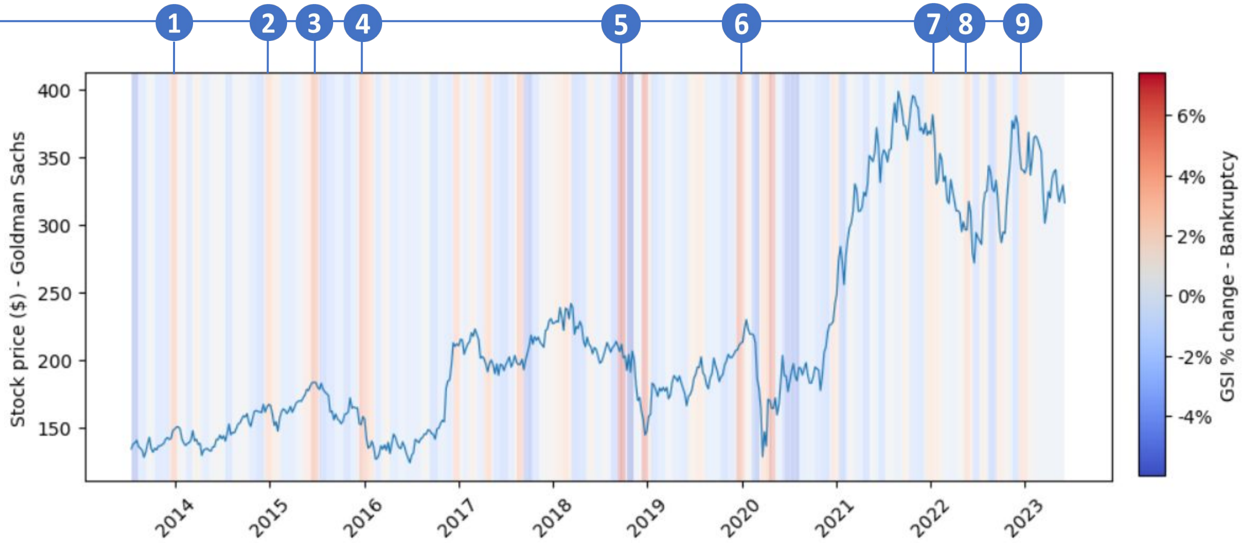


	IBM	HPQ	MSI	MSFT	NVDA	AAPL	KLAC	NTAP	AMZN	ADSK
Tech Bubble	-0.36	-0.48	-0.48	-0.44	-0.39	-0.38	-0.41	-0.57	-0.44	-0.47
Big Data	-0.12	-0.59	-0.71	-0.67	-0.58	-0.63	-0.63	-0.6	-0.7	-0.69
Buyback	-0.13	-0.7	-0.77	-0.79	-0.71	-0.74	-0.73	-0.73	-0.83	-0.84
Cybersecurity	0.38	0.86	0.95	0.9	0.86	0.91	0.91	0.76	0.75	0.76
Fintech	0.29	0.88	0.91	0.91	0.86	0.9	0.9	0.81	0.86	0.86
E-commerce	0.41	0.61	0.68	0.72	0.71	0.76	0.75	0.46	0.55	0.57
M&A	0.38	0.77	0.79	0.75	0.73	0.76	0.78	0.66	0.6	0.6
AI	0.41	0.48	0.63	0.52	0.61	0.56	0.6	0.35	0.25	0.31
Cloud Computing	0.26	0.37	0.38	0.5	0.47	0.51	0.49	0.32	0.47	0.49
Blockchain	0.057	0.56	0.41	0.45	0.45	0.41	0.41	0.62	0.57	0.56

FINANCE - Goldman Sachs vs. Bankruptcy

We investigated how stock prices react to changes in the GSI using def plot_term_percent_price(). The keywords that are strongly negatively correlated with stock prices can serve as an early warning signal of risk. Financial stocks tend to fall after the GSI of "Bankruptcy" increases. While bankruptcy statistics are lagging indicators, the GSI is a potential leading indicator that can capture bankruptcy sentiment as a proxy for bankruptcy statistics. Investors may be able to incorporate the pattern into trading strategies.

- *Goldman Sachs stock tends to fall after searches for the word "bankruptcy" increase*
- *Bankruptcies increase bad loans and write-offs, squeezing company profits*
- ***The The GSI keyword search pattern can be a good indicator for trading strategies***



Data Source: Pytrends | Alpha Vantage

STATEMENT OF WORK

Specialization

Team, Theme Initiation & Operation: Alvin contributes idea initiation, Naomi streamlines the thought, Vladimir consolidates the direction and judgement. Vladimir leads the direction, sets calendar and rubrics, Naomi and Alvin monitor the office hour update and the peer progress

Project Proposal: Naomi for Primary Dataset and Analysis, Alvin for Secondary Dataset, Visualization and Ethic Considerations. Vladimir for Project Summary, Cleaning/Manipulation and Contribution

Code & Presentation:

Vladimir builds the main model in code and Motivation, Data Manipulation Methods and Spelling, grammar, and style.

Naomi also builds the main model blocks in code and Analysis, Visualizations and Spelling, grammar, and style in presentation.

Alvin for mark-down, and comment in code and Data Sources and Statement of work in presentation.

Assessment

Team & Theme Initiation: The common financial background makes the team highly capable to investigate about this investment theme regarding the stock price sensitivity to the search keyword terms. All three members has variou kinds of domain knowledge of investment banking from macroeconomics, industry, stock & commodity price, assets management to portfolio management to support analysis with nuance.

Meetings: The team handled international virtual had meeting across America, Europe and Asia with 3-4 time zones with mentor 6 times and team meeting 10+ times successfully. All meetings are mostly effective 1 hour meeting with actionable items.

Project Proposal, Code and Presentation & Code: Covered with each other as possible we the team can reach out within time frame. Every proposal, code and presentation got submitted on time with the best team member could perform.

Improvement

Collaboration platform and tool: To have more time to sharpen the capabilities to master teamwork collaboration platform and tools introduced or mentioned by instructor teams (Version control & Git)

Collaboration Assessments: The assessment of strengths and weaknesses of team members correctly before allocating who did what.

REFERENCES

[1] ***Da, Z., Engelberg, J., and Gao P., (2009).* In Search of Attention.** University of California, San Diego (UCSD) - Rady School of Management. *AFA 2010 Atlanta Meetings Paper*

[2] ***Bijl, L., Kringhaug, G., Molnar, P., and Sandvik, E. (2016).* Google Searches and Stock Returns.** *International Review of Financial Analysis*.

[3] ***Huang, M.Y., Rojas, R.R., and Convery, P.D. (2020).* Forecasting stock market movements using Google Trend searches.** *Empirical Economics*, volume 59, pages 2821–2839.

[4] ***Preis, T., Moat, H.S., and Stanley, H.E. (2013).* Quantifying Trading Behavior in Financial Markets Using Google Trends.** *Scientific Reports*, volume 3, article number 1684.

[5] ***Ying, L., Geng, P., Lanyi, H., Jichang, D., and Qingqing, Z. (2019).* Using Google Trends and Baidu Index to analyze the impacts of disaster events on company stock prices**. *Industrial Management & Data Systems*.